JOIV : Int. J. Inform. Visualization, 8(1) - March 2024 409-416

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Distributed Aerial Image Stitching on Multiple Processors Using Message Passing Interface

Alif Wicaksana Ramadhan [a], Fira Aulia [a], Ni Made Lintang Asvini Dewi [a], Idris Winarno [b,*], Sritrusta Sukaridhoto [b]

[a] *Electrical Engineering Department, Politeknik Elektronika Negeri Surabaya, Jl. Raya ITS, Surabaya, 60111, Indonesia*
[b] *Information and Computer Engineering Department, Politeknik Elektronika Negeri Surabaya, Jl. Raya ITS, Surabaya, 60111, Indonesia*
*Corresponding author: *idris@pens.ac.id*

***Abstract*—This study investigates the potential of using Message Passing Interface (MPI) parallelization to enhance the speed of the image stitching process. The image stitching process involves combining multiple images to create a seamless panoramic view. This research explores the potential benefits of segmenting photos into distributed tasks among several identical processor nodes to expedite the stitching process. However, it is crucial to consider that increasing the number of nodes may introduce a trade-off between the speed and quality of the stitching process. The initial experiments were conducted without MPI, resulting in a stitching time of 1506.63 seconds. Subsequently, the researchers employed MPI parallelization on two computer nodes, which reduced the stitching time to 624 seconds. Further improvement was observed when four computer nodes were used, resulting in a stitching time of 346.8 seconds. These findings highlight the potential benefits of MPI parallelization for image stitching tasks. The reduced stitching time achieved through parallelization demonstrates the ability to accelerate the overall stitching process. However, it is essential to carefully consider the trade-off between speed and quality when determining the optimal number of nodes to employ. By effectively distributing the workload across multiple nodes, researchers and practitioners can take advantage of the parallel processing capabilities offered by MPI to expedite image stitching tasks. Future studies could explore additional optimization techniques and evaluate the impact on speed and quality to achieve an optimal balance in real-world applications.**

***Keywords*— MPI parallelization; image stitching; distributed tasks; parallel processing; optimization techniques.**

Manuscript received 19 Jun. 2022; revised 4 Jul. 2023; accepted 23 Oct. 2023. Date of publication 31 Mar. 2024.

I. Introduction

Image stitching is a process in which multiple images are seamlessly combined to create a single, larger image. This technique has become increasingly popular in recent years, particularly in photography, where it allows for creating panoramic shots and other wide-angle views. Image stitching combines several images in the same scene into a single, high-resolution panoramic image. Image stitching requires the closest resemblance or match between overlapping images and identical exposures to produce a seamless image [1]. Stitching algorithms have recently been applied in many fields, particularly photography, creating wide field-of-view (FOV) videos for surveillance and assisting automobiles [2]. Image stitching can be very helpful in many aspects. In the microscopic image area, Boyuan et al. [3] create large panoramas of tiny structure images to get the characteristics information of a material, and Kaiyue et al. [4] produce microscopic images of ceramic structures to get scientific data. In health areas, image stitching can combine pictures of deformity surgery to get exclusive photos of a person's spinal image [5] or hyperspectral imaging [6]. In industrial areas, image stitching can assist inspection processes [7], [8]. More promising, in aerial monitories and mapping systems, Image stitching is also applicable. In [9] and [10], image stitching has been implemented to create a large-scale image of a farmland area.

There are many types of approaches to image stitching. One of the famous approaches is feature-based image stitching. The feature-based method establishes correspondences between points, lines, edges, corners, or other geometric entities [1]. Image stitching can be a very lightweight task or vice versa, depending on the case. The process can be very long for large-scale image stitching because of the surplus of the data processed. There are many approaches to improve the speed of the stitching process

based on the algorithm. One of the faster stitching approaches is using a GPS sensor [11]. GPS helps predict the feature point location to shorten the matching process and improve the results' accuracy. An iterative optimization approach is used to construct the transformation matrix between keyframes. According to experiments, the suggested technique outperforms the RANSAC algorithm [12] under the same feature regarding matching outcomes and computing time where 3–4 frames per second are the typical frame rate.

Problems facing low speed and low accuracy are often experienced at the feature extraction stage of aerial images. Another proposed method is to process aerial image mosaics to obtain fast and accurate image stitching results [13]. This study uses an improved method based on ORB features due to its calculation speed. The work process is carried out in four stages:

1. Comparing the performance and speed of ORB to the other algorithms.
2. Setting up thresholding based on descriptor similarity to produce fast and robust feature matching.
3. Using a progressive LMedS & LS algorithm to eliminate spurious matches.
4. Using a multi-band fusion algorithm to fuse the matched images and realize a panoramic mosaic.
5. The results show that the method used can increase the efficiency of the captured images while ensuring the stitching effect and reducing cumulative errors.

Another research combined two methods for better image stitching performance [14]. Those methods are the FAST-Tomasi feature, an efficient method for detecting corners in an image by computing the smallest eigenvalue of the gradient covariance matrix at each pixel, and Delaunay triangulation, a computational geometry technique that is used to triangulate a set of points in a 2D plane or 3D space. Utilize the updated shape-preserving half-projective (SPHP) method for registration after using the FAST-Tomasi algorithm for feature recognition to get feature points rapidly and accurately. Next, use Hamming distance for rough matching before swiftly removing duplicate and incorrectly matched points with Delaunay triangulation. Final experimental findings demonstrate the stability and speed of the feature points retrieved using the method in this work. Matching accuracy and matching efficiency have significantly increased because of the Delaunay triangulation feature point elimination technique.

Another faster and more efficient approach has been implemented for large-scale aerial image stitching [15]. This research improves the stitching performance by optimizing the size of overlapping areas of each image, so the images that will stitch are manageable. The optimization is done by utilizing an adaptive selection of the image set. The overlapping areas estimation also improves the feature detection and matching process. It determines whether it is necessary or not to do feature detection and matching. This research tested and compared the proposed method to other stitching software. The processing time for the suggested approach grows linearly, but the processing time for different methods, particularly for Pix4d, Hugin, and Photoshop, climbs considerably. To ensure a fair comparison, the pre-processing time of the suggested procedure is also included. The total processing time of the suggested technique is 67.1, 106.5, and 135.7 (seconds), respectively, for stitching 100, 150, and 200 photos with a resolution of 5472 x 3648.

Many methods are approached to handle the rising demand for high-performance computation, and one of those is by implementing parallel processing. The incredible improvement in single-processor performance resulted from integrated circuits using transistors as electronic switches in ever-higher densities. Transistors' speeds can be raised as their sizes shrink, which also speeds up the integrated circuit. Nevertheless, as transistors in the embedded chip/CPU speed grow, so does the energy they use. Most of this power is lost as heat, and an integrated circuit loses reliability if it becomes too hot [16]. Due to the mentioned problems, parallel processing is the solution. The industry has combined several simple, full processors on a single chip rather than creating ever-faster, more sophisticated, monolithic CPUs [17].

One of the numerous approaches to using parallel processing is implementing a message-passing interface (MPI) [18]. MPI is used to create message-based parallel programming. It is extensively utilized in applications for high-performance computing and allows communication between computers to complete a task over a network. Implementing MPI provides a unique approach to building software with a specific function. A wide range of hardware and software platforms support it. MPI concepts utilize communicators, data types, point-to-point communication and collective communication [19].

Many tasks, including big data processing, have been implemented using MPI-based parallel processing [20]. In contrast to the ones now in use, a unique approach to extensive data processing and management was put forth in this work. The suggested strategy leverages memory space for reading and handling vast data and memory-mapped extended memory storage. From a methodological perspective, this study is new in that it uses memory mapping to massive partition data and then uses a parallel message-passing interface to broadcast all segments to various processors. From an application perspective, the study provides a high-performance method based on a homogeneous network that encrypts and decrypts massive data using the Advanced Encryption Standard (AES) algorithm while operating in parallel.

Jiang et al. have used MPI in the robotics industry to improve the issue of the communication bottleneck in ROS2 [21]. They proposed an innovative method, the adaptive two-layer serialization algorithm, which can effectively communicate various messages. According to experimental findings, our algorithm can significantly outperform conventional techniques in ROS2 by up to 93% when using our framework. Another MPI implementation is optimizing the clustering process using K-Means [22]. In terms of overhead costs and execution, the performance of the K-means method for clustering the data is studied in this work between the sequential run and the parallel run in the design of message-passing interfaces.

Fajrianti et al. [23] discuss several scenarios of applying CUDA and MPI to train the 14.04 GB corn leaf disease dataset—the use of CUDA and MPI in the image pre-processing process in their research. The results of the pre-processing image accuracy are 83.37%, while the precision value is 86.18%. In pre-processing using MPI, the load

distribution process occurs on each enslaved person, from loading the image to cutting the image to get the features carried out in parallel. The resulting features are combined with the master for linear regression. In the use of CPU and Hybrid without adding MPI, there is a difference of 2 minutes. Moreover, in the usage between CPU MPI and GPU MPI, there is a difference of 1 minute. The result demonstrates that implementing accelerated and parallel communications can streamline the processing of data sets and save computational costs. In this case, using MPI and GPU positively influences the proposed system.

In this research, we proposed implementing MPI to improve the process of aerial image stitching. We used the MPI tool, LAM/MPI, to create the environment because of its high performance, free availability, and open source [24]. It was researched, developed, and maintained at the Open Systems Lab at Indiana University. Due to its stability and availability, we also used a DJI drone to take aerial pictures. The rest of this paper is organized as follows, the designs and methods of the system and the method of approach are described in Section II. Section III includes the experiment's setup, results, and discussion. Finally, we conclude our study in Section IV.

II. Materials and Method

This section shows the system design, topologies, data and server preparation. In this research, we implement parallel processing using MPI for aerial large-scale image stitching. The area of our university, Politeknik Elektronika Negeri Surabaya (PENS), is chosen for the stitching area. Raw images are obtained using drones that are flown over the university area.

A. Data Collection

Using a DJI drone, the raw images were collected autonomously. A DJI drone equipped for autonomous mapping typically includes a high-resolution camera and software that enables the drone to fly a pre-planned flight path while taking images. The specifications of the camera used, which is Hasselblad, can be seen in Table I.

TABLE I
CAMERA SPECIFICATIONS

No	Items	Specifications
1	Camera	Hasselblad 11d-20c
2	Sensor size	20MP 1”
3	Storage	FOV 77° (28 mm) f/2.2

In order to navigate and maintain its position to follow the pre-planned flight path while in flight, the drone would also usually have a GPS receiver and sensors such as an Inertial Measurement Unit (IMU) and a barometer. Figure 1 shows the pre-planned flight path while taking images. The drone was flying at the height of 110 meters, with the size of area 332 x 205 meters2. The approximated total time taken from the data collection process was about 25 minutes.

During the data acquisition process, overlapping is required to help overcome problems such as shadows, reflections, and noise that can occur in drone images taken with a flat perspective. With sufficient overlapping, image processing software can combine information from each image to produce sharper and more detailed images, as well as improve the quality of data measurement and analysis. In this research, the applied overlapping is 75%-80%.

Fig. 1 Image capture route planning

B. MPI (Message Passing Interface)

The Message Passing Interface (MPI) is a key component in managing tasks on multiprocessor computers. A common standard communication mechanism called MPI makes it easier for numerous processes operating on various distributed system threads to send messages to one another. Utilizing multiple processing capacities enables programmers to build parallel applications that can quickly handle massive databases and carry out intricate calculations.

One of the freely available standard implementations of MPI is LAM. Since its inception in 1989, the LAM project has developed into a mature code base that is feature-rich and effective in its implementation, providing MPI users and developers with high performance and convenience [25]. The utilization of LAM/MPI has given the system the ability to perform multiple tasks. However, the tasks are limited to one computer. In order to make the LAM/MPI run on a cluster network with multiple computers as nodes, the Network File System (NFS) is used [24]. NFS allows a computer to share its folder with another in a network cluster. After the LAM/MPI is installed and configured on all the computers that are members of NFS, the LAM/MPI will recognize the computers as nodes. The LAM/MPI and the NFS setup is shown in Figure 2.

C. Image Stitching

A preprocessing step was performed to prepare the dataset for image stitching to eliminate unwanted noise or artifacts. The method used to handle this is contrast enhancement or histogram equalization, a technique used to improve an image's contrast by adjusting the pixels' intensity levels. It works by redistributing the pixel values in the image so that they cover a more comprehensive range of intensities. Some histogram equalization implementation has proven their ability to improve image quality [26]–[28]. The equation of histogram equalization is shown as follows.

$$a = \frac{(x_i - x_m in)}{(x_m ax - x_m in)} \quad (1)$$

$$b = \frac{(y_i - y_m in)}{(y_m ax - y_m in)} \quad (2)$$

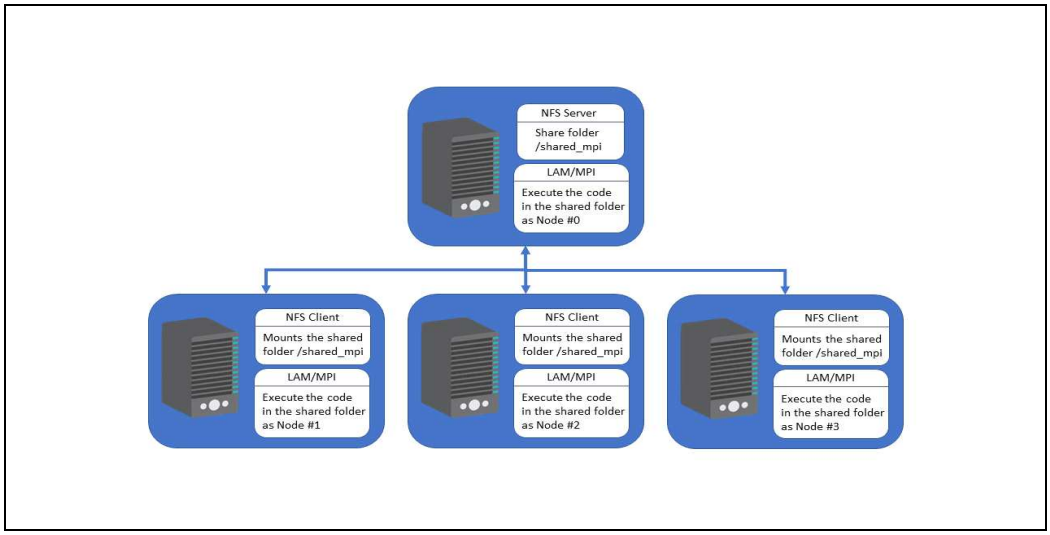

Fig. 2 Local design of MPI architecture for 2D stitching

with x_i is the x-axis index of the selected pixel, and y_i is the y-axis index of the selected pixel. a is the equalized value of x_i, and b is the equalized value of y_i. x_min and y_min is the index of the pixel with minimum value from the entire image.

After the image is enhanced with histogram equalization, the next step is to get the feature points of each image. A prevalent method, SIFT, is used for this task. SIFT is a feature extraction technique used in computer vision for detecting and describing local features in images. David Lowe developed it in 1999 [29]. It has become a popular feature-based image-matching and recognition method [1], [5], [9]. Four stages are mainly involved in SIFT feature detection: scale-space extreme value detection, key point placement, key point direction determination, and feature descriptor construction [5].

1) Scale-space extreme value detection: A method for locating essential characteristics like borders and angles when a picture is blurred to varying degrees. The method finds regions of the picture where the image changes noticeably by blurring it with a Gaussian filter at various blur levels and comparing pixel values.

2) Key point placement: Finding important areas in an image that are stable under different levels of blurring. These areas are selected based on their contrast and presence of local extrema in the Difference-of-Gaussian function. Key points are used to create descriptors that encode the characteristics of the feature at that location. These descriptors can be used for matching and recognition.

3) Key point direction determination: Giving each important area of a picture a direction. A histogram of gradient orientations is constructed to achieve this. Gradient orientation is assigned based on the apex in the histogram, and the gradient magnitudes and orientations around the key point are examined. If there are several summits, various angles might be given. A rotation-invariant description is made using alignment.

4) Feature descriptor construction: A local picture patch's gradient orientations and magnitudes are captured in a 128-dimensional vector by making this vector around a focal point. A histogram of gradient directions is created for each subregion from the picture patch. The feature description, invariant to translation, rotation, and scaling, is made by joining the resulting histograms.

As shown in Figure 3, a key point descriptor is produced by first calculating gradient amplitude and orientation at each image sample point near the key point. These are given weight by a Gaussian window, depicted by circular overlaid. These examples are combined into orientation histograms that summarize the contents over 4x4 subregions. Each arrow's length is determined by adding gradient magnitudes in the area close to that direction.

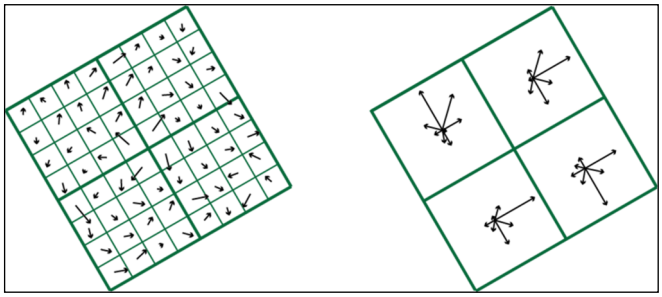

Fig. 3 SIFT descriptor generation [30]

The next step is featuring matching. Using the nearest neighbor approach, a popular feature matching method, each image feature will be matched with the other image. The result of this step is pairs of matched features. From those pairs of matched features, a homograph matrix is acquired. A homograph matrix is a 3x3 transformation matrix that maps points from one plane to another. The homograph matrix can transform one image to match the other after it has been estimated. It is frequently applied when stitching together multiple images to create a panoramic view [1], [2], [6].

D. Stitching Scenarios

Three topologies configuration has been defined to discover the scale-up factor of implementing the MPI in stitching processes. The three topologies are shown in Fig. 4, Fig. 5, and Fig. 6, respectively. The first stitching test is

processed using a single computer without using MPI shown in Figure 4. This test is carried out to discover the required processing time for the stitching process if MPI is not used. All the images are directly stitched with only one routine in one processor.

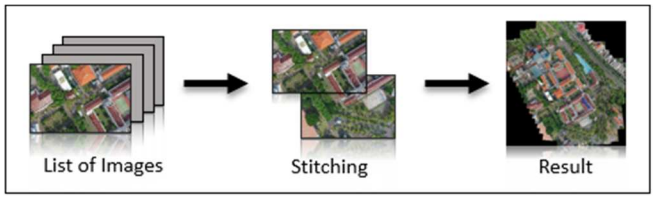

Fig. 4 Stitching Topology on single Processor (without MPI)

Figure 5 shows the topology of the stitching process using MPI with two processors parallelly called nodes. The node master is located on the first computer (first node). The node master is intended to store data that must be worked on during the image stitching. The obtained images will be divided according to the number of existing nodes with several additional overlapping images to keep the features point. After the image stitching process at each node has been completed, the stitched image results will be combined again with the final image stitching process to get the final stitching result.

Another image stitching test scenario is carried out to get the scale-up factor of MPI implementation in the image stitching task. It is shown in Figure 6, where there are four computer nodes. The node master is located at the first node working to divide work on other nodes. After that, the stitched image results from the four nodes will be combined at the node master to get the final stitched image result. The stitching process that was carried out would be tested for their speed performance, and each compared to the results obtained. To implement MPI in the image stitching processes, a python package, mpi4py [31]. The mpi4py package has developed to become the most popular Python binding for the MPI (MPI).

III. Results and Discussion

This section will show and discuss the results of the experiments. To implement parallel processing, we need more than 1 CPU processor. In this research, we created four virtual machine server computers hosted in the cloud from the university. Each used computer has the same specifications that are shown in Table II.

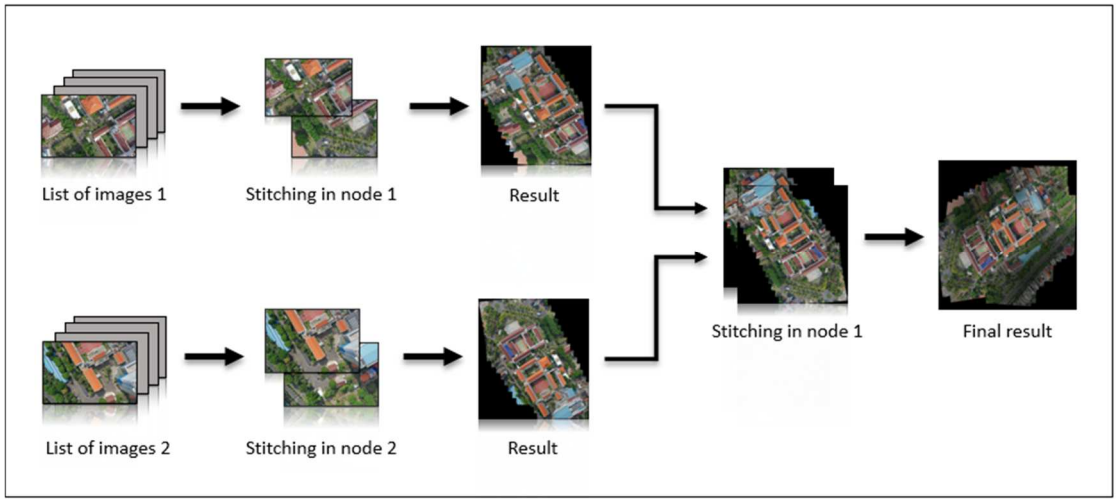

Fig. 5 Stitching topology using 2 processors.

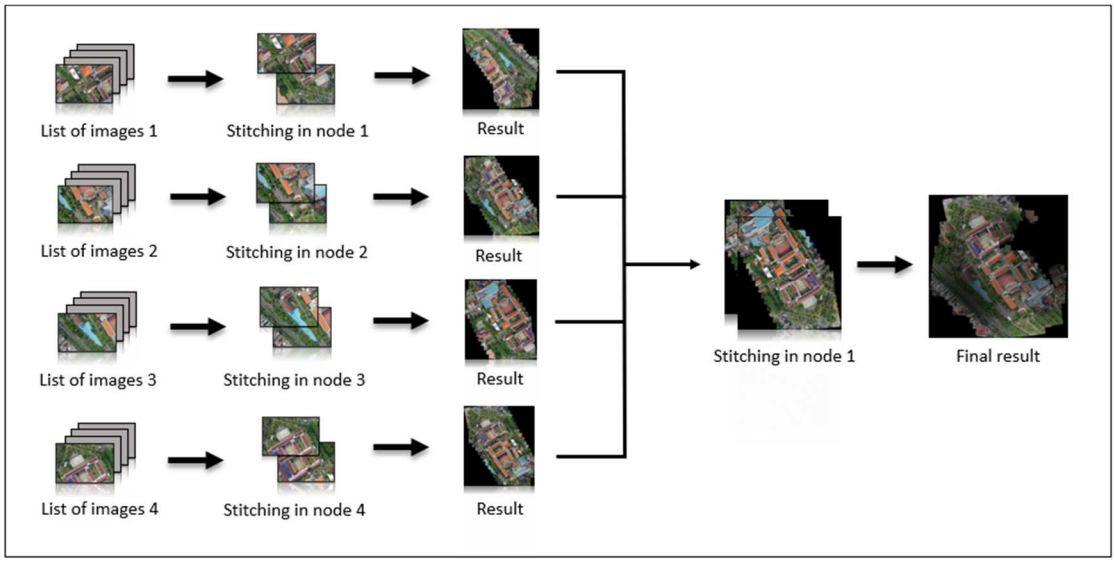

Fig. 6 Stitching topology using 4 processors.

TABLE II
COMPUTER SPECIFICATIONS EACH NODE

No	Items	Specifications
1	Processor	Intel(R) Xeon(R) CPU E5-2650 v2 @2.30 GHz
2	RAM	16 GB
3	Storage	20 GB

We obtained 86 images from the data collection process using DJI Drone, which we explained in section IIA, which took about 25 minutes. The total time taken from the data collection process was about 25 minutes. The data was then processed for the image stitching tests using several scenarios. The designed scenarios for image stitching are:

1. Without MPI, as shown in Figures 4.
2. Using MPI with two computer nodes, as shown in Figures 5.
3. Using MPI with four computer nodes, as shown in Figures 6.

Multiple image stitching scenarios were designed to get the time reduction of the image stitching processes and the scale-up factor for using more than two computer servers. The processing time in this research may also be reduced exponentially in proportion to the number of nodes employed. To verify the hypothesis, we propose several configurations of the number of nodes, which are 1, 2, and 4 nodes. Timestamps were recorded during the image stitching process to get the time of the image stitching process. The test parameter is only focused on observing the time because this research was aimed at reducing processing time while do image stitching using MPI. The result from the image stitching processes can be seen in Figures 7, 8, and 9, respectively.

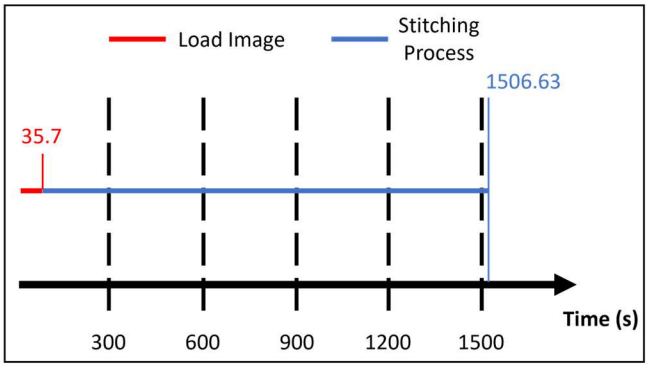

Fig. 7 Proceeded time needed for image stitching without MPI

There were only two phases when not using MPI, shown in Figure 7: the image loading phase and the image stitching phase. The time required to load 86 images is 35.7 seconds, while the time required to carry out the stitching process is 1506.63 seconds or about 25.1 minutes.

When using MPI, there were four phases: image loading phase, partial stitching phase, idle phase, and final stitching phase. The images are split into the number of used computer nodes with overlapping of five images to keep the overlapping features for the stitching process in the final stage. The number of overlapped images is eight images in these tests. The result from the image stitching process with MPI using two computer nodes shows that it needed 20 seconds to load images, with 47 images in each node. The partial image stitching process took 434.4 seconds in the first node and 561 seconds in the others.

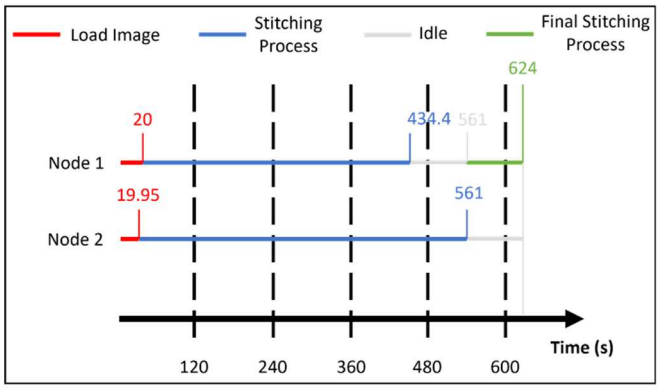

Fig. 8 Proceeded time needed for image stitching using MPI 2 node

When one node finishes the job faster than the others, it will be in the idle phase and wait for the next job. Then, for the final image stitching process, the node master (node 1) collects the stitched image from the other node and processes the final stitching phase. It took 624 seconds. A vast time reduction in the image stitching process has been obtained when image stitching runs with MPI, with a reduction of 881.63 seconds.

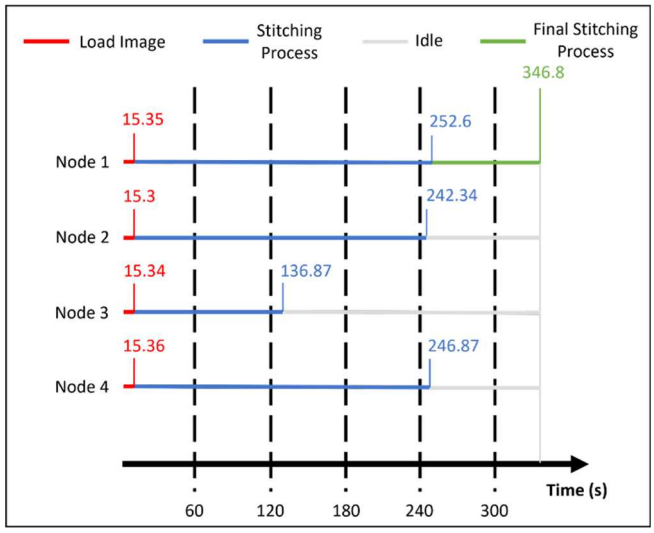

Fig. 9 Proceeded time needed for image stitching using MPI 4 nodes

Considering the time reduction from non-MPI image stitching to 2-node MPI image stitching, we expected to get more reduction in time. Figure 9 shows the test result from the 4-node MPI image stitching scenario. It only took about 15.35 seconds for each node at the same time. The partial stitching phase took variational time, 252.6 seconds for node 1, 242.34 seconds for node 2, 136.87 seconds for node 3, and 246.87 for node 4. The required final stitching process is 94.2 seconds, which takes longer compared to MPI 2 nodes. Those results were caused by increased stitched images from the partial stitching phase that need to be stitched in the final stitching phase to produce the final result.

Figure 10 shows the relation between the number of nodes and the image stitching time. The image stitching process has significant improvement in time from the first scenario to the second scenario. However, in contrast to what we had

expected, we found that the time decrements are not linear after testing the image-stitching processes with three scenarios. These unexpected results could be due to several factors. When the number of nodes increases, overlapping images also increase more than the number of nodes. These overlapping images must have burdened the partial image stitching processes. These findings have important implications for developing the implementation of MPI for image stitching tasks.

Furthermore, the number of MPI nodes also affects the result of the stitched image. Figure 11 shows the stitched image result from each scenario. Of course, there are drawbacks in every stitching process produced both using Non-MPI and MPI. Even though the non-MPI stitching process takes longer, the final results obtained are clearer when compared to stitching using MPI. The more MPI nodes used, the less clear and less accurate as shown in Figure 11.

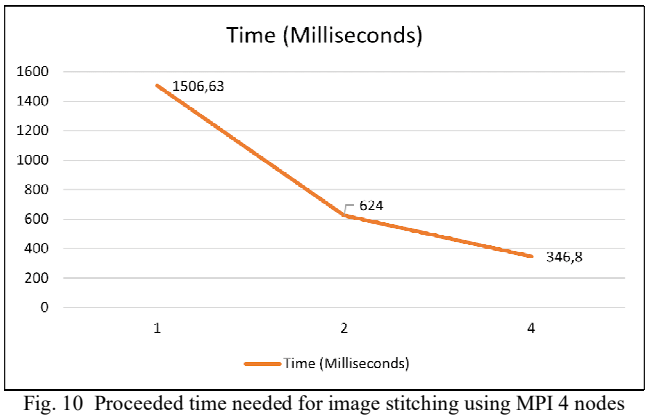

Fig. 10 Proceeded time needed for image stitching using MPI 4 nodes

(a) Non-MPI (b) MPI 2 nodes (c) MPI 4 nodes

Fig. 11 Final result

IV. Conclusion

In this study, we have investigated the effectiveness of MPI used in the stitching process. MPI can boost up the time needed for stitching processes by dividing the task into several nodes. We conducted this analysis by running a stitching process on MPI with two and four nodes and compared the results with non-MPI stitching process. The results show that the total time needed for stitching without MPI was 1506.63 seconds. For MPI with two computer nodes, it took 624 seconds, and it took 346.8 seconds for MPI with four computer nodes. In conclusion, the higher number of nodes that were used, the faster the stitching process can be. However, there is a disadvantage as more MPI nodes are used. Stitching results are becoming less accurate compared to stitching non-MPI. This is due to the lack of detected features due to the reduced capture area because the image is divided into several nodes.

Our results suggest that further research is needed to determine the optimal value of the number of nodes and the number of overlapping images that may have contributed to the unexpected results. In addition, MPI can be applied to 3D stitching, which takes longer time to process than 2D stitching [32].

Acknowledgment

We would like to express our deep appreciation to P3M PENS (Pusat Penelitian dan Pengabdian Masyarakat) for their generous funding, which has played a pivotal role in supporting this research project. Their commitment to advancing scientific knowledge in this research area has enabled us to make significant progress and achieve important milestones in our work. We are profoundly grateful for their invaluable support, which has been instrumental in advancing our research endeavors.

References

[1] E. Adel, M. Elmogy, and H. Elbakry, "Image stitching based on feature extraction techniques: a survey," *International Journal of Computer Applications*, vol. 99, no. 6, pp. 1–8, 2014.

[2] W. LYU, Z. ZHOU, L. CHEN, and Y. ZHOU, "A survey on image and video stitching," Virtual Reality & Intelligent Hardware, vol. 1, no. 1, pp. 55–83, Feb. 2019, doi: 10.3724/sp.j.2096-5796.2018.0008.

[3] B. Ma et al., "A fast algorithm for material image sequential stitching," Computational Materials Science, vol. 158, pp. 1–13, Feb. 2019, doi:10.1016/j.commatsci.2018.10.044.

[4] K. Li and G. Ding, "A Novel Automatic Image Stitching Algorithm for Ceramic Microscopic Images," 2018 International Conference on Audio, Language and Image Processing (ICALIP), Jul. 2018, doi:10.1109/icalip.2018.8455766.

[5] X. Li et al., "Full length image stitching algorithm for spinal deformity surgery," Procedia Computer Science, vol. 209, pp. 93–102, 2022, doi:10.1016/j.procs.2022.10.103.

[6] R. Xie et al., "Automatic multi-image stitching for concrete bridge inspection by combining point and line features," Automation in Construction, vol. 90, pp. 265–280, Jun. 2018, doi:10.1016/j.autcon.2018.02.021.

[7] G. Rosa et al., "Hyperspectral Images Acquisition: an Efficient Capture and Processing Stitching Procedure for Medical Environments," 2020 XXXV Conference on Design of Circuits and

Integrated Systems (DCIS), Nov. 2020, doi:10.1109/dcis51330.2020.9268658.
[8] S. Wang, C. Liu, and Y. Zhang, "Fully convolution network architecture for steel-beam crack detection in fast-stitching images," Mechanical Systems and Signal Processing, vol. 165, p. 108377, Feb. 2022, doi: 10.1016/j.ymssp.2021.108377.
[9] Y. Liu, M. He, Y. Wang, Y. Sun, and X. Gao, "Farmland Aerial Images Fast-Stitching Method and Application Based on Improved SIFT Algorithm," IEEE Access, vol. 10, pp. 95411–95424, 2022, doi:10.1109/access.2022.3204657.
[10] T. Hovhannisyan, P. Efendyan, and M. Vardanyan, "Creation of a digital model of fields with application of DJI phantom 3 drone and the opportunities of its utilization in agriculture," Annals of Agrarian Science, vol. 16, no. 2, pp. 177–180, Jun. 2018, doi:10.1016/j.aasci.2018.03.006.
[11] T. Zhang and M. Zhu, "GPS-assisted Aerial Image Stitching Based on optimization Algorithm," 2019 Chinese Control Conference (CCC), Jul. 2019, doi: 10.23919/chicc.2019.8866089.
[12] O. Chum and J. Matas, "Optimal Randomized RANSAC," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 30, no. 8, pp. 1472–1482, Aug. 2008, doi: 10.1109/tpami.2007.70787.
[13] G. Yang, X. Chang, and Z. Jiang, "A Fast Aerial Images Mosaic Method Based on ORB Feature and Homography Matrix," 2019 International Conference on Computer, Information and Telecommunication Systems (CITS), Aug. 2019, doi:10.1109/cits.2019.8862133.
[14] H. Zhao, Y. Du, H. Wang, and Y. Yue, "UAV aerial image mosaic algorithm based on FAST-Tomasi feature and Delaunay triangulation," 2020 IEEE International Conference on Mechatronics and Automation (ICMA), Oct. 2020, doi: 10.1109/icma49215.2020.9233570.
[15] N. T. Pham, S. Park, and C.-S. Park, "Fast and Efficient Method for Large-Scale Aerial Image Stitching," IEEE Access, vol. 9, pp. 127852–127865, 2021, doi: 10.1109/access.2021.3111203.
[16] J. L. Hennessy and D. A. Patterson, "Fundamentals of quantitative design and analysis," *Computer Architecture: A Quantitative Approach*, pp. 1–10, 2012.
[17] W. D. Hillis, "What is massively parallel computing, and why is it important?," *Daedalus*, vol. 121, no. 1, Art. no. 1, 1992.
[18] P. Czarnul, J. Proficz, and K. Drypczewski, "Survey of methodologies, approaches, and challenges in parallel programming using high-performance computing systems," *Scientific Programming*, vol. 2020, pp. 1–19, 2020.
[19] J. J. Dongarra, S. W. Otto, M. Snir, and D. Walker, "An introduction to the MPI standard," *Communications of the ACM*, vol. 18, 1995.
[20] S. A. Dheyab, M. N. Abdullah, and B. F. Abed, "A novel approach for big data processing using message passing interface based on memory mapping," *Journal of Big Data*, vol. 6, no. 1, Art. no. 1, 2019.
[21] Z. Jiang *et al.*, "Message passing optimization in robot operating system," *International Journal of Parallel Programming*, vol. 48, pp. 119–136, 2020.
[22] T. Ragunthar, P. Ashok, N. Gopinath, and M. Subashini, "A strong reinforcement parallel implementation of k-means algorithm using message passing interface," *Materials Today: Proceedings*, vol. 46, pp. 3799–3802, 2021.
[23] E. D. Fajrianti, A. A. Pratama, J. A. Nasyir, A. Rasyid, I. Winarno, and S. Sukaridhoto, "High-Performance Computing on Agriculture: Analysis of Corn Leaf Disease," *JOIV: International Journal on Informatics Visualization*, vol. 6, no. 2, Art. no. 2, 2022.
[24] C. A. Swann, "Software for parallel computing: the LAM implementation of MPI," Journal of Applied Econometrics, vol. 16, no. 2, pp. 185–194, Mar. 2001, doi: 10.1002/jae.595.
[25] J. M. Squyres and A. Lumsdaine, "A Component Architecture for LAM/MPI," Lecture Notes in Computer Science, pp. 379–387, 2003, doi: 10.1007/978-3-540-39924-7_52.
[26] S. M. Pizer *et al.*, "Adaptive histogram equalization and its variations," *Computer vision, graphics, and image processing*, vol. 39, no. 3, Art. no. 3, 1987.
[27] T. Arici, S. Dikbas, and Y. Altunbasak, "A histogram modification framework and its application for image contrast enhancement," *IEEE Transactions on image processing*, vol. 18, no. 9, Art. no. 9, 2009.
[28] B. S. Rao, "Dynamic histogram equalization for contrast enhancement for digital images," *Applied Soft Computing*, vol. 89, p. 106114, 2020.
[29] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004.
[30] T. Lindeberg, "Scale Invariant Feature Transform," Scholarpedia, vol. 7, no. 5, p. 10491, 2012, doi: 10.4249/scholarpedia.10491.
[31] L. Dalcin and Y.-L. L. Fang, "mpi4py: Status update after 12 years of development," *Computing in Science & Engineering*, vol. 23, no. 4, Art. no. 4, 2021.
[32] J. Satriawan, "3D Object Mapping using Drone Based on Autonomous Waypoint Navigation," *Unpublished Paper*, 2023.