Towards a fast image alignment technique for multi-aperture UAV and CubeSat payloads

Muhammad Ahmed Nana[1]*, *Rofhiwa* Seletani[1], and *Shrikant Virendra* Naidoo[1]

[1] Optronic Sensor Systems, Council for Scientific and Industrial Research (CSIR), South Africa

Abstract. This paper compares various image alignment techniques implemented on low-powered embedded devices for use in multi-aperture UAV and CubeSat payloads. A novel method was introduced and executed alongside two established methods on an embedded processor: utilising images taken from a dual-aperture UAV payload for the performance evaluation. The techniques were evaluated based on processing time, wherein significant differences were observed, outside of the universally comparable structural similarity index (SSIM). The outcomes of this research indicate that among the methods compared, the proposed method requires the least amount of processing time without sacrificing alignment quality, making it suitable for use in, applicable, low-power applications.

1 Introduction

Image alignment, also known as image registration, involves modifying the position of one or more images to ensure that the relevant features or objects within them match. This process is crucial in applications such as automated inspection systems [1, 2], medical imaging [3], satellite and aerial imaging [4], panorama stitching[5], robotics, and autonomous vehicles [6], etc., where precise alignment can improve object detection, tracking, and pattern recognition tasks, thus enhancing the efficiency and effectiveness of the systems. It involves aligning several images taken from different perspectives and identifying the same features from multiple cameras or various angles/views of the same camera.

Image alignment is typically used as a pre-processing step in multi-aperture electro-optical payloads to localise the same features captured from different camera views. This enables subsequent image processing stages to easily analyse common features. Various image alignment techniques have been developed in the last few decades, including Feature-Based Alignment [7], Pixel-Based Alignment [8], Fourier-Based Alignment [9], and Deep Learning Techniques [10], etc.

The selection of techniques for use in a particular application depends on various factors, including the accuracy, execution speed required, and the computational resources available. In the context of Unmanned Aerial Vehicle (UAV) and CubeSat payloads, computation capabilities are limited due to the use of low-powered embedded processors. Another constraint is the need for near real-time performance. Thus, it is important to implement efficient algorithms to meet the stringent time and power constraints [11].

* Corresponding author: mnana@csir.co.za

This paper compares various image alignment techniques suitable for implementation on low-powered embedded devices for use in multi-aperture UAV and CubeSat payloads. A new method is proposed that takes advantage of the perspective obtained from high-altitude aircraft by implementing stereo rectification and a horizontal shift. The new method offers two benefits over the other methods: (1) a faster execution time, and (2) the ability to achieve accurate alignment when no features can be detected from images.

2 Related work

The conventional image alignment technique generally comprises three primary stages: (1) the detection of similar features between two or more images, (2) the estimation of the geometric projection relationship between images (homography matrix), and (3) the application of a perspective warp to obtain the aligned images. There is extensive literature on different techniques that researchers have employed at each stage of the image alignment process.

In the first stage, feature extraction algorithms are used to detect and match similar features. L Yinhui et al. [12] reviewed different feature extraction techniques including Scale Invariant Feature Transform (SIFT), Speed Up Robust Features (SURF), and Orientated fast and Rotated Brient (ORB), etc. L Yinhui et al. [12] also provides a comprehensive review of various techniques used during second stage to estimate the homography matrix for image alignment applications, including Feature-based, pixel-based, and deep learning based, etc. RANSAC and other robust estimation algorithms are commonly used to extract a consistent subset from matched features enabling accurate homography matrix estimation [12].

The performance of traditional techniques can degrade in high noise or non-textured images (no features) [12]. Other researchers have explored machine learning and deep learning techniques to improve the accuracy of traditional image alignment techniques. Learning-based methods use neural networks to replace or improve the feature extraction stage [10]. Machine and deep learning techniques generally require high computational resources, making them less ideal for low-power embedded devices [11]. This paper considers targeting devices with minimal core counts and processing capabilities unsuitable for implementing computationally challenging deep learning algorithms. These family of algorithms have therefore been excluded from this study.

In applications involving two or more cameras that capture the same scene from different points of view, prior geometric mapping knowledge between the physical cameras can be used to align images. Stereo calibration involves computing intrinsic and extrinsic parameters for a stereo vision system with two or more cameras. The intrinsic parameters involve properties such as lens distortion coefficients, optical centres, and focal lengths, and the extrinsic parameters involve the orientations and positions of the cameras relative to each other. Calibration determines the accuracy of the stereo matching process, providing a reference frame from which depth can be perceived [13].

Stereo Rectification of a stereo image set transforms images so that corresponding epipolar lines align to be parallel to each other. This facilitates a simplified search for corresponding pixels among the two images and serves to transform a 2D correspondence problem into a 1D correspondence problem. This transform allows for the corresponding matching to occur on a horizontal line [14].

The Research in [15] proposed a novel solution to rectify stereo images captured through a parallel camera configuration. This addressed the geometric inaccuracies encountered due to manual misalignment of cameras and differences in their intrinsic parameters. The method proposes that the horizontal baseline, the distance between the apertures, is proportional to the disparity, i.e.: horizontal misalignment between the corresponding pixels in both images. In this technique, an optimised transformation is used to adjust the perspective of parallel optical axis images, ensuring the horizontal alignment of corresponding epipolar lines. Before horizontal alignment, geometric parameters derived from camera calibration are used to manipulate the vertical coordinates of each pixel to achieve rectification. However, this method is limited in that the images do not horizontally align and still require a disparity computation and horizontal offset. Yang et al. [16] details a computation of a pixelwise matching cost calculation, defined by disparity, which can become a resource intensive calculation when the variance of the image plane is a significant factor in the observed scene.

Yang et al. [16] proposes a method for novel image matching in stereo vision with the computation of a disparity map applied to the development of digital surface modelling. The research also indicates that the assumption of constant disparity is reasonable if the objects in the field of view occur in the same image plane. This typically holds true for bodies of water and flat grasslands. The calculation of the disparity map proposed in [16] is highly simplified to a uniform horizontal shift, thus reducing the amount of processing power required to compute the disparity per pixel match in the stereo pair.

Fig. 1 illustrates the convergence of the disparity to a constant at longer distances. In the case of remote sensing on reasonably flat surfaces such as water and farmlands, etc., aligned images can be approximated by matched images with a uniform shift applied by virtue of a constant disparity. The computational cost of such a use case is estimated to be less than other image alignment methods such as those in our results (see Table 1).

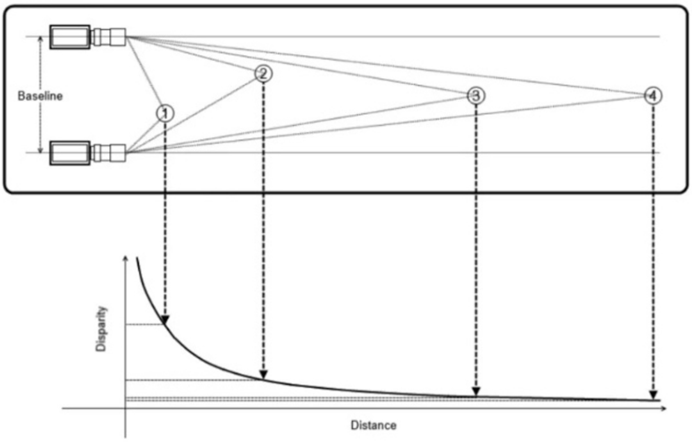


Fig. 1. Convergence of disparities at longer distances. Source [17]

3 Proposed method

The alignment technique method proposed in this research exploits that the disparity between horizontally corresponding points on two parallel optical axis image planes can be considered constant at longer distances and reasonably flat scenes. After stereo rectification, the disparity at longer distances is expected to affect a constant horizontal offset to the images. Image alignment in both axes can be achieved relatively inexpensively with traditional stereo camera calibration to serve the vertical axis, and the constant disparity offset can be applied to each pixel to align the horizontal axis.

The alignment technique block diagram is represented in Fig. 2. The stereo calibration is a pre-processing step and is performed only once at the outset, after which its computed parameters are stored for later use. The input image pair undergoes stereo rectification. One of the stereo-recorded images is then horizontally translated by the disparity, which, when coupled with the other rectified image, produces the required aligned images.

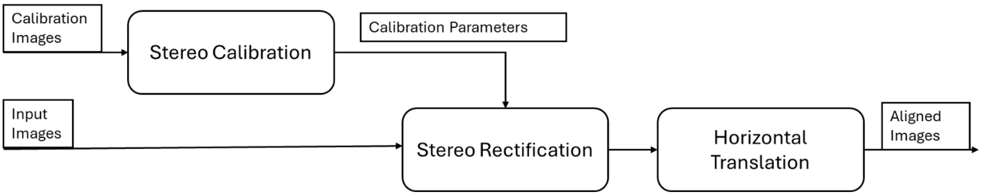
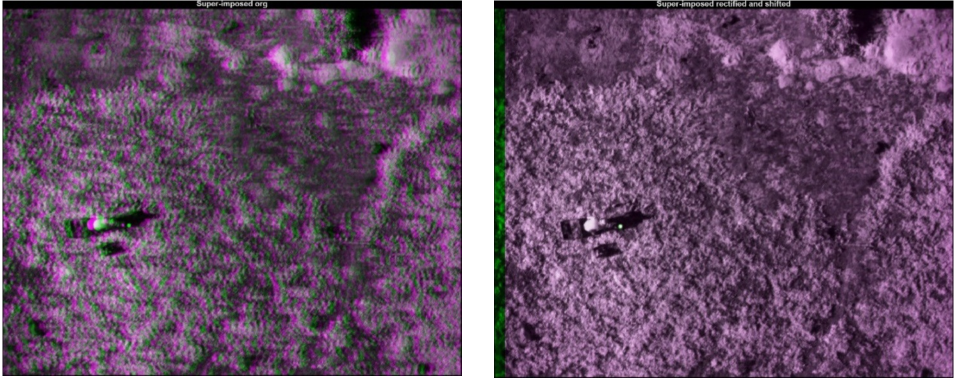


Fig. 2. Block diagram depicting the proposed alignment algorithm using stereo rectification and horizontal translation/shift.

The prototyping of the method was conducted in MATLAB. The stereo camera calibration toolbox was used to calibrate the camera system. MATLAB functions were then used to stereo rectify and horizontally shift one of the images. The input and output of the process are shown in Fig. 3. The images were generated using MATLAB's "imshowpair" function using the "falsecolor" visualisation method. This superimposed the images onto each other, enabling visual inspection of the alignment. The separation of green and magenta features shows the level of alignment; a greater separation shows a lower quality alignment.



(a) (b)

Fig. 3. Superimposed images: (a) original images before alignment; (b) aligned images using stereo rectification and horizontal shift.

4 Methodology

4.1 Comparative metrics

The algorithms were compared using two primary metrics: (1) their processing times, and (2) the Structural Similarity Index (SSIM) produced. The processing time of the algorithm was used as a measure of the use of computational resources. This assumption was made since all the algorithms were tested on the same hardware and executed in the same test program. The SSIM was used as a measure of the quality of the image alignment produced by the algorithms. The SSIM is an image quality assessment metric that considers luminance, contrast, and structural information when comparing images [18]. Structural information refers to the spatial relationships between pixels, considering features such as edges and textures. This allows the computed SSIM value to be used as a measure of how well the images aligned.

The two primary metrics, i.e., CPU and memory usage, were recorded at runtime using "cppuprofile" [19], an open-source C++ library that allows the collection of such metrics on embedded devices. It is a lightweight library with negligible expected interference with the execution of the algorithms. Any added latency or impediment is assumed to be constant in all implementation comparisons.

4.2 Data capture

A custom dual-aperture camera system, developed by the Council for Scientific and Industrial Research (CSIR), was used to capture the images used to test the algorithms. The camera system uses two 1.31MPix iDS UI-1242LE-NIR cameras coupled with Qioptiq 50mm MeVis-CF 1.8/50 lenses. The cameras were configured to capture images synchronously at 5 frames per second (FPS) and were stored as uncompressed video. The video was later separated into frames used for the testing. An example of the image pairs captured is shown in Fig. 4.

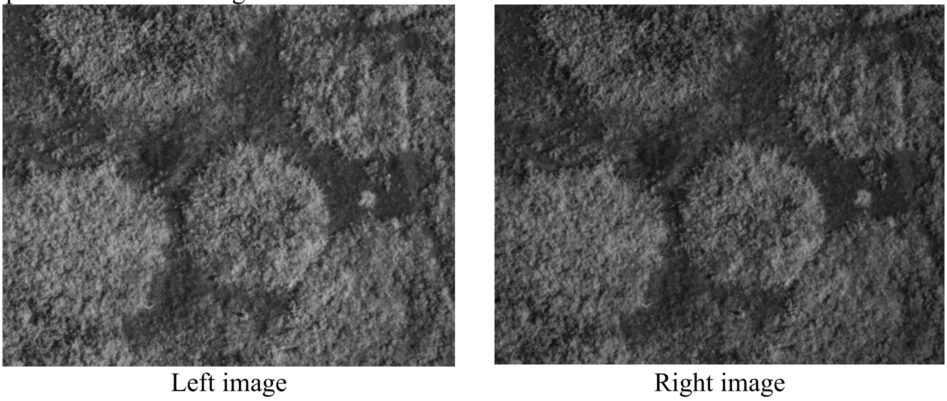


Fig. 4. Sample of the image pair taken from the camera system.

The camera system was mounted on a UAV using a gimbal. The gimbal was setup to stabilise in pitch and roll; and was locked with respect to yaw. The cameras were pointed vertically downward to view the ground at nadir, with the stabilisation of the gimbal used to maintain that orientation irrespective of the orientation of the UAV.

The UAV was flown in a grid-search pattern, with a 50/50 split between flat open ground and a forested area. The focus of these tests was on the open ground section, but some images containing trees were also used. The grid-search pattern was flown at different altitudes: 120m, 80m and 60m, with the 120m being the maximum legal altitude of the UAV’s mission profile. The grid-search pattern is shown in Fig. 5 with all the waypoints used during the flight.

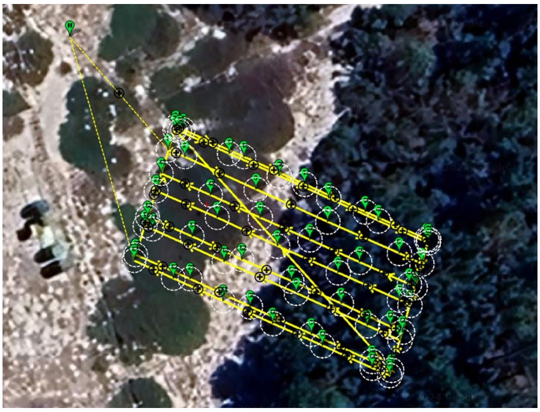


Fig. 5. The grid-search pattern flown by the UAV superimposed on a satellite image of the area.

4.3 Processing hardware

An Nvidia Jetson Xavier NX module with the Seeed Studio reComputer J202 carrier board was used to test the algorithms. The module was set to its default power mode (Mode 5) which is a 10W mode that enables four CPU cores at 1.9GHz with a fully functional desktop.

4.4 Software program

A single executable program was developed to test the algorithms. Each algorithm was applied sequentially to the same set of images. This was done to minimise the effect of Operating System (OS) interruptions such as software updates. In this way, any interruption is expected to affect the execution of each algorithm equally. The cppuprofile library was used to calculate the processing time of each algorithm and log CPU and memory usage. The logging was performed at 50ms intervals to prevent the logging from significantly affecting the execution of the algorithms. Data acquisition at intervals less than 50ms noticeably limited the algorithm execution and performance. Reducing the acquisition to 50ms yielded negligible interference on the algorithms, with the drawback that no data would be available for algorithms that take a shorter time to execute.

The algorithms were implemented using version 4 of the OpenCV library. OpenCV functions were implemented to parallelise the processing of the algorithms. The implementations were restricted to using the CPU cores only, intentionally excluding GPU usage to accommodate that many low-power devices may not possess GPU core instances in their fabric.

A block diagram to describe the software program developed to test the algorithms is illustrated in Fig. 6. The dashed lines indicate the sequence of operations and do not necessarily refer to data input.

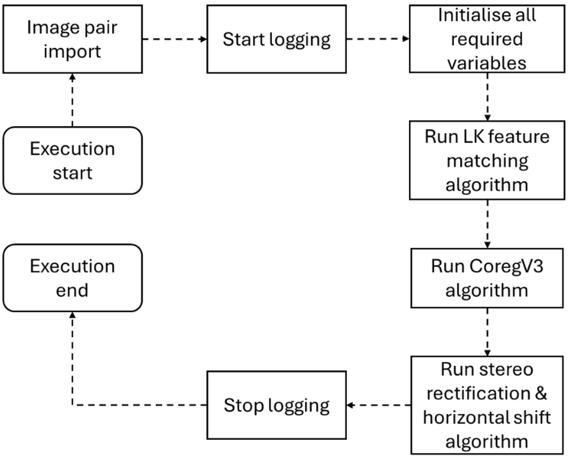


Image pair import
Start logging
Initialise all required variables
Execution start
Run LK feature matching algorithm
Execution end
Run CoregV3 algorithm
Stop logging
Run stereo rectification & horizontal shift algorithm

Fig. 6. Block diagram showing the processing sequence of the software executable.

5 Results and discussion

The software program was developed to take a pair of images as input. A total of 45 image pairs were used as sample points to generate the results. The images spanned three different altitudes, with 15 image pairs per altitude used. Eight image pairs contained trees with heights ranging from 10 to 20 meters on average. These were included to observe the robustness of the algorithms to sudden changes in target distance.

5.1 Processing time

Processing time was measured using the cppuprofile library. The library has functions that can be called to start and stop a measurement. These functions were used to measure the processing time required by each algorithm, omitting variable declarations and memory allocations. The stereo rectification algorithm has two parts: (1) calculating a mapping, and (2) applying the mapping. The calculation of the mapping only needs to be executed once per camera calibration, so it was excluded from the processing time measurement. Table 1 shows the minimum, maximum, mean, median, and standard deviation of the processing time for each algorithm in milliseconds.

Table 1. Processing time of each algorithm in milliseconds.

Property	LK feature matching	CoregV3	Stereo rectification + shift
Minimum	118.00	4658.00	12.00
Maximum	258.00	5818.00	32.00
Mean	136.95	4838.75	15.24
Median	127	4807	14
Standard deviation	29.79	175.53	3.65

The results show that there are orders of magnitude differences in the processing time for each algorithm. The proposed algorithm, stereo rectification and a horizontal shift, was the fastest with a mean of 15.24ms. This exhibits desirable performance, as it implies that this algorithm requires the least amount of processing resources, further implying that it would require the least amount of electrical power. This makes it the most suitable approach for low-power applications.

5.2 Structural similarity index metric (SSIM)

The structural similarity index (SSIM) was used to measure the quality of image alignment. Unity value indicates that the images are the same. Algorithms that achieve the closest value to one are considered to have the best performance. The SSIM was calculated using the function provided by OpenCV. The calculation of SSIMs was excluded from the processing time measurements described in Section 5.1. The minimum, maximum, mean, median, and standard deviation of the SSIM for each algorithm is shown in Table 2.

Table 2. Structural similarity index (SSIM) of each algorithm.

Property	Original images	LK feature matching	CoregV3	Stereo rectification + shift
Minimum	0.1473	0.8872	0.8788	0.7484
Maximum	0.4202	0.9213	0.9089	0.9147
Mean	0.2370	0.9084	0.8993	0.8900
Median	0.207	0.9121	0.9011	0.9014
Standard deviation	0.0791	0.0096	0.0062	0.0337

The SSIMs of the original images, before alignment, are given to show the improvement that the alignment algorithms produce. Overall, the algorithms performed comparably with SSIM values that are very close to each other. The only highlights are that the stereo algorithm's minimum is much lower than the respective value in the other algorithms and that its standard deviation is also much larger. To further explore this, the box and whisker plots of the SSIMs were drawn as shown in Fig. 7.

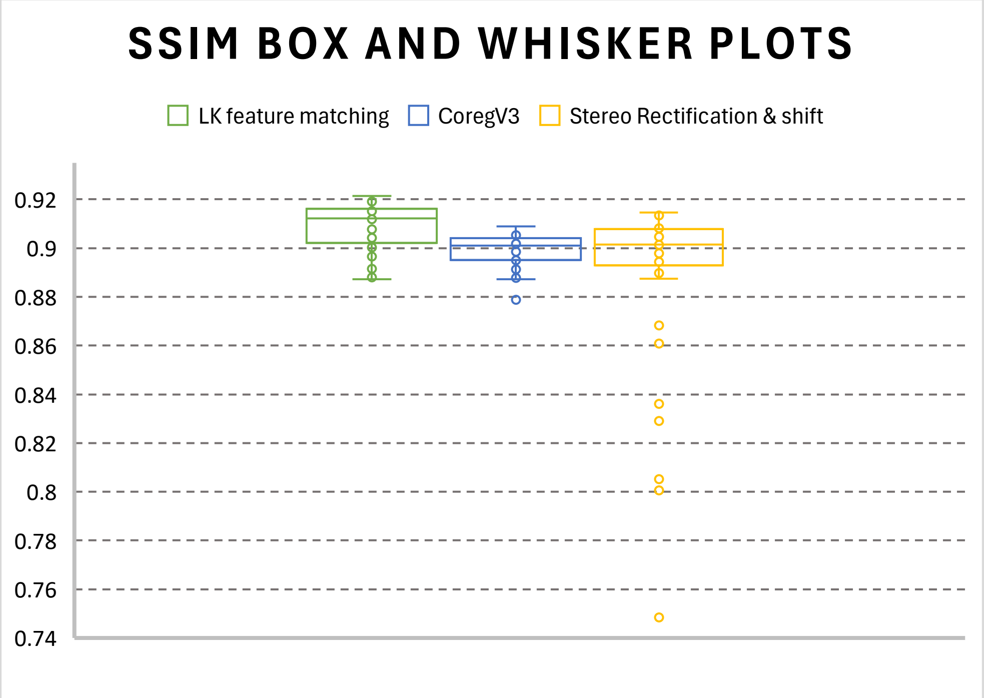


Fig. 7. Box and whisker plot of the SSIMs calculated for each algorithm.

Fig.7 reiterates the initial observation that the results are comparable with negligible difference between the algorithms. It also shows that there were numerous outliers for the stereo algorithm, which contributed to the larger standard deviation and a lower-than-expected minimum. Further analysis of the image pairs that produced the outliers is required.

5.3 CPU and memory usage

The CPU and memory usage was measured using the cppuprofile library. It was set to log at 50ms intervals to minimise the effect of this datalogging on the execution of the algorithms. A sample of the library output is shown in Fig. 8. The topmost graph in Fig.8 is a visual indication of the processing time used by each algorithm, whereas the centremost graph shows the CPU usage over time during the execution of the program. It shows that all four CPU cores were used during the execution of the LK and CoregV3 algorithms. The execution time of the stereo algorithm was too short for a logging sample to be taken during its execution. Nevertheless, it can be assumed that all the CPU cores were used during its execution, since it was developed using OpenCV functions that are parallelised. The lower graph in Fig. 8 shows the process memory usage for the software executable.

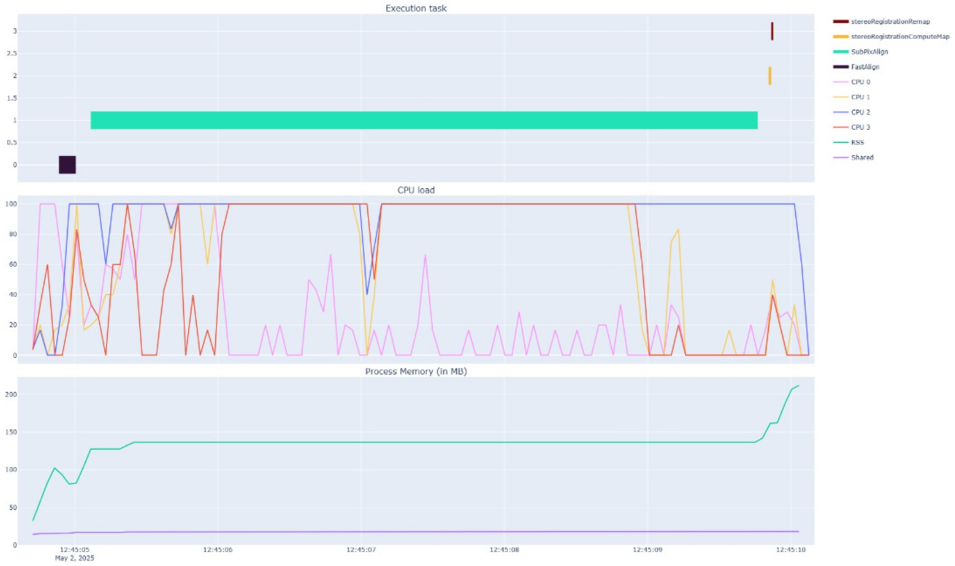


Fig. 8. Graph showing execution time, CPU usage, and process memory usage of the software executable generated with cppuprofile.

6 Conclusions and recommendations

Two previously used image alignment algorithms and a newly proposed one were compared in terms of processing time (milliseconds) and alignment quality (using SSIM). The processing times of the three algorithms were found to be different by orders of magnitude. The proposed algorithm, based on stereo rectification, was the fastest, with a mean processing time of 15.24ms. This was followed by the Lukas-Kanade based algorithm at 136.95ms, and finally the CoregV3 algorithm at 4838.75ms.

The alignment quality of the three algorithms was found to be comparable with very small differences between them. Stereo rectification, LK, and CoregV3 based algorithms produced mean SSIM values of 0.89, 0.9084, and 0.8993, respectively. When excluding the few outliers present in the results of the stereo algorithm, its mean SSIM increased to 0.9008. The SSIM results can be visually seen in Fig. 7, which suggests only small differences between the algorithms.

The exceptional processing time performance and comparable alignment quality of the proposed stereo rectification-based image alignment algorithm make it a good option for use on applicable, on-board, low-power applications. Another positive observation was that the proposed algorithm performed better when the image pairs did not have many good features that could be used in the other algorithms, i.e. it was invariant to available features in the images.

All the outliers present in the stereo-based algorithm were captured from a sequential 14s period. More detailed analysis of these images is needed to determine why they produced lower-than-expected alignment quality. The images containing trees yielded the same results as the other images, and none of them were part of the outliers. These two observations indicate that more research on the proposed algorithm is needed to fully articulate its limitations and provide a detailed indication of the conditions under which its use is valid.

References

1. S. Suh and H. Cho, Fast Image Alignment for Inspection of FCB Bump Coin System, in Proceedings of the 3rd ICIAE-2015 conference, International Conference on Industrial Application Engineering, Kitakyushu, Japan, March 28-31 (2015) 581-585. https://doi.org/10.12792/iciae2015.100
2. W. H. Li, Study on the Technology of Digital Image Processing, AMM, **687**, 3555-3558, (2014). https://doi.org/10.4028/www.scientific.net/AMM.687-691.3555
3. C.W. Wang, H.S. Chen, Improved image alignment method in application to X-ray images and biological images, Bioinformatics, **29**, 1879–1887, (2013). https://doi.org/10.1093/bioinformatics/btt309
4. M. Scaioni, L. Barazetti, M. Gianinetto, Multi-Image Robust Alignment of Medium-Resolution Satellite Imagery, Remote Sensing, **10**, 1969, (2018). https://doi.org/10.3390/rs10121969
5. K. Nidhal, E.L. Abbaid, S.A.A. Hassani, A.H. Abdulkhaleq, A Review Over Panoramic Image Stitching Techniques, in Preceedings of 2nd International Virtual Conference on Pure Science, Journal of Physics: Conference Series, online, April 21-22 (2021), 012115. https://iopscience.iop.org/article/10.1088/1742-6596/1999/1/012115
6. Y. Xia, J. Monica, W. -L. Chao, B. Hariharan, K. Q. Weinberger, M. Campbell, Image-to-Image Translation for Autonomous Driving from Coarsely-Aligned Image Pairs, in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), London, UK, May 29 – June 2 (2023), 7756-7762. https://doi.org/10.48550/arXiv.2209.11673
7. N. I. Thiruselvam, and S. J. Subramanian, Feature-assisted stereo correlation, strain, **55**, e12315, (2019). https://doi.org/10.1111/str.12315
8. A. Tavera, C. Fabio, C. M. Masone, B. Caputo, Pixel-by-Pixel Cross-Domain Alignment for Few-Shot Semantic Segmentation, in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, January 3-8 (2022), 1626-1635. https://doi.org/10.48550/arXiv.2110.11650
9. C. Chin-Sheng, Y. Chun-Wei, Y. Peng-Yeng, A novel Fourier descriptor based image alignment algorithm for automatic optical inspection, Journal of Visual Communication and Image Representation, **20**, 178-189, (2009). https://doi.org/10.1016/j.jvcir.2008.11.003
10. H. and Yoon, K. Cho, Event-image fusion stereo using cross-modality feature propagation, in Proceedings of the AAAI Conference on Artificial Intelligence, Association for the Advancement of Artificial Intelligence, online, February 22 – March 1 (2022), 454-462. https://doi.org/10.1609/aaai.v36i1.19923
11. Y. Zheng, Implementation and Optimization of Embedded Image Processing System, in Proceedings of the 2015 International Conference on Electrical, Computer Engineering and Electronics, Jinan, China, May 29-31 (2015), 1386-1391. https://doi.org/10.2991/icecee-15.2015.259
12. Y. Luo, X. Wang, Y. Lioa, Q. Fu, C. Shu, Y. Wu, Y.A. He, A Review of Homography Estimation: Advances and Challenges, Electronics, **12**, 4977, (2023). https://doi.org/10.3390/electronics12244977
13. C. Cai, X. Weng, B. Fan, and Q. Zhu, Calibration and rectification of vertically aligned binocular omnistereo vision systems, ERASIP Journal on Image and Video Processing, **2017**, 1-14, (2017). https://doi.org/10.1186/s13640-017-0195-0

14. L. Guo-Yu, C. Xu, Z. Wei-Gong, A Robust Epipolar Rectification Method of Stereo Pairs, in Proceedings of the International Conference on Measuring Technology and Mechatronics Automation, Changsha, China, March 13-14 (2010), 322-326. https://doi.org/10.1109/ICMTMA.2010.220
15. Y.S. Kang, Y.S. Ho, Efficient Stereo Image rectification method using horizontal Baseline, in Proceedings of the Pacific-Rim Symposium on Image and Video Technology, Gwangju, Korea, November 20-23 (2011), 301–310. https://doi.org/10.1007/978-3-642-25367-6_27
16. W. Yang, X. Li, B. Yang, and Y. Fu, A Novel Stereo Matching Algorithm for Digital Surface Model (DSM) Generation in Water Areas, Remote Sensing, **12**, 870 – 891, (2020). https://doi.org/10.3390/rs12050870
17. A. Kaehler, G. Bradski, Learning OpenCV 3, (O'Reilly Media Inc., ISBN: 9781491937990, 2016)
18. I. Bakurov, M. Buzzelli, R. Schettini, M. Castelli, L. Vanneschi, Structural similarity index (SSIM) revisited: A data-driven approach, Expert Systems with Applications, **189**, 116087, (2022). https://doi.org/10.1016/j.eswa.2021.116087
19. C. Chedaleux, cppuprofile, https://github.com/Orange-OpenSource/cppuprofile, (2025)