REAL-TIME PHOTOGRAMMETRY BASED ON PARALLEL ARCHITECTURE FOR 3D APPLICATIONS

CHEN LONG

PhD

The Hong Kong Polytechnic University

2024

The Hong Kong Polytechnic University

Department of Land Surveying and Geo-Informatics

Real-time Photogrammetry Based on Parallel Architecture for 3D Applications

CHEN Long

A Thesis Submitted in Partial Fulfilment of The Requirements for the Degree of Doctor of Philosophy

July 2023

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

CHEN Long (Name of student)

Abstract

Photogrammetry is the technique that allows capturing and reconstructing 3D models of objects and scenes from multiple images. In recent years, with the rising demand for efficient 3D reconstruction, real-time photogrammetry has attracted much attention in various domains, such as unmanned aerial vehicles (UAVs) navigation, disaster emergency response, human tracking, and autonomous driven. This research focuses on the enhancement of the computational efficiency of photogrammetric algorithms by taking advantage of parallel architectures and combining them with cutting-edge methods such as deep learning to achieve real-time photogrammetry in various scenarios.

The traditional visual navigation algorithms in a GPS-denied environment enable the acquisition of approximate relative poses of cameras. However, tradition methods, such as visual odometry (VO) suffers from attitude estimation errors that accumulate over time and cause the estimated trajectory to drift, and the data processing efficiency is relatively low. To address these challenges, this research firstly presents a feature-based cross-view image matching and retrieval method for real-time camera pose estimation by incorporating VO and photogrammetry algorithms. Specifically, the method uses a deep-learning feature extraction and matching method to improve the robustness of the relative pose estimation of the camera by VO. To correct accumulated errors by VO, the method selects keyframes and applies photogrammetric algorithm of space resection to determine the accurate pose information of the keyframes. The accurate camera pose information of keyframes are then used to rectify the possible drift caused by VO. Parallel architectures are implemented to enhance the data processing efficiency. Experimental analysis using real UAV datasets shows that the developed method achieves a root mean square error (RMSE) of 4.7 m for absolute positional error and 0.33° for rotation error, as compared with ground truth data. The developed method also achieves an efficiency of 12 frames per second (FPS) based on the parallel architecture implemented in a regular computer, indicating its real-time performance.

Dense image matching in real time is a challenging task because of the high computation demand and high degree of ambiguity that often occurs in practical

situations. The state-of-the-art methods such as the semi-global matching (SGM) with diverse local similarity metrics, offering favourable dense matching results against various types of noise and disturbances, such as illumination variations and the ability to handle textureless regions and preserve edges. However, the computational burden associated with SGM hinders its real-time processing capabilities. To overcome these challenges, this research leverages parallel structured systems, specifically graphic processing units (GPUs), to enabled real-time dense image matching. A comprehensive disparity estimation pipeline based on a GPU-accelerated device is developed and evaluated. An effective parallel scheme and data layout strategy is proposed for the core functions in the disparity estimation algorithm, and the algorithm codes are further optimised to enhance efficiency. The optimised algorithm is deployed on a high-end GPU, utilising the sum of absolute distance (SAD) as the similarity measurement, 64 disparity levels, and 8 path directions for the SGM method. As a result, the system achieves high-quality real-time dense matching results for different datasets, including a benchmark dataset, close-range images, and aerial images.

With the derived camera pose information and dense image matching results from the previous steps, 3D data (e.g., 3D point clouds) can be generated through photogrammetric space intersection (triangulation). However, existing methods seldom focuses on the efficiency of 3D data generation for real-time applications. To overcome this limitation, this research proposes a parallel architecture based framework that performs multi-image triangulation based on an optimised angle-based error metric. The proposed framework adopts a one-track-one-line strategy to exploit the parallel computing power of GPU and can achieve real-time performance. The performances of the proposed 3D data generation framework have been demonstrated by two application scenarios: (1) real-time 3D point cloud generation from aerial images, and (2) real-time 3D human motion acquisition and monitoring. The experimental results show that the proposed framework can process a pair of aerial images in 156 ms on average and generate a 3D point cloud incrementally displayed by an optimised grid map in real time. Moreover, the proposed framework was adopted to transfer human body feature from 2D to 3D. Experimental results show that the developed methods can capture and monitor 3D human motion at 17 FPS and achieve centimetre-level accuracy within a 15 m distance.

In conclusion, real-time photogrammetry offers significant benefits in enabling real-time 3D data acquisition and modelling for diverse applications and domains. This research presents novel contributions to the photogrammetry field by extending it to real-time photogrammetry. The novel approaches and implementations including real-time cross-view feature matching for camera pose determination, real-time dense image matching, and real-time triangulation for 3D data generation can serve as foundations for further research and development in real time photogrammetry. The developed real-time photogrammetric methods and systems will have great potential for various applications, such as more intelligent UAV applications based on real-time feedback control, disaster emergency response from real-time 3D mapping, enhanced human tracking and monitoring assisted with real-time 3D data, and autonomous driven supported by real-time 3D pose determination and 3D mapping of the surrounding environment.

Publications Arising from the Thesis

Journal Papers:

[2] **Chen, L.**, Wu, B., Duan, R., 2023. Real-Time Cross-View Feature Matching and Camera Pose Determination for 3D Point Cloud Generation. ***Photogrammetric Engineering & Remote Sensing***, under review.

[1] **Chen, L.**, Wu, B., Zhao, Y., Li, Y., 2021. A Real-Time Photogrammetric System for Acquisition and Monitoring of Three-Dimensional Human Body Kinematics. ***Photogrammetric Engineering & Remote Sensing***, 87(5), pp. 363-373.

[3] Li, Z., Wu, B., Liu, W. C., **Chen, L.**, Li, H., Dong, J., Rao, W., Wang, D., Meng, Q., Dong, J., 2022. Photogrammetric Processing of Tianwen-1 HiRIC Imagery for Precision Topographic Mapping on Mars. ***IEEE Transactions on Geoscience and Remote Sensing***, 60, pp. 1-16.

[4] Wu, B., Dong, J., Wang, Y., Rao, W., Sun, Z., Li, Z., Tan, Z., Chen, Z., Wang, C., Liu, W., **Chen, L.**, Zhu, J., Li, H., 2022. Landing Site Selection and Characterisation of Tianwen-1 (Zhurong Rover) on Mars. ***Journal of Geophysical Research: Planets***, 127(4), e2021JE007137.

[5] Wu, B., Dong, J., Wang, Y., Li, Z., Chen, Z., Liu, W. C., Zhu, J., **Chen, L.**, Li, Y., Rao, W., 2021. Characterisation of the Candidate Landing Region for Tianwen-1—China's First Mission to Mars. ***Earth and Space Science***, 8(6), e2021EA001670.

[6] Wu, B., Li, F., Hu, H., Zhao, Y., Wang, Y., Xiao, P., Li, Y., Liu, W., **Chen, L.**, Ge, X., Yang, M., Xu, Y., Ye, Q., Wu, X., Zhang. H., 2020. Topographic and Geomorphological Mapping and Analysis of the Chang'E-4 Landing Site on The Far Side of The Moon. ***Photogrammetric Engineering & Remote Sensing***, 86(4): pp. 247-258.

[7] Hu, H., Wu, B., **Chen, L.**, 2019. Color Balancing and Geometrical Registration Of High-resolution Planetary Imagery for Improved Orthographic Image Mosaicking. ***Planetary and Space Science***, 178, 104719.

Conference Proceedings:

[1] **Chen, L.**, Wu, B., Zhao, Y., 2020. A real-time photogrammetric system for monitoring human movement dynamics. ***The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences***, 43, pp. 561-566.

Acknowledgements

Studying for a PhD in Hong Kong was one of the most valuable experiences of my life. It has been like a long journey full of challenges. In the past five years, we have encountered various challenges and hardships. In the past five years, we have experienced difficulties and hardships. These difficulties have played a significant role in shaping our character and development during this period. There are many critical points in life, and many people and many thanks in life to remember.

Most importantly, I would like to express my deepest gratitude and respect to my supervisor, Prof. WU Bo. Thanks very much for his patient guidance and strong support for my research. I have learned more than just academic knowledge from him. His guidance and wise advice have not only helped me conduct scientific research in the past years and will also continue to benefit me in my future career.

I extend my thanks to thank all the academic and administrative staff at LSGI for supporting this research. In particular, I would like to thank Dr. YAN Wai-yeung for his professional insights on my confirmation. He has provided many pertinent suggestions and support for my research. Thanks to Dr. DUAN Ran for his selfess help and proofreading of this thesis. I also would like to thank the members of PRSlab, ZHU Jiaming, CHEN Zeyu, LI Hongliang, LI Zhaojin for their assistance and company both in my academic pursuits and personal life during my time in Hong Kong. Thanks to former team members Dr. LIU Wai Chung and Dr. HU Han for their valuable advice regarding my research..

Finally, I would like to thank my dear family for their constant understanding, support and pride. I am also very grateful to, Wang Quan, for all his encouragement and companionship.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Research Background

Photogrammetry has a rich historical background and has traditionally been used for extracting three-dimensional (3D) information from two-dimensional (2D) images. However, the introduction of real-time photogrammetry has transformed this domain by facilitating dynamic and interactive 3D reconstruction. As it can promote the generation of accurate and detailed 3D models in real time for diverse industrial applications, real-time photogrammetry has garnered significant attention. Moreover, real-time photogrammetry has emerged as a promising solution to address the growing demand for real-time 3D applications (Saouli, 2019). In particular, 3D models can be captured and reconstructed with immediate or near-instantaneous results by leveraging the principles of photogrammetry and advanced parallel architectures (Wang, 2019).

Real-time photogrammetry has presented a longstanding challenge in the computer graphics domain as considerable amounts of data must be processed to generate high-quality 3D models. The "real-time" aspect refers to the ability to process the images and produce results almost instantly (Technology et al, 1991). Real-time photogrammetry refers to the process of capturing and processing images or video in real time to generate 3D models of objects or environments (Förstner, 2005). The evolution of technology over the years has enabled the realization of real-time photogrammetry. Recent advancements in parallel architectures have facilitated the realisation of real-time 3D applications capable of generating real-time photogrammetric models. In this research, we explore the significance of parallel architectures in revolutionising the field of photogrammetry and their impact on the development of interactive and immersive 3D applications.

1.1.1 Importance of Real-time Photogrammetry and its Applications

Real-time photogrammetry techniques generate 3D models in real-time or near real-time through image capture, feature extraction, camera pose determination, and model

reconstruction. Real-time photogrammetry is a rapidly evolving field that has garnered considerable attention due to its wide ranging applications across various industries, such as unmanned aerial vehicles (UAVs), medicine, human tracking and monitoring, land surveying, emergency rescue operations, architecture, and construction.

In the UAV domain, real-time photogrammetry facilitates autonomous navigation and obstacle avoidance. Drones with real-time photogrammetry capabilities can capture images of their surroundings and generate 3D models in real time, which can promote obstacle detection and path planning, enabling UAVs to make immediate decisions and navigate safely in dynamic environments. Obstacle detection can be realised by comparing the current environment with the reconstructed model. Moreover, algorithms such as simultaneous localisation and mapping (SLAM) can be used to track the position and orientation of the drone relative to the 3D model (Frosi et al., 2023; Roy et al., 2023; Sawada and Hirata, 2023). By analysing the differences between the images captured in real time and reconstructed model, obstacles such as buildings, trees, or power lines can be detected, allowing the UAV to adjust its flight path for avoiding collisions.

Path planning is another key task facilitated by real-time photogrammetry. The 3D models generated in real time can accurately represent the environment, including terrain and obstacles. UAVs can leverage this information to identify optimal paths and trajectories for their missions. For example, in search and rescue operations, drones can apply real-time photogrammetry to rapidly generate 3D models of the disaster area. Based on these models, efficient paths can be planned to navigate through debris and locate survivors (Daud et al., 2022; Kim et al., 2019). Furthermore, the real-time nature of photogrammetry allows UAVs to adapt to dynamic environments. As the drone traverses its flight path, it can continuously capture images and update the 3D model in real time. This approach provides the UAV with up-to-date information regarding its surroundings, enabling it to react to environmental changes, such as moving objects or newly emerging obstacles (Shang and Shen, 2018; Vasudevan et al., 2016).

In the medical field, real-time photogrammetry has proven valuable across various aspects of surgical procedures. Surgeons can capture intraoperative images of the patient's anatomy and use real-time photogrammetry algorithms to reconstruct 3D

models of the surgical site. These 3D models enable precise visualisation, providing surgeons with enhanced spatial understanding and assisting them in making critical decisions during the procedure. In this manner, 3D models can help improve surgical outcomes and reduce patient risk. Postoperative assessment is another crucial application of real-time photogrammetry. By comparing preoperative images with postoperative images and 3D models, healthcare professionals can objectively evaluate the surgical outcomes, assess the effectiveness of the procedure, and monitor the recovery progress. Thus, real-time photogrammetry techniques can help identify potential complications or issues that may necessitate further intervention or adjustment to the treatment plan (Patias, 2002; Treleaven and Wells, 2007).

Real-time photogrammetry can facilitate human tracking and monitoring in various domains. By using camera networks and real-time photogrammetry algorithms, the 3D movements of individuals can be tracked and reconstructed in real time. Such frameworks have been applied in surveillance, crowd monitoring, and behaviour analysis. In surveillance scenarios, real-time photogrammetry enables the tracking and identification of individuals in real time. By reconstructing the 3D movements of individuals, abnormal behaviours or potential security threats can be effectively detected, thereby improving public safety and security (Geiger et al., 2011; Izadi et al., 2011). Crowd monitoring is another area where real-time photogrammetry plays a significant role. Patterns and behaviours can be identified by analysing the 3D movements and interactions of individuals within a crowd. This approach has implications for crowd management, crowd flow optimisation, and prevention of overcrowding in public spaces (Junior et al., 2010). Real-time photogrammetry can also facilitate behavioural analysis, in which the intentions or emotional states of individuals can be inferred by examining their movements and postures. By reconstructing the individuals' 3D movements, subtle cues and patterns can be identified, contributing to applications in psychology, human–computer interactions, and intelligent surveillance (Chen et al., 2021; Sarafianos et al., 2016).

The use of real-time photogrammetry in the architecture and construction industry can promote collaboration among the various stakeholders involved in a construction project. By generating real-time 3D models, architects, engineers, contractors, and clients can visualise the project in a comprehensive and interactive manner. Such

visualisation can foster effective communication and allow stakeholders to better understand the design intent and construction progress (Balali et al., 2018). Real-time photogrammetry also supports decision-making processes during the construction phase. Construction professionals can utilise real-time 3D models to assess construction progress, monitor quality control, and evaluate compliance with design specifications. In addition, these models enable virtual inspections, reducing the need for physical site visits and enhancing the overall efficiency (Shang and Shen, 2018).

1.1.2 Challenges in Real-time Photogrammetry

One of the main challenges in achieving real-time performance in photogrammetry is the high computational requirements. Generating 3D models from images requires extensive computational power, especially when dealing with large datasets or complex scenes. The algorithms used for feature extraction, matching, and reconstruction are computationally intensive, often requiring significant processing time.

To overcome these challenges, the use of parallel architectures has emerged as a promising solution (La Salandra et al., 2021; Moustafa et al., 2016). Parallel architecture involves the use of multiple processors or computing units to divide and conquer tasks, thereby increasing the processing speed and efficiency. By distributing the computational load across multiple units, parallel architectures can significantly improve the efficiency of real-time photogrammetry applications. Several researchers have recognised the potential of parallel architectures in improving the real-time capabilities of photogrammetric applications.

For instance, La Salandra et al. (2021) developed a parallel algorithm that leverages the power of multiple computing units to process images simultaneously. This approach divided image processing into subtasks assigned to different computing units, which enabled parallel execution and reduced the overall processing time. The results demonstrated the feasibility of achieving real-time photogrammetry using parallel architectures. In addition to La Salandra et al., other researchers have explored the use of parallel architecture in real-time photogrammetry. For example, Moustafa et al. (2016) developed a parallel framework that used multi-core CPUs and graphics

processing units (GPUs) to accelerate the photogrammetric pipeline. Real-time performance was achieved by leveraging the parallel processing capabilities of GPUs for computationally intensive tasks, such as dense reconstruction.
These studies collectively highlight the potential of parallel architectures as a promising solution for achieving real-time photogrammetry. By harnessing the power of multiple processors or computing units, parallel architectures facilitate the efficient distribution of computational tasks, resulting in faster processing times and enhanced real-time performance. However, the effectiveness of parallel architecture may depend on various factors, such as the application, dataset size, and hardware configuration. Further research and optimisation efforts are required to fully exploit the benefits of parallel architectures in real-time photogrammetry fully.

In conclusion, real-time photogrammetry plays a crucial role in real-time 3D applications. However, the realisation of real-time performance is challenging due to the high computational requirements involved. Parallel architectures have emerged as a promising solution for enhancing efficiency by leveraging multiple processors or computing units. Researchers have explored the use of parallel architectures in various photogrammetric algorithms and demonstrated their potential in achieving real-time photogrammetry.

1.1.3 Advances in Parallel Architecture for Real-time Photogrammetry

Parallel architectures are pivotal in enhancing the performance and efficiency of various computational tasks, including real-time photogrammetry. In recent years, significant advancements have been made in parallel computing, enabling researchers to exploit the power of multiple processors or computing units to accelerate complex computations.

In parallel architectures, multiple tasks or subtasks are simultaneously executed, resulting in the distribution of the workload among multiple processors or computing units. By dividing a task into smaller units and processing them in parallel, the overall processing time can be significantly reduced, leading to increased efficiency (Foster

and Kesselman, 2003). This computational power of multiple processors can be harnessed to rapidly execute computationally intensive algorithms.

Parallel architectures can be implemented using various strategies, such as multi-core CPUs, GPUs, and specialised hardware accelerators. Multi-core CPUs consist of multiple processing units within a single chip, enabling concurrent execution of multiple threads or processes. Conversely, GPUs excel in parallel processing due to their large number of cores and high memory bandwidth. GPUs have been extensively used in graphics rendering and are increasingly being leveraged for general-purpose parallel computing (Owens et al., 2007). Specialised hardware accelerators offer dedicated hardware components tailored for specific computational tasks, such as field-programmable gate arrays (FPGAs) and application-specific integrated circuits. These accelerators can enable efficient parallel processing due to their custom-designed architecture and optimised circuits (Pesce et al., 2022; Thomasian, 2022).

The effectiveness of parallel architectures depends on several factors, such as the nature of the task, degree of parallelism in the algorithm, and hardware configuration. Certain tasks exhibit higher inherent parallelism, enabling more efficient utilisation of parallel architecture, whereas others may involve dependencies or sequential portions that limit the achievable level of parallelism. In the context of real-time photogrammetry, parallel architectures have facilitated efficiency improvements and the realisation of real-time performance. The computational load can be distributed by leveraging multiple processors or computing units, thereby reducing the processing time required for complex photogrammetric algorithms. Several studies have demonstrated the effectiveness of parallel architecture in tasks such as feature extraction, matching, bundle adjustment, and dense reconstruction (Knyaz et al., 2020).

In summary, parallel architectures have emerged as a powerful tool for enhancing the performance and efficiency of computationally demanding tasks, particularly in real-time photogrammetry. These frameworks enable the simultaneous execution of multiple tasks or subtasks across multiple processors or computing units, significantly reducing the overall processing time. Various strategies, such as multi-core CPUs, GPUs, and specialised hardware accelerators, can be used to implement parallel

architecture. The successful utilisation of parallel architectures in real-time photogrammetry highlights their potential in enabling real-time 3D applications.

1.2 Research Motivation

Research on real-time photogrammetry based on parallel architectures for 3D applications must be conducted given the growing demand for advanced visual navigation, human tracking and monitoring, and 3D mapping capabilities in various fields, such as UAVs, surveillance systems, and urban planning.

Visual navigation for UAV positioning has attracted significant attention due to the expanding applications of UAVs across various industries, such as aerial photography, disaster management, and package delivery. Real-time photogrammetry plays a crucial role in UAV navigation by enabling the generation of 3D models from onboard images. However, real-time performance must be achieved to ensure accurate and precise UAV positioning, thereby enabling obstacle detection, collision avoidance, and precise control. Parallel architectures can enhance the execution efficiency of real-time photogrammetry algorithms, allowing UAVs to navigate in real time with improved accuracy and efficiency.

Human tracking and monitoring systems rely on real-time photogrammetry to capture and analyse human movements in complex environments. These systems are widely used in security surveillance, sports analysis, and healthcare monitoring. Real-time photogrammetry combined with parallel architectures can rapidly extract human pose information and enable motion tracking, facilitating immediate response and analysis. By using parallel architecture, the processing time can be reduced, thereby promoting the timely detection and tracking of human activities and enhancing safety and security measures.

The demand for 3D mapping has significantly grown, driven by applications such as urban planning, virtual tourism, and archaeological preservation. Real-time photogrammetry serves as a valuable tool for capturing the geometry and texture of real-world objects and scenes. However, the generation of high-quality 3D data in real time is computationally intensive, especially when dealing with large-scale environments or dynamic scenes. Parallel architectures can address these challenges by distributing the computational workload across multiple processors or computing units,

facilitating faster and more efficient 3D data generation. Thus, real-time photogrammetry based on parallel architectures can revolutionise the way we visualise and interact with 3D maps, enabling dynamic updates and immersive experiences.

In conclusion, the motivation for research on real-time photogrammetry based on parallel architectures stems from the increasing demand for advanced visual navigation, human tracking and monitoring, and 3D data generation capabilities. Notably, these applications necessitate real-time performance for accurate and efficient operations. By leveraging parallel architecture, the execution efficiency of real-time photogrammetry algorithms can be significantly enhanced, resulting in improved positioning accuracy, enhanced human tracking and monitoring, and efficient 3D map generation. This research aims to advance the field by exploring the potential of parallel architectures in addressing the computational challenges associated with real-time photogrammetry in the relevant domains.

1.3 Objectives and Contributions

This thesis presents novel approaches and strategies for real-time photogrammetry applications based on parallel architectures. The objective is to address the challenges associated with achieving real-time performance and demonstrate the effectiveness of parallel processing in improving the efficiency and accuracy of 3D data generation. This research provides a practical and adaptable solution that can be readily implemented in various domains requiring real-time photogrammetry. The objectives and contributions of this research can be summarised as follows:

(1) Developing approaches and algorithms that enable accurate and efficient matching of images captured from different viewpoints. This work focuses on feature-based cross-view image matching and camera pose determination, including techniques such as deep-learning feature detection and feature-based similarity search for image matching and retrieval methods. The contributions of this work include the advancement of state-of-the-art cross-view image-matching techniques to acquire camera poses for aerial robot visual navigation in a global positioning system (GPS) denied environment and provision of insights into the effectiveness and efficiency

of different approaches. The results of comparing the accuracy of our proposed method with other popular image matching and retrieval methods show that our method is higher than other popular methods at the top one and top five of the search results from aerial image datasets, with an accuracy of ~60% and ~73%, respectively. Eventually, this method is combined with a deep learning-based VO method to achieve real-time camera pose determination in GPS denied environments only using aerial images, where the RMSE can reach 4.7 m, and the efficiency of the algorithm execution sustains around 12 FPS.

(2) Improving the efficiency of dense image matching algorithms for generating accurate depth maps and 3D reconstructions. Dense image matching involves the computation of the correspondences between pixels in multiple images, which is crucial for generating accurate depth maps and 3D reconstructions. This work explores semi-global matching (SGM)-based dense image matching, taking into account efficiency considerations such as matching costs and similarity measurements. The contributions of this work include the optimisation of dense image matching techniques by leveraging GPU acceleration and derivation of insights into the performance enhancement of these algorithms. Compared with the traditional SGM method, our proposed method, not only improves the disparity generated using benchmark dataset, close-range and aerial images, but also surpasses the traditional SGM method in accuracy and efficiency.

(3) Development of strategies that enable real-time generation of 3D data for real-time 3D photogrammetric applications. This work first explores GPU-accelerated triangulation methods for the real-time generation of 3D point clouds. Subsequently, algorithms from earlier chapters on real-time photogrammetry are incorporated to facilitate the instantaneous acquisition and monitoring of 3D human kinematics. Contributions of this work include advancing complex triangulation algorithms from static to real-time implementations. The advantage of photogrammetry over traditional machine vision algorithms is the ability to acquire 3D information in large-scale images. The application of this advantage is demonstrated in practical scenarios such as aerial real-time 3D point cloud generation, motion capture, and analysis of human kinematics. This work serves as a valuable reference for the development of real-time photogrammetry applications.

Overall, the research objectives and contributions are focused on enhancing the efficiency and accuracy of real-time photogrammetry applications. By addressing the challenges encountered by the relevant algorithms in different scenarios, this research aims to advance the field of real-time photogrammetry, provide valuable insights, and identify techniques for various domains that rely on real-time 3D data generation and analysis.

1.4 Outline of the Thesis Structure

This thesis consists of six chapters, and the remainder is organised as follows:

Chapter 2 presents a comprehensive literature review establishing the background and fundamentals principles of real-time photogrammetry. Furthermore, state-of-the-art studies on visual navigation and parallel architectures in 3D photogrammetric applications are explored. The final section provides a brief overview of the latest research and developments in real-time aerial mapping.

Chapter 3 focuses on real-time cross-view image matching and camera pose determination. The objective of the research described in this chapter is to identify approaches and algorithms for accurate and efficient feature-based image matching from different image sources. Additionally, the integration of photogrammetric methods and computer vision algorithms is proposed to achieve accurate visual navigation for aerial robots.

Chapter 4 focuses on real-time dense image matching by leveraging GPU acceleration. It describes GPU architecture and performance analyses, GPU-based centre-symmetric census transform (CT) and matching cost computation, and optimisation of disparity map generation and parallel computing. By leveraging the computational power of GPUs, the efficiency and speed of dense image-matching algorithms can be significantly improved, enabling real-time processing for depth estimation in different scenarios.

Chapter 5 explores real-time triangulation for 3D data generation, focusing on techniques for generating point clouds and extracting 3D information from aerial images. Real-time triangulation is implemented with multi-threading and deep-learning algorithms to acquire and monitor 3D human body kinematics. Using real-time algorithms and strategies, experimental evaluations of real-time photogrammetric applications are conducted across various domains that rely on real-time 3D data generation and analysis.

Chapter 6 presents the concluding remarks and highlights potential future research directions.

This thesis aims to develop and evaluate real-time photogrammetric methods for 3D applications. Figure 1.1 presents the logical relationships among the chapters in this thesis. The literature review in Chapter 2 elaborates upon the fundamental principles and state-of-the-art approaches referred to in the main body of the thesis, forming a solid basis for the analysis and discussions in subsequent chapters.

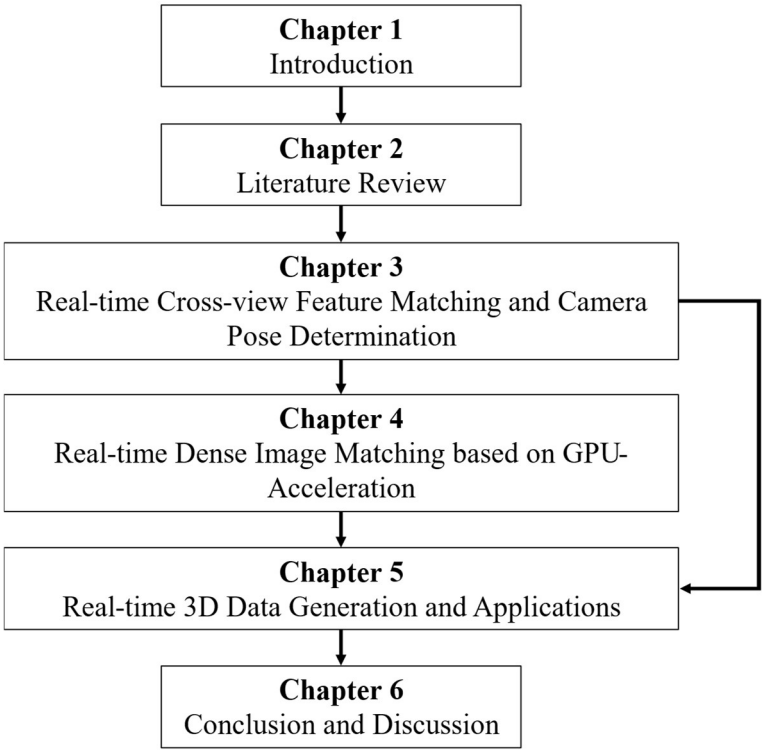


Figure 1.1 Logical connections across chapters

Real-time photogrammetry poses significant challenges in terms of computational efficiency, accuracy, robustness, and scalability. To address these challenges, Chapter 3 to Chapter 5 of this thesis proposes novel algorithms and techniques that leverage parallel computing, computer vision, and deep learning to achieve real-time performance in different photogrammetric tasks. Chapter 6 concludes the thesis by summarising the main contributions and findings, discussing the limitations and challenges, and suggesting future directions for research.

Chapter 2 Literature Review

Photogrammetry is defined as "the science or art of obtaining reliable measurements by means of photographs" (Manual of Photogrammetry, 1966). A more recent definition by the American Society for Photogrammetry and Remote Sensing (ASPRS) is "the art, science, and technology of obtaining reliable information about physical objects and the environment through processes of recording, measuring, and interpreting photographic images and patterns of recorded radiant electromagnetic energy and other phenomena." In simple terms, photogrammetry enables the acquisition of 3D measurements (e.g., position, orientation, shape, and size) of objects from photographs. The fundamental principle of photogrammetry is triangulation, which involves the calculation of the intersection of an object's image points from multiple perspectives to determine its position. Through the analysis of the geometry and features within a set of images, photogrammetry enables the creation of photorealistic and precise 3D models.

This chapter discusses the fundamentals of photogrammetry, encompassing key concepts and techniques. Moreover, it provides an overview of how these principles and foundations are applied in photogrammetry and highlights their significance and relevance in various research fields. Furthermore, this chapter delves into the latest research advancements in photogrammetric applications with parallel architectures, aiming to uncover potential research directions and emerging trends in the field. By elucidating the foundational principles, applications in relative disciplines, and state-of-the-art research progress, this chapter aims to provide a comprehensive understanding of photogrammetry and clarify the scope for its advancement.

2.1 Fundamentals of Photogrammetry

This section presents a comprehensive literature review on the fundamental aspects of photogrammetry, focusing on feature point extraction and matching, dense image matching, triangulation, and space resection. These methods are essential in subsequent studies and are discussed in detail in this section, including their principles and applications. The review includes a thorough examination of the literature, offering an

in-depth understanding of the key concepts, techniques, and advancements in photogrammetry. Furthermore, the significance and relevance of these methods in various applications are highlighted, allowing the readers to grasp the fundamental principles and state-of-the-art research associated with photogrammetry.

2.1.1 Feature Extraction and Matching

Photogrammetry, which is aimed at extracting 3D information from 2D images, relies on feature detection and matching. These methods are fundamental to numerous photogrammetry applications, such as 3D reconstruction, image registration, and object tracking. Feature detection and matching techniques enable the accurate alignment of images, extraction of depth information, and creation of high-quality 3D models through the identification of distinctive points or regions and establishment of correspondences between these points. This section explores the various feature detection and matching approaches, including traditional and deep-learning techniques. Understanding these methods is essential for the advancement of photogrammetry and development of more robust and effective algorithms.

2.1.1.1 Traditional Methods

Feature matching and detection are pivotal in photogrammetry and serve as the foundation for subsequent processes. Feature detection involves the identification of salient points or regions within images, which have unique characteristics, such as corners, edges, or textures. These features can serve as reliable reference points for subsequent computations. Upon detecting features, correspondences are established between corresponding features in different images. This correspondence information is vital for tasks such as image alignment and 3D reconstruction, as it enables the tracking of the same feature points across multiple images.

i) Scale-Invariant Feature Transform (SIFT)

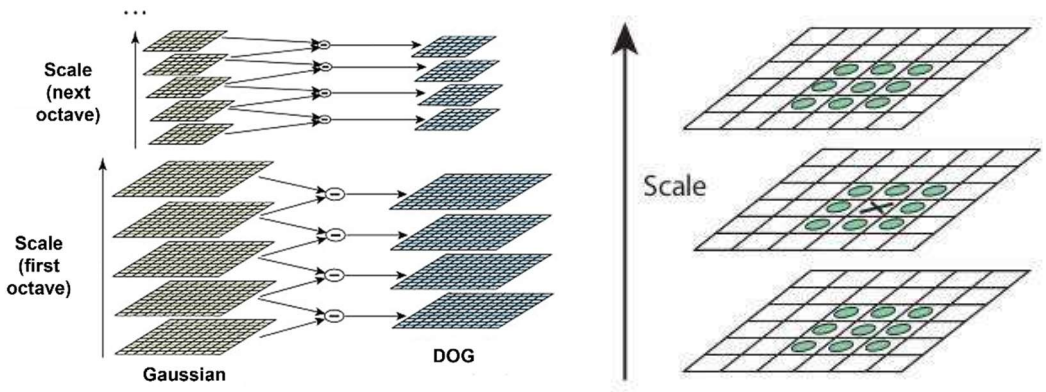
One notable traditional feature detection and matching method is the SIFT (Lowe, 2004). The SIFT algorithm was developed to address challenges associated with scale, rotation, and affine transformations, which are commonly encountered in images. The objective is to achieve invariance to these transformations by constructing a scale-space representation of the image and identifying keypoints at multiple scales.

The SIFT algorithm begins by constructing a scale-space pyramid through the repeated convolution of the image with Gaussian filters at different scales. This process results in a series of blurred images at different levels of scale. Next, the difference of Gaussians (DoG) is computed by subtracting adjacent scales in the scale-space pyramid, as shown in Figure 2.1(a). The DoG images enhance the regions with significant intensity variations and potential keypoints. The blurred image pyramid L is obtained using the following equations:

$$L(x, y, \sigma) = G(x, y, \sigma) \times I(x, y) \tag{2.1}$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2.2}$$

where G is the Gaussian blur operator, I is an input image, (x, y) denotes the location of each pixel in I, and σ is the scale factor of the corresponding pixel.



(a) DoG processing (b) Keypoint represented in scale

Figure 2.1 Overview of SIFT algorithm using DoG (Bradski and Kaehler, 2008).

The SIFT algorithm applies a process named keypoint localisation to identify the keypoints. As shown in Figure 2.1(b), the algorithm examines the extrema in the DoG scale space to identify keypoints that are stable and invariant to changes in scale. This process involves comparing a pixel with its 26 neighbours across the current and adjacent scale levels. Keypoints that do not have extreme values are discarded. After identifying the keypoints, SIFT descriptors are computed to represent the local image structure around each keypoint. These descriptors capture the gradient magnitudes and orientations within the local neighbourhoods of the keypoints. They are highly distinctive and invariant to changes in scale, rotation, and affine transformations. In particular, the descriptors are constructed by dividing the local region surrounding the keypoint into subregions and computing the gradient orientations and magnitudes within each subregion. The resulting descriptor is a high-dimensional vector that encodes the local image information.

The SIFT algorithm has been applied in various areas of photogrammetry and computer vision. One of its primary applications is feature matching (Hua et al., 2010), in which SIFT descriptors are used to establish correspondences between keypoints in different images. By comparing the SIFT descriptors, similarity measures such as the Euclidean distance (Hua et al., 2012) or cosine similarity (Wang et al., 2022) can be used to identify the best matches. Feature matching using SIFT has been widely implemented in applications such as image stitching (Zhang et al., 2017), object recognition (Alhwarin et al., 2008), and image retrieval (Chhabra et al., 2020), where robust and accurate matching is essential. SIFT also plays a significant role in image registration, which involves aligning multiple images into a common coordinate system (Ma et al., 2016). The distinctive SIFT keypoints and descriptors facilitate the estimation of geometric transformations, such as affine or perspective transformations, to achieve accurate alignment of images. Image registration using SIFT has been applied in medical imaging, where the precise alignment of images is crucial (Allaire et al., 2008). Additionally, SIFT has been used in 3D reconstruction tasks (Du et al., 2011). By extracting SIFT keypoints from images captured from different viewpoints, correspondences between keypoints can be established, enabling the estimation of camera poses and reconstruction of 3D scenes.

Despite its effectiveness, the SIFT algorithm exhibits computational complexity and significant memory requirements, which may limit its real-time performance on large-scale datasets. Owing to these characteristics, SIFT cannot be effectively applied in real-time and resource-constrained environments. Moreover, SIFT is sensitive to noise and blur, and significant changes in the viewpoint can affect its performance. Because this algorithm primarily focuses on local features, it cannot effectively capture global contextual information. Additionally, the performance of SIFT is vulnerable to illumination changes.

ii) Speeded-Up Robust Features (SURF)

SURF is a feature detection and description method that was developed by Du et al. (2011) as a more efficient and accurate alternative to SIFT. SURF uses integral images to approximate the Laplacian of Gaussian (LoG), enabling rapid calculation of the scale-space extrema. Owing to the use of integral images, the computational complexity of SURF is lower than that of traditional methods, rendering SURF well-suited for real-time applications.

One of the key contributions of SURF is its ability to approximate the LoG using integral images, facilitating rapid computation of scale-space extrema, which is essential for detecting features at different scales. The integral images can be computed as follows:

$$I_{\sum}(x) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(i,j) \tag{2.3}$$

where $I_{\sum}(x)$ represents an integral image at location (i,j). The integral image is the sum of all pixels in the input image I within a rectangular region formed by the origin and x. To further refine the keypoints, SURF applies the Hessian-matrix-based detector technique. The Hessian matrix at each potential keypoint is computed based on the second-order derivatives of the Gaussian scale-space and used to assess the stability and repeatability of keypoints. Points with low contrast or poorly defined locations are discarded, resulting in a more accurate set of keypoints. For scale adaptation, the image

is filtered using a Gaussian kernel. Thus, given a point $X = (x, y)$, the Hessian matrix $H(x, \sigma)$ at x and scale σ is defined as

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \tag{2.4}$$

where $L_{xx}(x, \sigma)$ is the convolution of the Gaussian second-order derivative with the image I at point x, and $L_{xy}(x, \sigma)$ and $L_{yy}(x, \sigma)$ are similarly defined.

The SURF descriptor is based on the Haar wavelet responses in a square region around the feature point. The descriptor is designed to be compact and efficient while still capturing adequate information to distinguish different features. The square region surrounding the keypoint is divided into smaller square subregions, and the Haar wavelet responses are computed for each subregion. The responses are then accumulated to form a feature vector normalised for illumination and contrast invariance. The resulting descriptor is 64-dimensional and can be efficiently compared using metrics such as the Euclidean distance or cosine similarity.

The computationally efficiency of SURF is particularly advantageous in large-scale photogrammetry projects requiring numerous images to be processed (Afriansyah et al., 2019). The rapid computation of SURF features enables prompt analysis of extensive image datasets, which helps reduce the time required for image matching and enables more efficient photogrammetric workflows. Additionally, SURF's robustness to scale and rotation changes is instrumental when dealing with challenging real-world scenarios in photogrammetry. Environmental conditions, camera perspectives, and object variations often introduce image scale and rotation variations (Teke et al., 2011). As SURF features are robust to these changes, accurate and reliable matching can be achieved across diverse image conditions, leading to more accurate 3D reconstructions. Another primary application area is the generation of dense point clouds (Diskin and Asari, 2013) and 3D reconstructions (Zhang et al., 2014a). By leveraging SURF features for image matching, researchers can accurately align images and triangulate the corresponding points to reconstruct the 3D structure of a scene. The high speed and robustness of SURF features help enhance the efficiency and accuracy of reconstruction. Additionally, SURF features have been used in photogrammetric applications such as

image-based localisation and camera pose estimation (Kim et al., 2014; Sheta et al., 2012). By matching SURF features between reference and new images, researchers can accurately determine the position and orientation of a camera with respect to a given scene.

Despite its computational efficiency and robustness, SURF has several limitations. In scenarios involving significant viewpoint changes or occlusions, SURF may not be as accurate as more complex methods, such as SIFT. Furthermore, the performance of SURF depends on the lighting conditions, as it lacks explicit illumination invariance.

iii) Oriented FAST and Rotated BRIEF (ORB)

ORB, proposed by Rublee et al. (2011), is an effective alternative to SIFT and SURF for feature detection and matching in the field of computer vision. ORB combines the efficiency of the FAST (features from accelerated segment test) corner detector with the robustness of the BRIEF (binary robust independent elementary features) descriptor. This combination renders ORB a popular choice for real-time applications that require a balance between speed and accuracy.

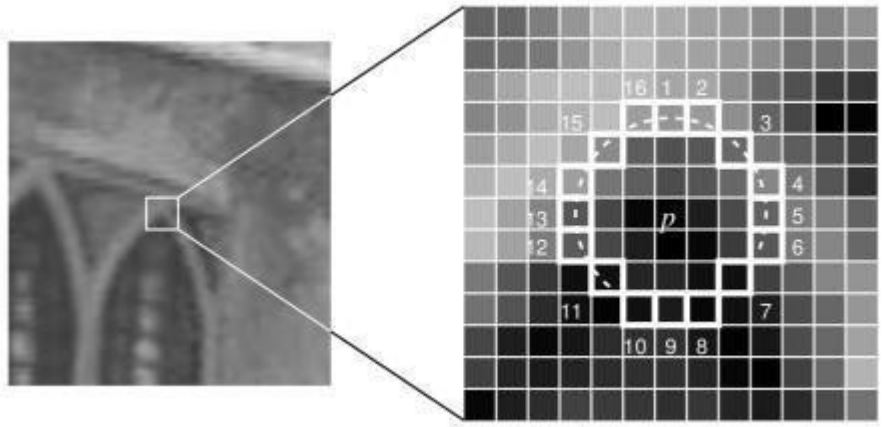


Figure 2.2 Overview of the FAST feature detector (Rosten and Drummond, 2006)

The FAST corner detector operates by analysing the intensity patterns in a circular neighbourhood of each pixel. By comparing the intensity values of the central pixel with those of its neighbouring pixels, FAST can determine whether the pixel is a corner. The FAST detector examines a set of pixels in a circle with a radius of three pixels to

identify a corner. As shown in Figure 2.2, the detector compares the intensity value of the central pixel with the intensities of 16 surrounding pixels p located at the 12, 3, 6, and 9 o'clock positions, as well as four additional pixels on the diagonals. If a sufficient number of consecutive pixels have intensities greater than that of the central pixel plus a threshold or lower than that of the central pixel minus the threshold, the central pixel is classified as a corner.

This high-speed corner detection method has several advantages. First, it compares a small number of intensity values and is thus computationally efficient. Second, it is robust to image noise as only a limited number of pixel comparisons are required for corner identification. Lastly, its simplicity and speed make it suitable for real-time applications that require quick and efficient feature extraction, such as robotics, augmented reality, and video analysis.

The BRIEF descriptor used in ORB is a compact binary descriptor that encodes the intensity comparisons between pairs of pixels. The fundamental concept of BRIEF is to generate a set of binary tests based on randomly selected pixel pairs and compute a binary code that represents the results of these tests. The resulting binary code can be considered a unique fingerprint of the local image structure around the keypoint. The use of binary codes enables efficient matching of image features and reduces the memory required for storage and computation. The binary test can be defined as

$$\tau(p; x, y) = \begin{cases} 1: p(x) < p(y) \\ 0: p(x) \geq p(y) \end{cases} \tag{2.5}$$

where τ represents the binary test, and $p(x)$ is the intensity of p at a point x. The BRIEF descriptor capture local image properties by comparing the intensities of pixel pairs. To achieve computational efficiency and compactness, binary strings are used to represent these comparisons. The binary tests are designed to be simple and fast, typically involving the calculation of intensity differences between pixel pairs at specific locations. The outcome of each test is encoded as either "1" or "0," which indicate whether the intensity of the first pixel is greater than the second pixel. Repeating this process for multiple pixel pairs generates a binary code that represents the unique intensity comparison pattern of a particular image patch. By combining the FAST

corner detector and BRIEF descriptor, ORB achieves both efficiency and robustness in feature extraction. The corner detection step identifies keypoints, which are then described by the BRIEF descriptor. These descriptors capture the distinctive properties of the keypoints, enabling accurate feature matching across different images or frames.

ORB has been widely applied in computer vision tasks such as object recognition, image stitching, and SLAM. In object recognition, ORB features are used to identify and track objects in images or videos (Rosten and Drummond, 2006). In SLAM, ORB features are used to estimate the camera position and map the environment in real-time (Mur-Artal et al., 2015). ORB is also valuable in dense image matching, which involves determining the correspondences between every pixel in a pair of images (Chen et al., 2020). Dense matching is essential for tasks such as surface reconstruction, orthophoto generation, and digital elevation model (DEM) generation. By leveraging the robustness and efficiency of ORB, dense image matching algorithms can accurately match pixels across images, enabling the generation of dense and accurate 3D models.

Despite its efficiency and robustness, ORB has several limitations. Specifically, its performance may be limited compared with those of more advanced methods such as SIFT or SURF in scenarios with significant viewpoint changes or image noise or when dealing with image sequences with repetitive patterns or no distinctive features.

2.1.1.2 Deep-Learning Methods

In recent years, deep-learning methods have garnered significant attention in feature detection and matching applications in photogrammetry. These methods leverage the power of artificial neural networks to learn discriminative features and automatically perform robust matching between images. Although traditional feature descriptors, such as SIFT and SURF, have been widely used, deep-learning-based methods exhibit superior accuracy and robustness, especially in challenging scenarios involving viewpoint changes, occlusions, or illumination variations.

i) SuperPoint and SuperGlue

SuperPoint and its companion method SuperGlue have emerged as popular deep-learning-based approaches for feature detection and matching in photogrammetry. SuperPoint, proposed by DeTone et al. (2018), is a fully convolutional neural network (CNN) that concurrently generates dense local feature keypoints and descriptors. This simultaneous generation of keypoints and descriptors enables efficient and accurate feature extraction.

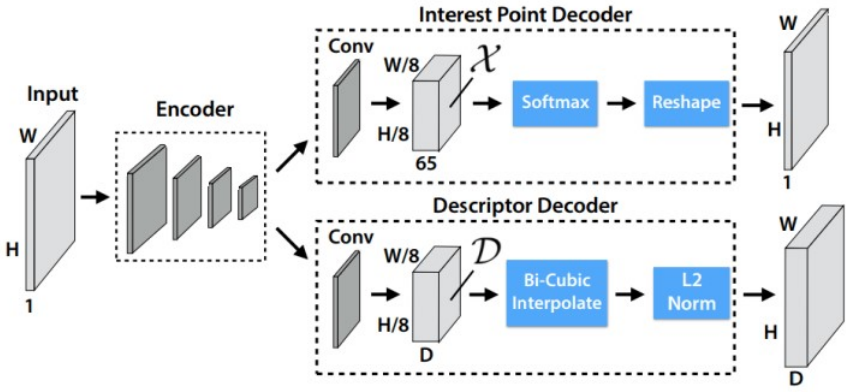


Figure 2.3 Overview of the SuperPoint framework (DeTone et al., 2018)

As shown in Figure 2.3, The input to the SuperPoint network is an image, typically in the form of a two-dimensional array of pixel values. The size of the input image may vary depending on the specific implementation or requirements of the application. The network takes the image as input and processes it through its layers to extract meaningful feature representations. The SuperPoint network generates two main outputs: local feature keypoints and their corresponding descriptors. SuperPoint generates a dense set of local feature keypoints across the input image. These keypoints represent distinctive points in the image that can be used for further analysis, such as feature matching or tracking. Each keypoint is characterized by its location (coordinates) within the image and other properties that describe its local image structure. These properties could include scale, orientation, and possibly other characteristics depending on the network architecture and the specific implementation of SuperPoint. In addition to keypoints, SuperPoint also produces descriptors for each detected keypoint.

Descriptors are compact and informative representations of the local image structure surrounding each keypoint. These descriptors encode information about the gradient or intensity variations near the keypoint. They are designed to be invariant to certain image transformations, such as changes in viewpoint, scale, and lighting conditions, while maintaining discriminative power for accurate feature matching. The descriptors generated by SuperPoint typically take the form of vectors or feature embeddings. The dimensionality of the descriptors can vary depending on the network architecture, but they are often designed to be compact to facilitate efficient storage and matching. The descriptors' length and content capture each keypoint's distinctive characteristics, enabling subsequent matching or recognition tasks.

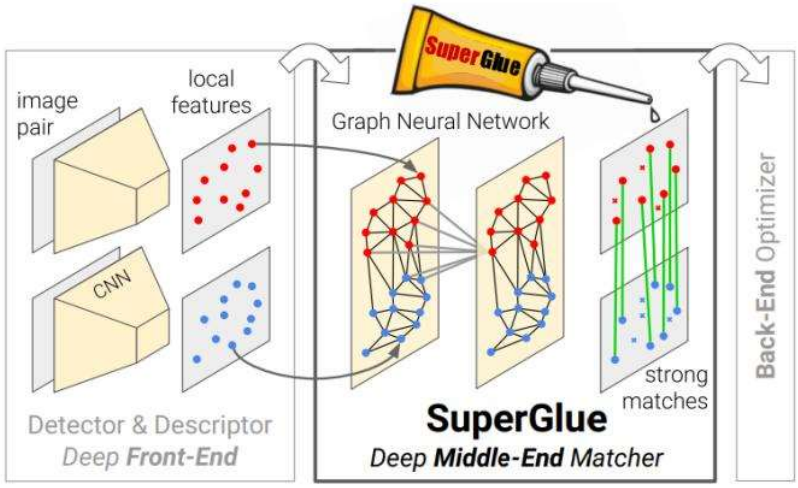


Figure 2.4 Overview of the SuperGlue framework (Sarlin et al., 2020).

The integration of SuperPoint with SuperGlue (Sarlin et al., 2020) enhances the feature-matching process. SuperGlue performs pairwise matching of the extracted keypoints and descriptors to estimate correspondences between multiple images. By using the dense local feature keypoints and descriptors obtained from methods such as SuperPoint, SuperGlue incorporates local and global information to enhance the reliability and accuracy of feature matching. By considering the context and relationships between keypoints across different images, SuperGlue can mitigate ambiguities and enhance the consistency of the matching results.

The input to the SuperGlue network consists of two sets of keypoints and descriptors, extracted from two images by SuperPoint, which are being compared for feature

matching. The SuperGlue network processes these inputs to estimate correspondences between the keypoints from the two images, considering both local and global information to improve the matching reliability and accuracy. The network combines the local descriptors with a global context obtained through a graph neural network module. The output of the SuperGlue network is a set of correspondences between the keypoints from the two images, which represent matches or associations between the keypoints considered the same or similar across the images. The correspondences can be represented as pairs of keypoints, each consisting of a keypoint from the first image and its corresponding keypoint from the second image. Additionally, the network may provide a confidence score or similarity measure for each correspondence, indicating the quality or strength of the match.

SuperPoint and SuperGlue have been widely used in various computer vision tasks. Li et al. (2021) demonstrated the capability of SuperPoint in detecting interest points in texture-less areas in images, with SuperGlue used to perform feature matching and correspondence estimation. These methods typically outperform traditional techniques, rendering them valuable in computer vision applications. Xu et al. (2020) highlighted that SuperPoint and SuperGlue play a crucial role in tasks such as visual SLAM and visual odometry (VO) in visual navigation systems. These methods can be used to extract and match features across consecutive frames, aiding in camera pose estimation and map creation, which are important for navigation and localisation in various domains, such as robotics and autonomous vehicles. In dense reconstruction applications, SuperPoint and SuperGlue can extract and match features in images captured from different viewpoints, enabling the generation of dense point clouds (Deng et al., 2022; Qin et al., 2022). These dense point clouds are crucial for 3D reconstruction tasks, such as the creation of digital surface models (DSMs), generation of orthophotos, and derivation of terrain information. These clouds also facilitate structure-from-motion (SfM) workflows by detecting and matching features across images captured from different viewpoints. This allows for accurate camera pose estimation and the creation of 3D point clouds, which are essential for reconstructing the 3D structure of a scene. Furthermore, SuperPoint and SuperGlue can be used for tie point extraction in change detection analysis in remote sensing (Deshmukh et al., 2023). By comparing the features and correspondences between different periods, these

methods can help identify and analyse changes in land cover, vegetation, or other environmental factors.

Despite their promising performance, SuperPoint and SuperGlue have certain limitations that remain to be addressed. As SuperPoint and SuperGlue are typically trained on synthetic or specific datasets, they may not be effective in handling diverse and complex real-world environments. Their performance may be limited in scenarios with occlusions, uncommon object types, or scenes that differ significantly from their training data (Sarlin et al., 2020). SuperPoint and SuperGlue are somewhat robust to scale and rotation invariance. However, they may struggle in scenarios with extreme scale variations or highly rotated objects. Traditional feature detectors such as SIFT or SURF may outperform them in such challenging scenarios (DeTone et al., 2018). Researchers are actively addressing these limitations and seeking to improve the robustness, generalisation, and efficiency of SuperPoint and SuperGlue (Sun et al., 2021).

ii) SIFT-Based CNN (SIFT-CNN)

SIFT-CNN, which was introduced by Mahendran and Vedaldi (2015), is a pioneering deep-learning-based feature detection and matching method. This framework combines the robustness of SIFT with the discriminative power of CNNs to achieve state-of-the-art results. The principle behind SIFT-CNN is to train a CNN to directly learn feature descriptors from image data, eliminating the need for actively determining descriptors. Traditional feature descriptors, such as SIFT, have been designed based on domain knowledge and manual engineering. However, with the advent of deep learning, it has become possible to train neural networks to automatically learn discriminative feature representations. SIFT-CNN leverages the strength of CNNs in learning hierarchical and invariant feature representations directly from raw image data.

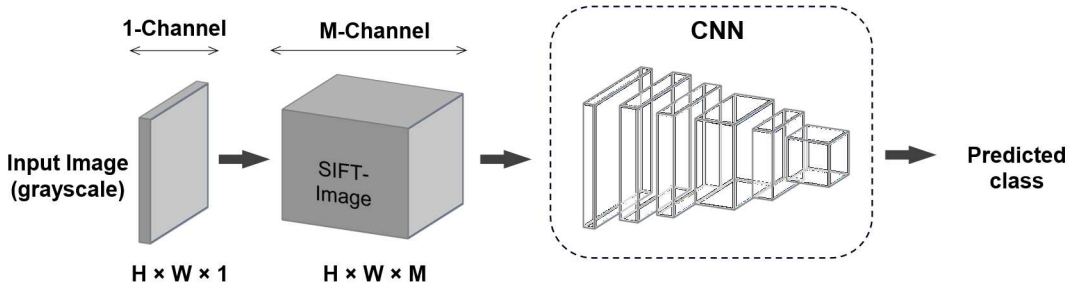


Figure 2.5 Overview of the SIFT-CNN frameworks (Tsourounis et al., 2022)

During training, the CNN learns to transform the raw pixel values of the images into a hierarchical representation of features. This hierarchical representation enables the network to capture local and global information, allowing it to perceive and understand the visual content of the images at multiple scales. The framework of SIFT-CNN is shown in Figure 2.5. The network uses the SIFT image representation as its input and is guided to learn features from the local gradient information of images. This approach enables the SIFT-CNN to implicitly incorporate local rotation invariance. The training process involves minimising a loss function that measures the discrepancy between the predicted features and ground truth descriptors. The network gradually learns to extract features invariant to typical image transformations through this optimisation process.

An essential aspect of the SIFT-CNN method is its integration with the traditional SIFT framework. By aligning with the principles of SIFT, SIFT-CNN ensures compatibility with existing SIFT-based methods. The learned feature descriptors can seamlessly replace SIFT descriptors in various applications without requiring extensive modifications to the existing pipeline. This integration can facilitate a smooth transition from traditional feature detection and matching techniques to deep-learning-based approaches while preserving the accuracy and efficiency of the SIFT framework.

The applications of the SIFT-CNN method are extensive and cover a wide range of computer vision and photogrammetry tasks. In the context of object recognition and classification, the learned feature descriptors can effectively identify and categorise objects within images or videos (Rashid et al., 2019). The robustness of the features to scale, rotation, and illumination variations makes them suitable for object recognition in challenging scenarios. Additionally, the integration of SIFT-CNN with existing SIFT-based methods can help enhance the matching accuracy and efficiency in tasks

such as image alignment (Ye et al., 2018) and 3D reconstruction (Fan et al., 2019). Additionally, the learned features from SIFT-CNN can be used in visual localisation and SLAM (Zhao et al., 2018).

iii) Learnable Feature Descriptor and Descriptor Matcher (LF-Net)

LF-Net is a deep-learning-based feature detection and matching method that has demonstrated exceptional performance in challenging conditions and real-world scenarios. This method, proposed by Ono et al. (2018), combines the advantages of SIFT-based and CNN-based methods and addresses their limitations.

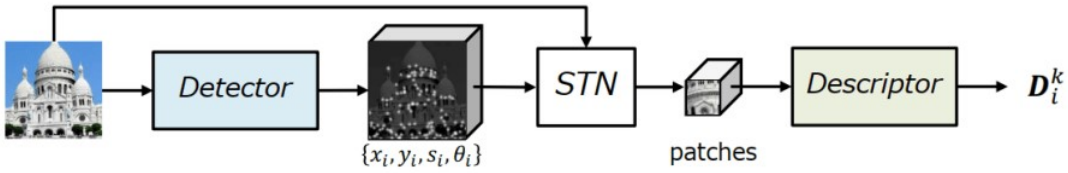


Figure 2.6 Overview of the LF-Net framework

The LF-Net framework takes a pair of images, typically greyscale or RGB images, as the input. These images can be obtained from various sources, such as aerial or satellite imagery, stereo image pairs, or multi-view image sequences. The input images are preprocessed to ensure that they are appropriately aligned and normalised for further processing. The output of LF-Net consists of two main components: local feature keypoints and descriptors, and a global feature vector. The local feature keypoints represent distinct points of interest in the input images, while the descriptors provide a compact representation of the local image patches around each keypoint. These local features are generated by a fully convolutional network within the LF-Net architecture.

LF-Net has been applied in various photogrammetry and remote-sensing-related research areas, including 3D reconstruction, stereo matching, and image registration. Mizginov and Kniaz (2019) used LF-Net for feature detection and matching in multi-view stereo reconstruction. The method achieved state-of-the-art results on the ETH3D benchmark dataset, demonstrating its potential for improving the accuracy and efficiency of 3D reconstruction. Xu et al. (2022) applied LF-Net for feature matching

and registration in high-resolution satellite images. The method improved the accuracy and robustness of registration compared with traditional methods, particularly in challenging scenarios involving large viewpoint variations and low-texture regions.

2.1.2 Space Intersection (Triangulation) and Space Resection

The most fundamental device in photogrammetry is the camera. Cameras acquire the images that are used to generate photogrammetric products. The interior orientation (IO) of the camera is essential for triangulation and space resection. The IO parameters include the focal length and principal centre. Figure 2.8 illustrates the basic pinhole camera model. The following section briefly describes the mathematical background of this model.

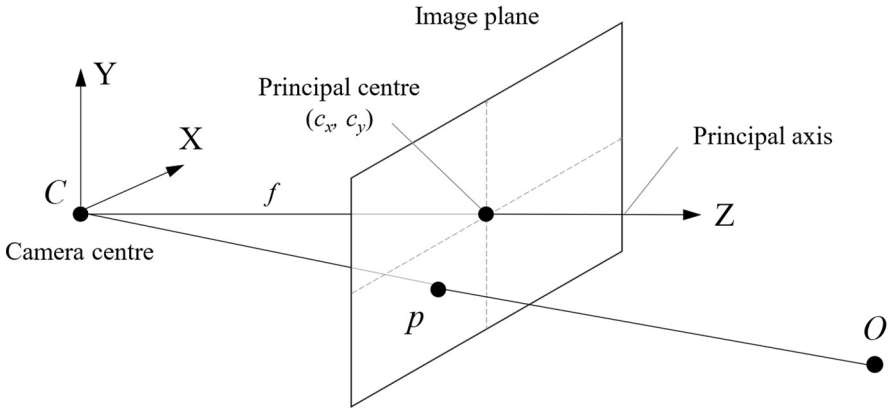


Figure 2.7 Basic camera model with the camera reference (*X, Y, Z*)

The model consists of the optical centre *C* and image plane. When a 3D point *O* is projected onto the camera, it forms an image point *p* at the intersection of the image plane with the line connecting *C* and *O*. The line perpendicular to the image plane and passing through *C* is termed the principal axis (*Z* axis in Figure 2.7). This axis intersects with the image plane at the principal point. The distance between the camera centre and image plane is the focal distance, which is negative in real cameras, where *C* is positioned behind the image plane. The IO matrix *K* of the camera can be defined as follows:

$$K = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

where (f_x, f_y) denotes the focal length of the camera in pixels, and (c_x, c_y) denotes the principal centre of the camera on an image plane. In modern photogrammetry, the IO of the cameras, described by the focal length, principal point, and distortion coefficients, can be determined through camera calibration.

In photogrammetry, collinearity equations are used to model the projection of a 3D object onto a 2D image plane. These equations define the relationship between the coordinates of points in the 3D space (object space) and their corresponding image coordinates in the 2D image space. Both triangulation, also known as space intersection in photogrammetry, and space resection rely on known variables to solve the collinearity equation and are thus fundamentally similar. These principles form the basis for accurately determining the positions and orientations of cameras and the spatial coordinates of points of interest within the captured images. The collinearity equation can be written as

$$\begin{aligned} x - x_0 &= -f \frac{m_{11}(X - X_S) + m_{12}(Y - Y_S) + m_{13}(Z - Z_S)}{m_{31}(X - X_S) + m_{32}(Y - Y_S) + m_{33}(Z - Z_S)} \\ y - y_0 &= -f \frac{m_{21}(X - X_S) + m_{22}(Y - Y_S) + m_{23}(Z - Z_S)}{m_{31}(X - X_S) + m_{32}(Y - Y_S) + m_{33}(Z - Z_S)} \end{aligned} \tag{2.7}$$

This set of equations establishes a direct relationship between an image point (x, y) and its corresponding 3D position (X, Y, Z) within the object space. The principal point (x_0, y_0) denotes the foot of the perpendicular drawn from the image of the principal centre, while f represents the focal length of the camera. The coordinates of the camera centre in the object space are denoted as (X_S, Y_S, Z_S). The rotation matrix, determined by three angles of rotation R (ω, φ, k) between the camera frame and object space, is represented in terms of elements m_{ij} as

$$R = R_\omega R_\varphi R_k = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \tag{2.8}$$

2.1.2.1 Triangulation for Object Location Determination

The exterior orientation (EO) of a camera can be determined through single image space resection, as introduced in the following section. However, the determination of the spatial location of an object point based on the coordinates of the image points from a single image is a challenging task because the EO parameters of a single image only provide information regarding the spatial orientation of the object. To overcome this problem, a stereo image pair can be used by considering the same image point on both images, enabling the determination of the directions of two rays in the same spatial coordinate system. Because these two rays must intersect in space, their point of intersection represents the actual spatial location of the object point (Wolf et al., 2014). This concept can also be demonstrated by the collinearity equation (Eq. 2.7).

Figure 2.8 illustrates basic triangulation from multi-view images, based on collinearity. The collinearity equations can be formulated by using the corresponding image points p and p' from different views to calculate the space coordinates of point O by triangulation. As the six EO parameters are known, the only remaining unknowns in the equations are (X, Y, Z). These coordinates can be obtained by iterating the initial approximations to determine the object space coordinates of O. The left and right cameras are denoted by their optical centres C_L and C_R, respectively.

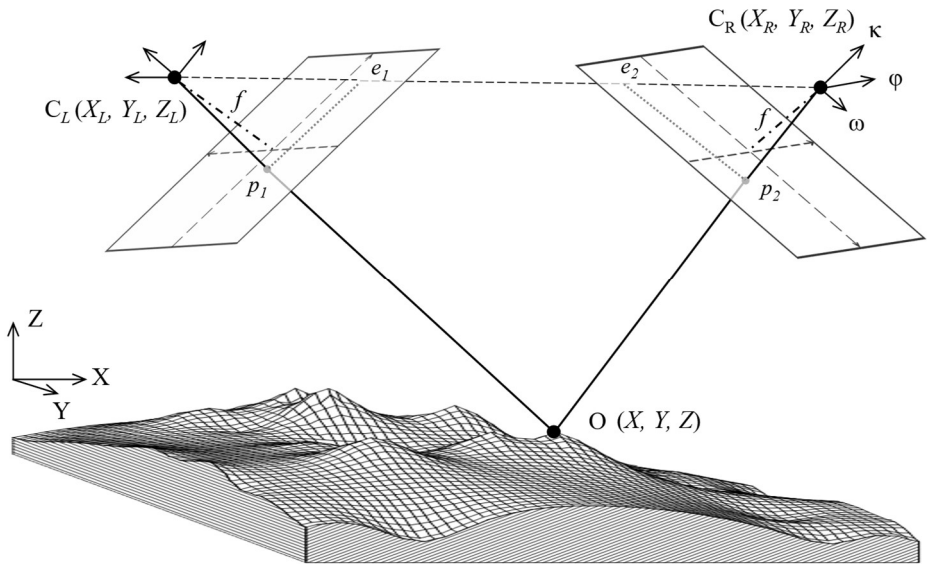


Figure 2.8 Basic geometry for multi-view image triangulation

To further refine the triangulation process, rectification can be performed to compute the transformation and rotation matrices to align the epipolar lines of the images to be parallel and horizontal. The rectification process is aimed at simplifying multi-view image matching by ensuring a consistent geometric relationship between the corresponding points in the rectified images. The transformation matrix determines the translation and scaling necessary to align the images, while the rotation matrix rotates the cameras to achieve parallel epipolar lines. By combining the IO of the cameras (which can be obtained through camera calibration), transformation matrix, and rotation matrix, the object coordinates can be calculated through rectification. The following equation succinctly expresses the relationship between the object coordinates and camera parameters:

$$\begin{aligned} s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} &= \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \\ &= K \cdot [R|T] \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \end{aligned} \tag{2.9}$$

where (x, y) denotes the image coordinates of the object; (X, Y, Z) denotes the space coordinates of the object; s is the scale factor; K is the camera matrix representing the camera IO; and R and T are the rotation and transformation matrices, respectively, which indicate the relative orientation between two cameras. Based on Eq. 2.9, given two points p and p' located on the image planes of two cameras, (Figure 2.9), the relationship between the corresponding coordinates can be expressed as follows:

$$\mathrm{s}\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} m_{00}^1 & m_{01}^1 & m_{02}^1 & m_{03}^1 \\ m_{10}^1 & m_{11}^1 & m_{12}^1 & m_{13}^1 \\ m_{20}^1 & m_{21}^1 & m_{22}^1 & m_{23}^1 \end{bmatrix} \begin{bmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \\ 1 \end{bmatrix} = M_1 \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.10}$$

$$\begin{aligned} \mathrm{s}\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} &= \begin{bmatrix} m_{00}^2 & m_{01}^2 & m_{02}^2 & m_{03}^2 \\ m_{10}^2 & m_{11}^2 & m_{12}^2 & m_{13}^2 \\ m_{20}^2 & m_{21}^2 & m_{22}^2 & m_{23}^2 \end{bmatrix} [X \quad Y \quad \mathrm{Z} \quad 1] \\ &= M_2 \cdot [\mathrm{X} \quad \mathrm{Y} \quad Z \quad 1] \end{aligned} \tag{2.11}$$

where M_1 and M_2 represent the relative orientation and transformation parameters of the two cameras, respectively. The space coordinates of the object can be solved according to the collinearity equation (Eq. 2.7) with the two given image coordinates (assuming $s = 1$).

Digital photogrammetry has revolutionised the field of 3D reconstruction by harnessing the power of computer vision algorithms. These algorithms enable the efficient computation of rotation and translation matrices by leveraging the relationships between the feature points in two images. Accurately estimating these matrices is essential for reconstructing 3D scenes from 2D images. This technological advancement has been extensively applied across various industries and research areas.

One prominent example of the application of computer vision algorithms for 3D reconstruction is in the field of remote sensing and cartography. Satellite imagery and aerial photographs can be processed using photogrammetric techniques to extract elevation data and generate accurate topographic maps (Jiménez-Jiménez et al., 2021; Pulighe and Fava, 2013). This information is crucial for urban planning, environmental monitoring, and disaster management. Another notable example of applying triangulation for 3D reconstruction is SfM, which was introduced by Longuet-Higgins (1981). SfM algorithms can reconstruct the 3D environment surrounding a vehicle, enabling better perception and understanding of the surroundings. For example, Zhang (2003) used SfM techniques to reconstruct a 3D scene from stereo camera images, facilitating accurate depth estimation and object detection for autonomous driving applications. Hu (2015) used SfM to reconstruct 3D models of ancient buildings, providing a valuable tool for preserving and visualising cultural heritage.

In human motion analyses, triangulation techniques can capture human movements with high precision. For example, Pfister et al. (2014) developed a triangulation-based approach using an infrared emitter and a depth sensor to calculate the position of human body joints. The advantage of using triangulation-based systems such as the Kinect is their ability to capture real-time motion without the requirement of attaching markers to the body of the target individual. Gait parameters, such as the stride length, joint angles, and gait symmetry, can be calculated by analysing various triangulated joint positions. In particular, this method can provide insight into the technique, balance, and

overall performance of athletes. LaViola Jr et al. (2017) proposed a triangulation-based system using multiple infrared sensors for real-time hand gesture recognition. The system used the principles of triangulation to accurately track the position of the user's hand in 3D space. By placing multiple sensors at different locations, the system could determine the hand coordinates through the triangulation of the infrared signals.

2.1.2.2 Space Resection for EO Determination

In photogrammetry, space resection is used to determine the camera EO parameters (position and orientation) through a single image based on known image coordinates of ground control points (GCPs). GCPs are reference points with known ground coordinates. The camera EO parameters are determined using the collinearity equations. GCPs that correspond to known coordinates in both the object space and corresponding image space contribute two observations to the estimation process. The EO parameters can be solved using three GCPs according to Eq. 2.7. Notably, four or more control points are typically used to achieve higher accuracy in practical applications. The introduction of a larger number of control points enables a more robust estimation of the EO parameters, typically through least-squares adjustment. By iteratively refining the parameter estimates, the least-squares adjustment minimises the discrepancies between the observed image coordinates and projected object coordinates. Figure 2.9 illustrates the geometry of space resection. With four given GCPs (O_1, O_2, O_3, O_4) and camera IO parameters, the camera EO parameters (ω, φ, k, X_C, Y_C, Z_C) can be calculated using the collinearity equation.

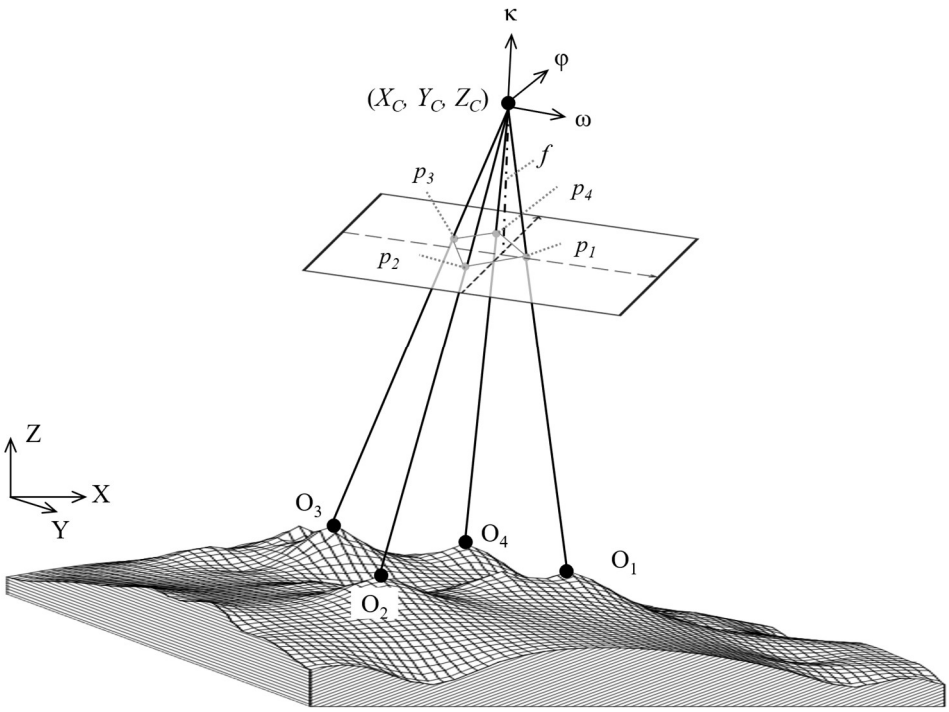


Figure 2.9 Geometry of space resection with four known GCPs

With the advent of digital imaging and computerised processing, space resection techniques have undergone significant advancements. The development of digital cameras and image sensors has enabled the acquisition of precise and high-resolution images, which can provide more reliable input data for space resection. Furthermore, the use of digital imagery has facilitated the automation of the measurement process, which has helped reduce human error and increase the speed of calculations (Tsai, 1987; Zhang, 2000). The introduction of computer algorithms and software has played a crucial role in advancing space resection. These algorithms can automatically detect and match GCPs in images, enabling the accurate estimation of camera parameters (Lowe, 2004; Szeliski, 2022). Iterative optimisation algorithms, such as the least-squares method, can refine the initial estimates and improve the accuracy of the results. Another significant development in space resection is the integration of inertial measurement units (IMUs) to measure the camera orientation and motion. By combining the measurements from these sensors with image data, space resection techniques can incorporate additional sources of information, enhancing the accuracy and reliability of camera pose estimation (Abdi et al., 2016). In recent years, deep-learning techniques have been widely used to facilitate space resection. For example, the use of CNNs for feature extraction and matching has helped enhance the efficiency

and accuracy of space resection algorithms (Kendall and Cipolla, 2016). Additionally, deep-learning-based methods have been explored for direct camera pose estimation from images, thereby eliminating the traditional feature matching and triangulation steps (Kendall et al., 2015).

2.1.3 Bundle Adjustment (BA)

BA is a classic photogrammetric technique for improving the accuracy of image orientation parameters. BA is based on the principle of collinearity, according to which a 3D point in the object space, its corresponding image point, and the perspective centre of the camera are collinear. This implies that an optical ray can be traced from the image point, through the perspective centre, to the 3D point. A bundle of optical rays can connect the images and object space by matching tie points on two or more images. Ideally, the optical rays from different images should intersect at the same 3D point, but this typically does not occur due to errors and uncertainties in the image orientation parameters. Therefore, BA aims to minimise these errors by adjusting the image orientation parameters so that the optical rays converge as close to the 3D point as possible (Wu, 2021).

The foundation of a BA system is the observation equations derived from the collinearity equations (Eq. 2.7). As the collinearity equations are nonlinear, they are linearised by applying the first-order terms of Taylor's series. The following expressions present the four types of observations formulated in a BA system, based on the least-squares principle:

$$\begin{aligned} Av + B\Delta &= f \\ v_x - I\Delta &= f_x \\ A_c v_c + C\Delta_c &= f_c \\ A_{\text{ap}} v_{\text{ap}} + D\Delta_{\text{ap}} &= f_{\text{ap}} \end{aligned} \tag{2.12}$$

where A is the matrix of observation coefficients, B is the matrix of parameter coefficients, Δ is a vector containing the unknown EO parameters, and v is the vector of residuals. The first observation equation relates to the matching of tie points. These measurements are connected to their corresponding 3D coordinates via collinearity

equations. By establishing this connection, the first observation equation permits the incorporation of image measurements into the framework for BA. The second observation equation focuses on the unknown EO parameters and 3D object coordinates of the tie points. This equation regulates the estimation and optimisation of these unknown variables, enabling the exact determination of the EO parameters and 3D coordinates of the tie points. The third observation equation imposes constraints on the BA parameters, which provide additional information that can help enhance the solution precision. By incorporating these constraints into the third observation equation, the BA system can ensure adherence to these conditions. The fourth observation equation facilitates self-calibration by allowing additional camera IO parameters to be simultaneously solved within the BA framework. This equation permits the estimation of additional IO parameters, enabling the refinement of the camera calibration result.

The basic idea behind BA is to minimise the reprojection error, which indicates the discrepancy between the observed image projections of 3D points and their corresponding predicted projections based on the estimated camera poses and parameters. BA iteratively refines the camera poses and 3D points by minimising this error until an optimal solution is reached. This approach has been widely used in various research fields, such as photogrammetry, remote sensing, and computer vision.

Triggs et al. (2000) reported one of the seminal works on BA, providing a comprehensive overview of BA algorithms, covering various optimisation techniques, robust estimation methods, and strategies for addressing large-scale problems. Researchers have actively sought to develop efficient BA algorithms. Several optimisation frameworks, such as Levenberg–Marquardt, Gauss–Newton, and sparse matrix factorisation techniques, have been explored to solve the nonlinear optimisation problem involved in BA (Bernecker and Idini, 2022; Chen et al., 2019; Lourakis and Argyros, 2005). In recent years, BA has been advanced by incorporating additional priors or constraints. For example, BA with priors, such as temporal or geometric constraints, has been implemented to enhance the accuracy and efficiency of 3D reconstruction and SfM systems (Sibley et al., 2019; Wei et al., 2020).

2.1.4 Dense Image Matching

Dense image matching is a fundamental task in many domains, such as photogrammetry, computer vision, and image analysis. The objective is to extract dense point clouds from multiple images with known orientation parameters. At present, image-based surveying and 3D modelling techniques can deliver point clouds with accuracies comparable to those produced by laser scanning (Remondino et al., 2014) for many terrestrial and aerial applications in a reasonable time. Owing to the inherent nature of multi-spectral images, rich textural information can be extracted. Moreover, the accuracy of a point cloud can be assessed based on the redundant measurements extracted from imagery. Image-based 3D reconstruction is widely applied for 3D modelling, mapping, robotics, and navigation due to its lightweight nature, convenience, cost-effectiveness, and ability to generate textured point clouds comparable to those obtained using LiDAR systems (Szeliski, 2022).

Traditional dense image matching algorithms are further discussed in the following section. With advancements in camera technologies and the advent of innovative matching approaches, many state-of-the-art image-based algorithms and software for 3D modelling and reconstruction have been developed, as discussed in the subsequent sections.

2.1.4.1 Traditional Dense Image Matching Methods

Photogrammetry has played a significant role in the development of image matching algorithms, especially those focused on aerial images. Early matching algorithms were developed by the photogrammetry community in the 1950s (Hobrough, 1959). With the significant progress made in computer vision algorithms over the years, this task has been transformed into the stereo vision problem (Trucco and Verri, 1998). Stereo vision techniques aim to produce a depth map in the image space. The disparity measure, which represents the horizontal motion between corresponding image points, is inversely proportional to the distance between the camera and object. Figure 2.10 illustrates the fundamental geometry of stereo vision, which involves a pair of cameras positioned at a baseline distance from each other.

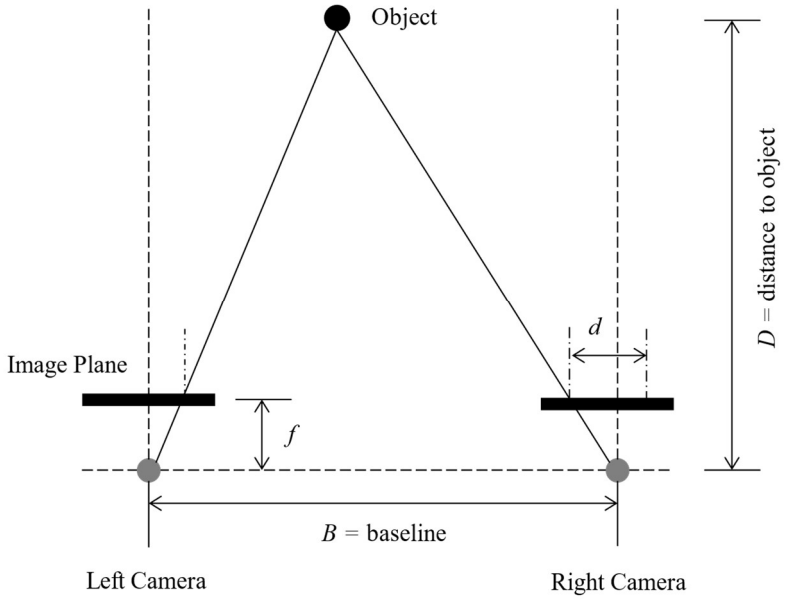


Figure 2.10 Basic geometry of stereo vision

This configuration mimics the human eye perception and can capture the depth information of a scene. The stereo camera system provides two slightly different views, enabling the computation of the distance D between the cameras and a target object using Eq. 2.13:

$$D = \frac{B \cdot f}{d} \tag{2.13}$$

where D represents the distance between the cameras and target object, B denotes the baseline distance separating the cameras, f is the focal length of the camera, and d is the disparity value obtained from the positional discrepancy of the target object in the stereo image pair. Various cost-matching metrics have been used to measure the similarity between pixels and determine their disparities in an accurate and reliable manner.

Traditional methods comprise the early algorithms based on basic cost-matching metrics, such as the sum of absolute differences (SAD), normalised cross-correlation (NCC), mutual information (MI), and CT. These metrics have been used to measure the similarity between corresponding pixels in stereo images, enabling the estimation of depth information.

The SAD metric determines the dissimilarity between corresponding pixels in two images by calculating the absolute differences in their intensity values and summing them. This metric assumes that corresponding pixels have similar intensity values in both images. For example, the block matching algorithm can use the SAD to divide the reference image into small blocks and identify the best matching block in the target image. The disparity between the reference and target blocks is estimated based on the block with the lowest SAD (Lu and Liou, 1997). SGM is another widely used algorithm for dense stereo matching that incorporates SAD. The SGM uses dynamic programming techniques to determine the optimal disparity values that minimise the overall cost, considering local matching costs and the consistency of disparity values along different scanline directions (Hirschmuller, 2005). The adaptive support-weight approach (ASW) also uses the SAD metric for dense image matching. ASW incorporates adaptive support weights that dynamically adjust the influence of neighbouring pixels based on their SAD values. This scheme helps alleviate the effect of noisy or unreliable matches, leading to improved accuracy in dense matching results (Yang et al., 2008).

NCC indicates the similarity between two images by normalising their cross-correlation coefficients. Specifically, this metric measures the degree of linear dependency between corresponding image patches. For example, template matching algorithms use the NCC as the matching metric to locate a template image within a larger image. By sliding the template over the larger image and calculating the NCC score at each location, the algorithm determines the site at which the template best matches the image, enabling object detection and localisation (Briechle and Hanebeck, 2001; Viola and Jones, 2001). The iterative closest point (ICP) algorithm is commonly used for point cloud registration and 3D surface reconstruction. In each iteration, ICP uses the NCC to establish correspondences between the points in the reference and target point clouds. By maximising the NCC score, ICP iteratively refines the transformation parameters to align the two point clouds (Besl and McKay, 1992).

MI is a statistical metric that measures the amount of information shared between two images. This metric estimates the statistical dependency between the intensity values of corresponding pixels based on their joint histogram. Studholme et al. (1999)

proposed the normalised MI (NMI) approach for medical image registration and multi-modal image alignment. NMI measures the statistical dependency between the intensities of corresponding pixels in the images. By maximising the NMI score, the algorithm determines the optimal transformation parameters that align the images. Hirschmuller (2007) combined MI and block matching techniques for dense stereo matching. This approach divided images into blocks and searched for the best matching block in the target image, based on the MI score. The disparities between corresponding pixels in stereo images were estimated by maximising the MI. The optimal disparities were determined by comparing the statistical dependencies between the intensities of the blocks. MI also has been used as a matching criterion in the optical flow method. Roth and Black (2007) proposed an MI-based optical flow approach for motion estimation and video analysis tasks. This strategy calculates the MI between pixel intensities in neighbouring frames and estimates the displacement by maximising the MI score.

The CT metric encodes the spatial arrangement of pixel intensities in a binary code. This strategy compares the census codes of corresponding pixels to compute the dissimilarity between images. The CT, which can effectively manage photometric variations and occlusions, is widely used in stereo-matching applications. The SGM algorithm, proposed by Hirschmuller (2007), uses the CT for dense stereo matching. Specifically, this algorithm uses the CT to compute matching costs between pixels in stereo images. By comparing the CT codes, SGM estimates the disparities by minimising a global energy function. The CT helps capture the local image structure, thereby improving the robustness of the matching process. Another popular CT method for dense stereo matching is the non-parametric local transform, proposed by Zabih and Woodfill (1994). This method applies the CT to image patches to encode the local neighbourhood structure. The CT converts the pixel intensities in a patch to binary codes, thereby capturing the ordinal relationships between the pixels. The disparities between stereo images can be estimated by comparing the CT codes between corresponding patches. Adaptive CT (ACT), developed by Perri et al. (2013), is another algorithm that uses CT for stereo dense image matching. The objective is to improve the robustness and efficiency of image transformation on an FPGA chip. This strategy computes the CT by weighting the contributions of neighbouring pixels based on their

similarity to the centre pixel. By adapting the window size according to the local image structures, ACT can achieve more accurate and reliable dense matching results than CT.

These traditional stereo dense image matching methods have been extensively used for tasks such as image registration, image alignment, and depth estimation. In particular, these techniques played a crucial role in early computer vision applications, providing a foundation for subsequent research and developments in the field.

2.1.4.2 Deep-Learning Dense Matching Methods

With advancements in deep-learning approaches, the cost-matching pipeline has been replaced by CNNs. Deep-learning-based algorithms have attracted significant attention owing to their excellent performance in benchmark testing. Depending on the learning task, deep-learning stereo methods can be classified into learning-based cost metrics and end-to-end (E2E) learning approaches.

The application of learning-based cost metrics was pioneered by Krizhevsky et al. (2017), who highlighted the transformative impact of deep learning on stereo vision applications in terms of enhanced performance. Ciregan et al. (2012) discussed the limitations of traditional stereo-vision methods compared with human performance in recognition tasks and argued that deep-learning algorithms can bridge this gap through their ability to emulate human-like recognition. This observation has motivated researchers to integrate deep-learning techniques into stereo vision algorithms to achieve human-level performance.

The advent of machine learning has had a profound impact on stereo vision based research, as noted by Tonioni et al. (2017). Advancements in machine learning techniques have driven relevant research and provided valuable opportunities for algorithm refinement and real-world applications. In the context of image classification, Chauhan et al. (2019) used CNNs for vehicle counting and classification in the transport engineering domain. This framework could accurately classify different types of vehicles, demonstrating the potential of practically applying deep learning in stereo vision for addressing complex tasks in real-world scenarios.

Deep-learning methods have rapidly evolved into E2E learning algorithms, replacing classical multistage optimisation with trainable networks that can directly predict disparity from stereo images. These neural networks can capture more global features, potentially improving the task performance. Kang et al. (2019) advanced stereo vision techniques by introducing dilated convolution into their E2E network architecture. The authors emphasised the computational advantages of dilated convolution over 3D CNN methods, which can lead to improved efficiency. Evaluation over the KITTI dataset demonstrated significant enhancements compared with the original DispNet implementation. The integration of dilated convolution offers a promising avenue for advancing stereo vision systems, as it can yield more accurate depth estimation and address the challenges associated with texture-less areas. Future research can be aimed at refining the application of dilated convolution in stereo vision algorithms.

Yang et al. (2019) introduced HSMNet, an E2E network architecture designed for stereo matching. This network has an encoder–decoder structure incorporating a coarse-to-fine hierarchy for stereo matching. To extract multi-scale features, a downsampling mechanism progressively reduces the input resolution. The pyramid feature module incorporates residual blocks and spatial pyramid pooling layers to enhance the receptive fields, thereby facilitating hierarchical matching. The authors emphasised the real-time computational efficiency of their network, which could enable on-demand computation. The network could estimate large disparity objects before the end of the pipeline, resulting in improved efficiency. This approach outperformed other E2E networks, such as those proposed by Chang and Chen (2018), Kendall et al. (2017), and Song et al. (2019), when evaluated on the Middlebury and KITTI datasets.

Unlike the abovementioned methods, certain techniques integrate context learning into specific components of the conventional pipeline without entirely aligning with any of the mentioned deep-learning paradigms. One such example is SGM-Net (Seki and Pollefeys, 2017), which focuses on learning the smoothness penalty on a per-pixel basis. Similarly, GA-Net (Zhang et al., 2019) trains networks to guide the cost-aggregation process. These approaches deviate from the purely deep-learning-based methods and instead enhance specific stages within the traditional pipeline by leveraging context learning techniques. SGM-Net, for instance, focuses on refining the estimation of the

per-pixel smoothness penalty, resulting in more accurate disparity maps. GA-Net optimises the cost-aggregation process using learned networks, thereby enhancing the disparity estimation.

2.2 Visual Odometry

Localisation is an essential task for autonomous vehicles to be able to track their paths and effectively detect and avoid obstacles. Vision-based odometry is a robust technique for vehicle localisation. The concept of estimating the pose of a vehicle solely from visual input was first introduced by Moravec in the early 1980s (Nistér, 2004; Scaramuzza and Fraundorfer, 2011). From 1980 to 2000, research on VO was driven by NASA's preparations for the 2004 Mars mission. The term "visual odometry" was coined to describe the process of incrementally estimating vehicle motion by integrating pixel displacements between image frames, similar to how wheel odometry estimates motion by integrating the number of wheel turns over time (Scaramuzza and Fraundorfer, 2011).

VO is a pose estimation process commonly implemented by various agents, such as vehicles, humans, and robots. VO frameworks use continuous images from one or multiple attached cameras (Fraundorfer and Scaramuzza, 2011). At the core of VO lies camera pose estimation, which involves determining the agent's relative motion based on the visual input (Ni and Dellaert, 2006). This online estimation of ego motion from video input is an effective non-contact method for effectively positioning mobile robots (Munguia and Grau, 2007). By analysing image sequences captured by a camera, VO enables incremental online estimation of the vehicle position (Campbell et al., 2005; Gonzalez et al., 2012).

Images contain rich and informative data that can be leveraged to estimate camera movement, and thus, VO is a viable solution for motion estimation (Rone and Ben-Tzvi, 2013). Unlike wheel encoders and low-precision inertial navigation systems (INSs), VO is less prone to local drift, resulting in more accurate motion estimation (Howard, 2008). VO is particularly advantageous in scenarios involving uneven terrains, in which wheel slippage may occur, or GPS-denied environments (Scaramuzza and Fraundorfer,

2011). VO can be combined with GPS and INS measurements to maximise the accuracy. One of the distinct advantages of VO over laser and sonar localisation systems is its non-invasive nature, as it does not emit detectable energy into the environment. Unlike GPS, VO does not rely on external signals for operation, thus providing greater flexibility in various environments (Ni and Dellaert, 2006). The use of cameras for robot localisation offers several benefits, such as cost reduction, seamless integration with other vision-based algorithms (e.g., obstacle, pedestrian, and lane detection), and elimination of sensor calibration requirements (Wang et al., 2011). Cameras are compact, affordable, lightweight, energy-efficient, and adaptable, making them suitable for a wide range of vehicles (land, underwater, or air) and other robotic applications (such as object detection and recognition).

In scenarios in which the distance between the stereo camera and scene exceeds the stereo baseline, the effectiveness of stereo VO diminishes, and it becomes analogous to the monocular case, which relies on 2D bearing data to estimate both the relative motion and 3D structure (Scaramuzza and Fraundorfer, 2011). Nistér (2004) presented a real-time VO algorithm capable of estimating camera motion from a monocular or stereo camera. The authors introduced the first real-time large-scale VO specifically designed for monocular cameras. This approach used feature tracking and incorporated random sample consensus for robust outlier rejection. The algorithm consisted of three phases: feature detection, feature tracking, and motion estimation. Although the overall framework remained the same for monocular and stereo vision systems, slight differences existed in the motion estimation phase. In the monocular case, a five-point pose algorithm was used to calculate the pose for each tracked feature. The 3D position of each feature was then computed using the first and last acquired images, which facilitated the estimation of the camera's 3D pose. In the stereo case, the 3D position of each feature was obtained through stereo matching of corresponding features between the two camera images.

VO can be considered a subset of SLAM: VO focuses solely on estimating the camera or robot motion without explicitly building a map of the environment, whereas SLAM is aimed at mapping the environment while simultaneously estimating the camera or robot motion (Souici et al., 2013). Visual SLAM leverages camera sensors to gather observation data for map creation. In feature-based SLAM, the robot uses

environmental features to update its position by extracting and reobserving these features as it moves. Notably, real-time algorithms such as large-scale direct monocular (LSD)-SLAM (Engel et al., 2014) and ORB-SLAM (Mur-Artal et al., 2015) have been developed for SLAM using a freely moving monocular camera. ORB-SLAM is a feature-based method that excels in challenging scenarios with significant motion clutter and enables robust loop closing and re-localisation. In contrast, LSD-SLAM takes a direct approach that avoids the need for feature extraction, rendering it suitable for generating semi-dense reconstructions in low-texture environments and resistant to blur. This approach achieves localisation by directly optimising image pixel intensities.

Gonzalez et al. (2012) and Yu et al. (2011) highlighted that the primary limitations associated with VO systems are the high computational expense and vulnerability to variations in light and imaging conditions. These conditions encompass factors such as direct sunlight, shadows, image blur, and disparities in image scale. In regions in which the floor exhibits a smooth and low-textured surface, the directional sunlight and lighting conditions may result in non-uniform scene lighting. Additionally, the presence of shadows caused by stationary or moving objects, include those originating from the vehicle, can impede the accurate calculation of pixel displacement, leading to errors in displacement estimation (Gonzalez et al., 2012). Monocular vision systems encounter challenges related to scale uncertainty (Kitt et al., 2011; Zhang et al., 2014b). When the surface is uneven, the image scale tends to fluctuate, which makes it difficult to estimate the image scaling factor. Kitt et al. (2011) suggested that scaling factor may be inaccurately estimated in scenarios involving a significant change in the road slope, resulting in erroneous estimation of the trajectory.

2.3 Parallel Architecture for 3D Applications

Parallel architectures are computer systems that use multiple processing units or cores to simultaneously perform tasks. Instead of relying on a single processor to manage all computations, parallel architectures divide the workload into smaller tasks that can be executed in parallel, thereby improving computational efficiency and reducing the processing time.

Parallel architectures play a crucial role in enhancing the overall performance and capabilities of photogrammetric systems. Photogrammetry involves computationally intensive tasks, such as feature extraction, matching, triangulation, point cloud generation, mesh construction, and texturing. Using parallel architecture, these tasks can be distributed among multiple processing units, such as CPU cores or GPUs, allowing their simultaneous processing. Such parallel processing techniques can accelerate the overall reconstruction process, enabling faster generation of 3D models from the input data (Fu et al., 2023; Wiechert et al., 2012). Additionally, parallel architectures allow photogrammetric applications to efficiently handle large-scale reconstructions and process massive datasets. By dividing the workload among multiple processing units, the system can effectively use the available computational resources and scale with the complexity and size of the data. This scalability is particularly beneficial in applications that require the processing of many images or generation of high-resolution 3D models (Buttinger-Kreuzhuber et al., 2022; Pepe and Prezioso, 2016). Parallel architectures, especially those with GPU acceleration, can achieve real-time or near-real-time performance in 3D photogrammetric applications. The parallel processing capabilities of GPUs, specifically designed for handling large amounts of data simultaneously, enable rapid execution of computationally demanding tasks such as feature matching and triangulation (Wang, 2019). Furthermore, parallel architectures can optimise the available hardware resources, such as CPU cores or GPU threads. Instead of leaving processing units idle during certain stages of the photogrammetric pipeline, parallel architectures can ensure that the available resources are efficiently utilised, leading to better overall system performance and reduced processing time (Choudhary et al., 2012).

2.3.1 Multi-threading on CPU

Multi-threaded technologies have emerged as a promising approach for accelerating computationally intensive algorithms in various fields, including photogrammetry. Researchers have recognised the potential of multi-threading in leveraging CPU resources, leading to improved algorithm performance. Shigeto and Sakai (2011) generated a DEM from input images by using CPU multi-threading acceleration. The authors used two Intel Xeon W5590 dual-core CPUs to implement their method, offering 240 clocks with an impressive 61,440 threads dedicated to image processing.

Vladimir (2016) proposed a multi-threaded approach for dense point cloud generation from stereo images. The proposed method accelerated the point cloud generation process by leveraging parallel processing capabilities. The authors implemented their algorithm on a multi-core CPU architecture, effectively distributing the computational load across multiple threads. This parallelisation resulted in accelerated processing, enabling efficient and rapid reconstruction of dense 3D point clouds [3].

Grazioso et al. (2019) developed a photogrammetric system with 3D body scanners for health-related applications. The objective was to generate a comprehensive 3D body model from data acquired by the 3D scanners. To expedite data processing, the researchers implemented a multi-threaded strategy. However, the system failed to achieve real-time 3D human body model reconstruction despite the use of multi-threading acceleration. The failed realisation of real-time performance was attributable to the inherent complexity of the image processing algorithm and substantial volume of data involved.

The utilisation of multi-threading in CPUs used in photogrammetry research can help enhance the computational performance and expedite data processing. Photogrammetric algorithms can distribute computation across multiple threads or cores by exploiting the parallel processing capabilities of multi-threading, resulting in increased efficiency and reduced processing time. The benefits of multi-threading become particularly pronounced in large-scale data processing scenarios, where the

parallelisation of tasks can effectively harness the computational power of modern CPUs.

While multi-threading technology can accelerate photogrammetric applications, it still exhibits several limitations. Multi-threading relies on the availability of tasks that can be executed concurrently. In photogrammetry, specific tasks may have dependencies that may also be sequential, limiting the potential for parallelisation (Schiele et al., 2012). Although multi-threading allows for efficient utilisation of CPU resources, its scalability can be constrained by factors such as the memory bandwidth, cache coherence, and thread synchronisation overhead. As the number of threads increases, these factors can limit the performance gains achieved from parallel execution, leading to diminished returns (Yang and Zhang, 2015). Therefore, depending on the nature of the algorithm and available computational resources, alternative approaches, such as GPU acceleration or distributed computing, may need to be incorporated to overcome the limitations of multi-threading and enhance the performance of photogrammetric techniques.

2.3.2 GPU Acceleration

In recent years, the development and evolution of GPU technologies have revolutionised image processing capabilities, providing new opportunities for real-time processing in photogrammetry applications. Dense matching algorithms have traditionally relied on the CPU for feature point extraction and image processing. However, by using powerful GPUs, the processing time and data volume can be increased. Hardware-oriented approaches have harnessed the computational power of modern graphics machines to achieve enhanced performance in photogrammetry tasks. For instance, Zach et al. (2004) presented a hierarchical disparity estimation algorithm implemented on a programmable 3D GPU. This method, capable of processing rectified or uncalibrated image pairs, employed bidirectional matching in combination with a locally aggregated sum of absolute intensity differences. Implementation over an ATI Radeon 9700 Pro framework led to an impressive processing rate of up to 50 frames per second (fps) for 256×256-pixel input images.

In addition to these algorithms, another notable example of real-time dense image matching with GPU acceleration is the SGM algorithm, as illustrated by Hernandez-Juarez et al. (2016). The SGM algorithm minimises a global energy function consisting of data and a smoothness term. To achieve real-time operation, the authors leveraged the parallelism offered by GPUs. They implemented a hierarchical belief propagation method that optimised the smoothness term iteratively while removing redundant computations to ensure fast convergence. Experimental results demonstrated the effectiveness of this approach: a processing speed of 42 fps was achieved for self-recorded images with dimensions of 640×480 pixels and 128 disparity levels. The experiments were conducted using an NVIDIA Tegra X1 graphics card, with four path directions used for the SGM.

Kern et al. (2020) also used GPUs for photogrammetry by developing a GPU-accelerated method for real-time 3D reconstruction using UAVs. This approach leveraged the parallel computing power of GPUs to enable rapid image feature extraction, dense matching, and 3D reconstruction. Experimental results indicated the achievement of real-time performance, enabling on-the-fly reconstruction during UAV flights. Maoteng et al. (2017) proposed a GPU-accelerated BA algorithm for large-scale reconstruction scenarios. By harnessing the parallel processing capabilities of GPUs, significant improvements in computational efficiency were achieved. The GPU implementation demonstrated a 20-fold reduction in processing time compared with CPU-based approaches, enabling faster and more efficient large-scale reconstructions.

The integration of dense matching algorithms with GPU acceleration has significantly improved the efficiency and performance of image processing in real-time applications. By harnessing the parallel processing capabilities of GPUs, these algorithms can handle larger data volumes with reduced processing times. Real-time processing is especially important in various applications, such as robotics, augmented reality, and autonomous systems, where immediate feedback and timely decision-making are critical. Further exploration of GPU-based algorithms for dense matching can lead to advancements in real-time image processing in photogrammetry. The ever-evolving GPU technology, coupled with algorithmic optimisations and hardware innovations, is expected to further advance dense matching techniques, enabling faster and more accurate reconstruction of 3D models from images.

2.4 Real-Time Aerial Mapping

UAVs have revolutionised mapping technologies and serve as rapid and cost-effective solutions for capturing aerial imagery and generating accurate maps. The interdisciplinary nature of UAV data collection and the wide range of applications make them invaluable for various scientific investigations. Mapping is one of the primary applications for UAVs. By combining images captured by onboard cameras, UAVs can generate accurate maps using photogrammetric techniques. These techniques involve extracting features from images and matching them to create orthoimages and DSMs.

Traditional photogrammetric techniques can effectively generate precise and accurate reconstructions. However, their application is limited by their computational complexity and inability to incorporate incremental updates. By requiring the simultaneous input of all data, such methods impose time constraints that may hinder real-time applications or implementation in scenarios in which data collection occurs over an extended period. For example, Pix4D (Pix4D, 2017) and Agisoft PhotoScan (Agisoft, 2014) are commercial photogrammetry applications well known for their precise reconstructions. These frameworks use photogrammetric techniques to process digital images and generate 3D spatial data, including dense point clouds and texturised polygonal models. Furthermore, they offer parallel computing and distributed processing capabilities to optimise the execution time. However, the processing time may still be significant for large datasets or complex scenes (Barbasiewicz et al., 2018). COLMAP (Schönberger et al., 2016) is a free and open-source software for photogrammetric reconstruction, which uses optimised algorithms for accurate reconstructions. However, the acquisition of precise results often requires user expertise and careful parameter tuning, particularly in challenging scenarios. These steps necessitate additional learning for the user and time for data processing (Schönberger et al., 2016). In addition, this software lacks incremental update capabilities, which limits its adaptability to dynamic environments or situations in which new data becomes available over time. Overall, the processing time and inflexibility of traditional photogrammetry techniques hinder their use in practical mapping applications.

To overcome the limitations of real-time processing in traditional photogrammetry approaches, fast mapping solutions based on image mosaic techniques can be applied, given their capacity for incremental mapping. Bu et al. (2016) proposed a notable fast image stitching approach. Their open-source framework, Map2DFusion, replaces the traditional picture alignment module in a stitching pipeline with a cutting-edge SLAM algorithm. Map2DFusion, which is a mature and well-studied framework, can alleviate the challenges associated with loop closing, global optimisation, and robust tracking in visually challenging environments. This framework uses 3D camera posture information to build 2D maps. Figure 2.11 shows the process flow for using Map2DFusion to create precise and dependable 2D maps.

First, the input images are subjected to distortion removal, and features are retrieved. Visual SLAM is used to find unique keyframes in the image sequence. Local optimisation and loop detection are performed to refine the camera pose estimates. Each image is accompanied by synchronised GNSS measurements to establish a geographic reference and transmit the camera posture. The framework computes a 2D best-fitting plane using the 3D triangulated sparse cloud of the scene. This plane is used to project and align pinhole-camera-modelled pictures to create a worldwide mosaic. All the pictures are used to rapidly build maps post-flight.

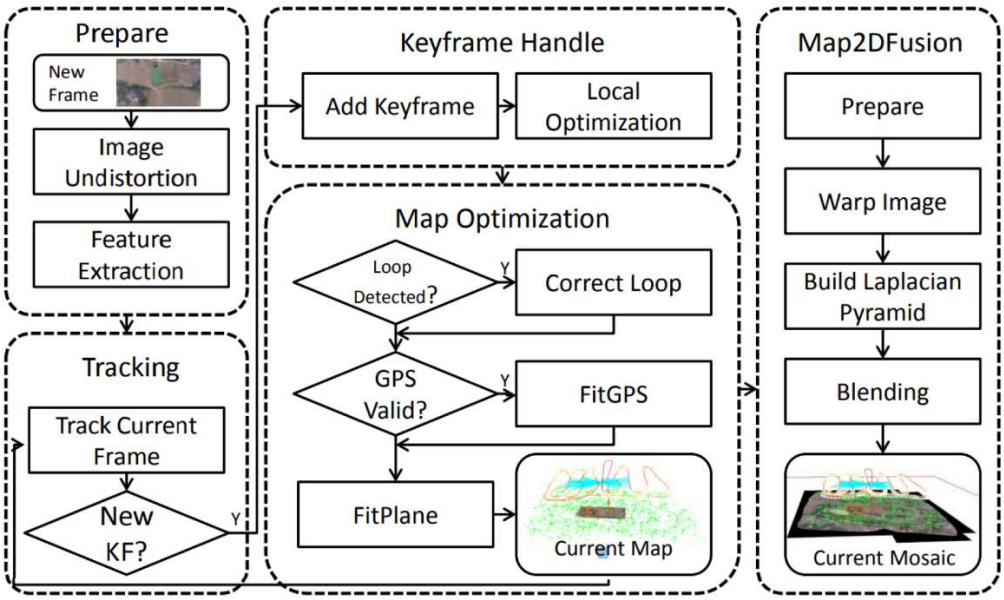


Figure 2.11 Framework of Map2DFusion (Bu et al., 2016).

Hinzmann et al. (2018) performed extensive work on real-time aerial mapping. Unfortunately, during the research period of this thesis, certain modules of their framework remained closed source, limiting the possibility of conducting an extensive practical survey. Nonetheless, their theoretical work inspired the fundamental algorithms implemented in Chapter 5 of this thesis. Figure 2.12 shows an overview of their system.

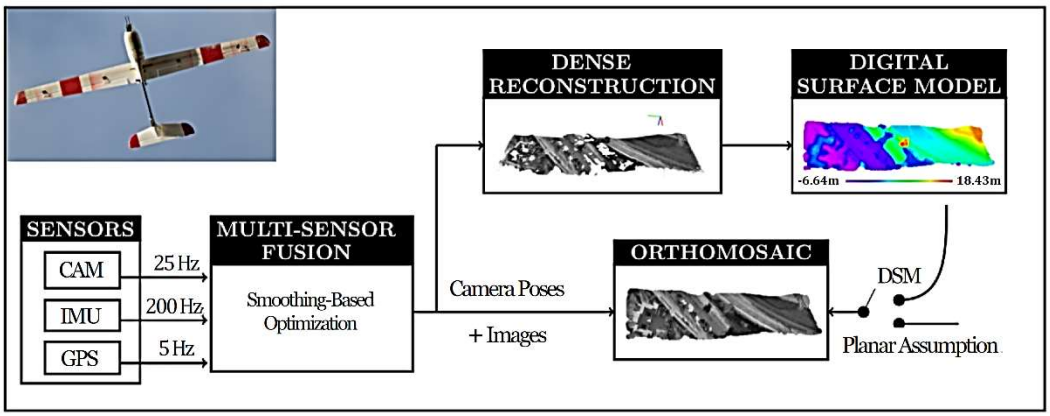


Figure 2.12 Overview of aerial mapper system proposed by Hinzmann et al. (2018).

Hinzmann et al. (2018) estimated camera postures without visual SLAM, unlike Bu et al. (2016). The authors used a KandeLucas-Tomasi feature tracker, IMUs, and a GNSS module to continuously estimate the state. A dense point cloud of the observed scene was reconstructed by merging data from various sensors to generate a 3D camera posture. This point cloud was incorporated into a multi-layer grid map to create a 3D DEM of the surface. Using this model, the authors generated an orthogonal mosaic to comprehensively visualise the mapped area.

The method proposed by Hinzmann et al. (2018) closely resembles traditional offline aerial photogrammetry. The camera poses are determined as the surface is identified as a 3-space quantity, and the resulting map is rectified. In this manner, the geometric distortion of the surface is considered and corrected. Unlike Map2DFusion, the aerial mapper was designed to perform calculations on UAVs. Although this information can be used for navigation, it is of limited value to the user. Global maps on ground stations (GCS) are extremely valuable and should be transmitted as soon as possible. In addition, as part of the overall mapping procedure, certain tasks can be divided between the UAV

and GCS for ground station validation. This approach is particularly suitable for tasks that are computationally intensive and may require GPU acceleration. This strategy can help achieve higher map resolution and is expected to have more significant real-world use cases in the future.

2.5 Summary

Chapter 2 provides a comprehensive review of research on photogrammetry, focusing on the fundamentals of photogrammetry, monocular VO, use of parallel architectures in 3D photogrammetric applications, and real-time applications associated with aerial mapping.

The section on the fundamentals of photogrammetry discusses various techniques for feature detection and matching. Traditional methods such as SIFT, SURF, and ORB are explored alongside deep-learning methods, such as SIFT-CNN, SuperPoint, SuperGlue, and LF-Net. Additionally, the chapter covers dense image matching, distinguishing between stereo dense image matching algorithms and deep-learning-based stereo-matching approaches. Furthermore, the concepts of triangulation and space resection are introduced, encompassing topics such as 3D reconstruction and EO determination. The following section delves into monocular VO, which involves estimating camera poses using state-of-the-art algorithms. This section highlights the importance of accurate camera pose estimation for visual navigation and positioning applications. The last section describes the use of parallel architectures in 3D photogrammetric applications, including multi-threading on CPUs and GPU acceleration. Notably, multi-threading enables the efficient utilisation of CPU resources, while GPU acceleration leverages the parallel computing power of GPUs to accelerate computationally intensive tasks in photogrammetry.

This chapter highlights the advancements and current trends in photogrammetry. However, several areas warrant further investigation. Specifically, the potential of combining traditional and deep-learning-based methods for feature detection and matching, dense image matching, and 3D reconstruction must be explored. Hybrid approaches can potentially leverage the strengths of both techniques to improve

accuracy and efficiency. Additionally, future research can focus on developing robust and efficient algorithms for monocular VO that can operate in real-time, facilitating autonomous navigation and robotics applications.

Chapter 3 Real-Time Cross-View Feature Matching and Camera Pose Determination

In recent years, robotic vision has become increasingly popular owing to its versatility and applications in various fields, such as industrial inspection, remote sensing for mapping and surveying, and rescue operations. Despite these capabilities, the autonomous navigation capacity of robots remains limited, particularly in GPS-denied environments where GPS signals are unavailable or unreliable. While GPS technology has revolutionised location-based services, it has limitations in some environments. GPS signals can be blocked by buildings, trees, or other obstacles, and their accuracy can be affected by atmospheric conditions. Implementing computer vision technology on robots is a potential solution to improve the navigation capacity of robots in unfavourable environments.

3.1 Overview of Approach

This chapter presents a novel approach for visual-based camera-pose determination of aerial robots (e.g., UAVs and drones). This method offers a cost-effective alternative to traditional navigation methods, such as those relying on GPS, inertial measurement units, and laser or radar sensors. Additionally, the method provides a flexible solution for use in the environments where GPS signals are interfered with or blocked. The proposed method is a coarse-to-fine approach that localises the robot by two sequential processes: a) feature-based cross-view image matching and retrieval for matching the aerial images with a pre-built database constructed from a large-scale orthoimage base map; b) camera relative and absolute pose determination based on the integration of VO and space resection. The first process narrows the down the region for visual positioning, while the second process (i.e., space resection) identifies the exact location and orientation of the aerial robot through VO and photogrammetry techniques.

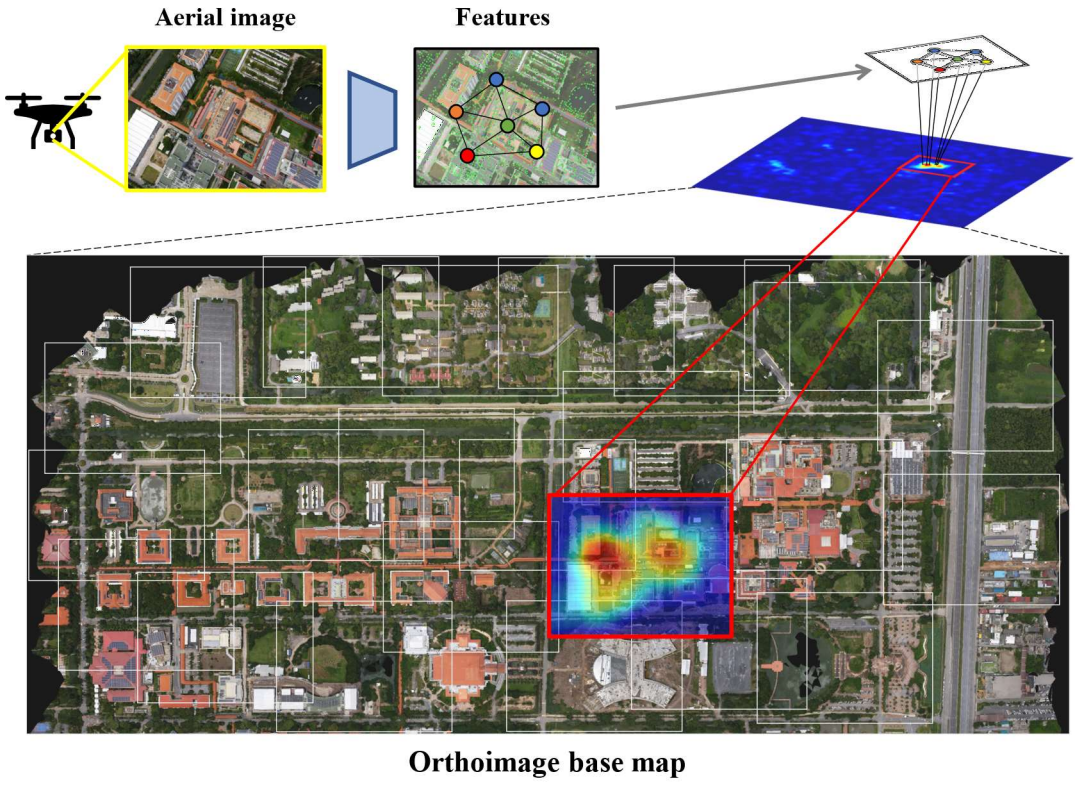


Figure 3.1 Overview of the feature-based cross-view image matching and retrieval for camera pose determination. The similarity between the local feature from the aerial image and the global feature from the orthoimage base map enables efficient recall and matching of cropped orthoimage tiles from a pre-built database.

The concept of the approach (Figure 3.1) is introduced in the following sections:

(1) In Section 3.2, a feature-based approach for cross-view image matching and retrieval is presented. The pre-built database consists of cropped orthoimage tiles and a DSM with features extracted using a deep learning-based algorithm. These features are saved in separate files as the global feature points and descriptions for feature image retrieval and matching.

(2) In Section 3.3, an integrated VO method is proposed to determine the camera position and orientation. Our approach consist of state-of-the-art deep learning algorithms for VO and space resection to obtain the absolute pose of the camera in real-world coordinates. The details of our approach are presented in the subsections.

(3) In Section 3.4, the experiment and evaluation are presented, and the effectiveness of our method is demonstrated using open-access image data. The trajectory of the estimated camera pose and the evaluation results of the accuracy and efficiency of the method are presented.

3.2 Feature-Based Cross-View Image Matching and Retrieval

Image matching and retrieval are crucial in several applications in which images are matched with a vast reference image database, such as robotic vision, navigation, and positioning (Arras et al., 1998, Chen et al., 2016, Deng et al., 2012). Because these tasks can be computationally intensive, particularly in cases involving large datasets, implementing an effective feature extraction strategy is essential to alleviate the computational burden. This strategy involves selecting the most informative features from the images, which reduces data dimensionality, computational costs, and the risk of overfitting. Furthermore, effective feature extraction enables the extraction of critical discriminative information from images, enhancing recognition and retrieval accuracy. For example, in place recognition applications, distinctive features such as unique buildings or landmarks can be captured, facilitating accurate recognition even in the presence of other similar-looking places. Images of the same or similar places may significantly vary depending on lighting conditions, viewpoint directions, and occlusions. Effective feature extraction captures the key characteristics of an image that remain invariant to these variations, providing robustness to image variations and enabling reliable place recognition and retrieval. Another advantage of feature extraction is its scalability to large image datasets, making it suitable for real-world applications such as online image retrieval. However, selecting the appropriate feature extraction strategy depends on several factors, including the specific application and characteristics of the adopted dataset.

3.2.1 Feature Extraction Methods and Evaluation

Feature extraction is a crucial aspect of computer vision and photogrammetry, as it forms the foundation for essential applications such as VO and simultaneous localisation and mapping. These applications rely on the accurate extraction and

matching of feature points across images. Feature extraction involves identifying unique, distinguishable features from images, which can then be used for subsequent matching and tracking. These features may include corners, edges, blobs, or more complex structures that can be represented mathematically. Feature extraction is often followed by feature description and matching, in which the extracted features are described with local descriptors and matched across different images. Achieving accurate and reliable results in the presence of various environmental factors, such as varying lighting conditions, occlusions, and image noise, is a key challenge in feature extraction. To address this, researchers have developed various feature extraction methods, such as scale-invariant feature transform (SIFT; Lowe, 2004), binary robust independent elementary features (BRIEF; Calonder et al., 2010), features from accelerated segment test (FAST; Rosten et al., 2006), and oriented BRIEF (ORB; Rublee et al., 2011). These methods have been widely adopted in numerous computer vision applications and have proven effective in achieving fast and accurate feature extraction. However, the methods are characterised by high computational costs, sensitivity to illumination changes, and difficulty in handling large datasets. To overcome these challenges, researchers have explored the application of deep learning techniques to enhance feature extraction capabilities, leveraging the power of neural networks to learn feature representations directly from data.

The images captured by the onboard camera of a drone often feature motion blur, uneven illumination, and occlusion owing to the speed and jitter of the drone during flight. To overcome these problems, a deep learning feature detection algorithm, SuperPoint, was used in this study to further encode aerial images and the corresponding cropped orthoimage tiles to prepare the pre-built database for image matching and retrieval. SuperPoint (DeTone et al., 2018) is a lightweight neural network model for computing image keypoints and local feature descriptors. It is designed to be lightweight and efficient, making it well-suited for real-time applications. SuperPoint features considerably fewer fine-tuned weight parameters than other deep learning-based feature detection methods, such as D2-Net and LF-Net (Dusmanu et al., 2019, Ono et al., 2018); moreover, its parameter file size is only ~800 kb, which makes it suitable for tasks that require fast response and real-time applications on mobile devices. In addition, SuperPoint is highly accurate and robust to challenging imaging

conditions such as motion blur and occlusions, owing to its ability to learn features specific to the problem domain and efficiently handle large datasets.

Figure 3.2 shows the framework of the clustered SuperPoint feature extraction. The input is a one-dimensional greyscale image with size $W \times H$. The image is passed to a convolutional neural network (CNN) from a VGG-16 (Simonyan et al., 2014) architecture encoder. The encoder consists of 10 convolutional layers, pooling-based spatial downsampling layers, nonlinear activation functions, and three max-pooling layers. The input greyscale image of size $W \times H$ is first resized to 256 × 256 pixels and then normalised to zero mean and unit variance. The convolutional layers in the encoder extract features from the input image at different spatial resolutions. The first few convolutional layers capture low-level features such as edges and corners, while the deeper layers capture high-level features such as object parts and textures. All of the features are delivered to max-pooling layers for feature-map downsampling and spatial resolution reduction. This step enhances the robustness of the network to variations in scale and viewpoint. After each convolutional layer in the encoder, a nonlinear rectified linear unit (ReLU) activation function is applied to introduce nonlinearity into the network and improve its expressive power. In the last step, three max-pooling layers are implemented to reduce the feature maps to a size of $W/8$, $H/8$, and D.

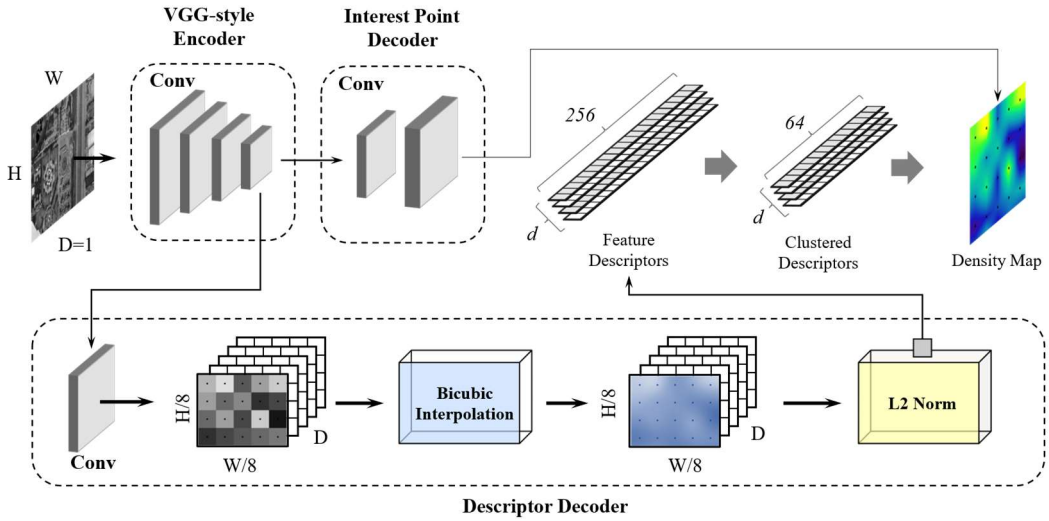


Figure 3.2 Framework of the SuperPoint feature extraction

The intermediate tensor with a size of $W/8$, $H/8$, and D is first sent to a neural network similar to the universal correspondence network (UCN; Choy et al., 2016) for descriptor

vector refinement. The descriptor decoder of SuperPoint uses a learned pooling operation to aggregate local feature descriptors around a given interest point. This pooling operation is parameterised by a set of weights that depend on the spatial location of each descriptor relative to the keypoint. The UCN-like neural network is used to learn these weights end-to-end. The output of the network (a set of weights) enables the descriptor decoder to consider the spatial relationships between the features in the images and results in more robust and discriminative descriptors. Then, the size of the feature descriptor maps is fixed via bicubic interpolation. This up-sampling step is performed to obtain a dense representation of local geometry and appearance information around the interest points in a grid of feature values. Afterwards, the resulting descriptor vector is typically L2-normalised and clustered, which scales the vector to unit length, to improve the robustness of the descriptor for matching across multiple images.

The robustness and execution time of traditional and deep learning-based feature detection methods has been assessed by implementing homography estimation on sequences of the benchmark dataset HPatches (Balntas et al., 2017). This involved performing nearest neighbour matching of the interest points and descriptors detected in the first image with those in the second image sourced from the benchmark dataset. The dataset comprised 116 image sequences that were grouped into two categories: illumination changes and viewpoint alterations (Figure 3.3). The former consisted of 57 sequences that exhibited exclusively photometric modifications, while the latter consisted of 59 sequences characterised by geometric deformations. This division differentiated the effects of changes in illumination conditions from the effects of variations in viewpoint. Each sequence consisted of one reference image and five target images.

Figure 3.3 Examples of two viewpoint image sequences (rows 1 and 2) and two illumination image sequences (rows 3 and 4) from the HPatches dataset.

Table 3.1 Mean execution times and mean average precision (mAP) of three tasks for traditional and deep learning-based detector–descriptor pairs.

Detector	**Descriptor**	**Execution Time (ms)**	**mAP**		
			Verification	Matching	Retrieval
SuperPoint	*SuperPoint*	13	29%	57%	29%
LF-Net	*LF-Net*	196	5%	52%	37%
ORB	*ORB*	17	15%	45%	23%
FAST	*BRIEF*	32	10%	48%	17%
FAST	*SIFT*	163	27%	60%	64%
SIFT	*SIFT*	195	11%	59%	26%
SIFT	*BRIEF*	122	14%	56%	35%

The evaluation methods of above algorithms were based on the study by Balntas et al. (2017). The robustness of the traditional and deep learning-based detectors and descriptors was represented by the mean average precision (mAP) of three tasks: keypoint verification, image matching, and keypoint retrieval. The mAP was determined according to the precision and recall values of a ranked list, L_K, with K elements. The precision and recall values were computed for every $k < K$, which refers to the top-k elements of the ranked list. Precision and recall were calculated for L_K, and the values were averaged across all L_k instances where the recall increases. This process resulted in the computation of the average precision measure for the ranked list L_K.

The evaluation results (Table 3.1) reveal the performances of three top traditional feature detectors and descriptors and two deep learning-based methods. For traditional algorithms, the default parameters from OpenCV (Bradski, 2000) were used. For SuperPoint and LF-Net evaluation, pre-trained models were implemented according to the fine-tuned outdoor weight provided by DeTone et al. (2018). SuperPoint yielded the most favourable results on the three tasks, outperforming LF-Net on the keypoint verification task. The results highlight the superior computational efficiency of SuperPoint, with a keypoint detection time of 13 ms for a single image. It was significantly faster than LF-Net, whose execution time was 60 times longer. The evaluation also showed the performances of different traditional detector–descriptor combinations. The SIFT detector and descriptor yielded the most successful outcomes, particularly in image matching and retrieval tasks. However, the BRIEF descriptor performed poorly in all combinations. While the FAST + SIFT combination outperformed other traditional algorithms on all three tasks, its execution time (163 ms) was significantly longer than that of ORB + SuperPoint. The evaluation results emphasise the effectiveness of SuperPoint for practical applications requiring rapid performance.

3.2.2 Feature-Based Matching for Cross-View Image Retrieval

In image matching and retrieval, similarity search is typically performed through feature-based approaches. Features are distinctive descriptors extracted from an image, and they capture local geometric information for image recognition. To identify the location of a query image in an orthoimage, we propose a feature-based similarity search approach for image place recognition that involves the following steps: 1) Global feature points and descriptors are extracted from the orthoimage base map and then saved as an individual file and as part of the pre-built database for image retrieval. 2) Feature points and descriptors extracted from the query image are also extracted to match the spatial structure obtained from the previous step. 3) The k-nearest neighbour (KNN; Cover et al., 1967) search algorithm is used to compute the Euclidean distance between the feature point descriptors of the query image and those in the pre-built database. 4) The k search results with the smallest L2 distance to the feature point

descriptors of the query image are selected. 5) The corresponding cropped orthoimage tile of the query image in the database is retrieved according to the maximum density of feature points among the *k* search results. Figure 3.4 illustrates the framework of our approach, and each step is detailed below.

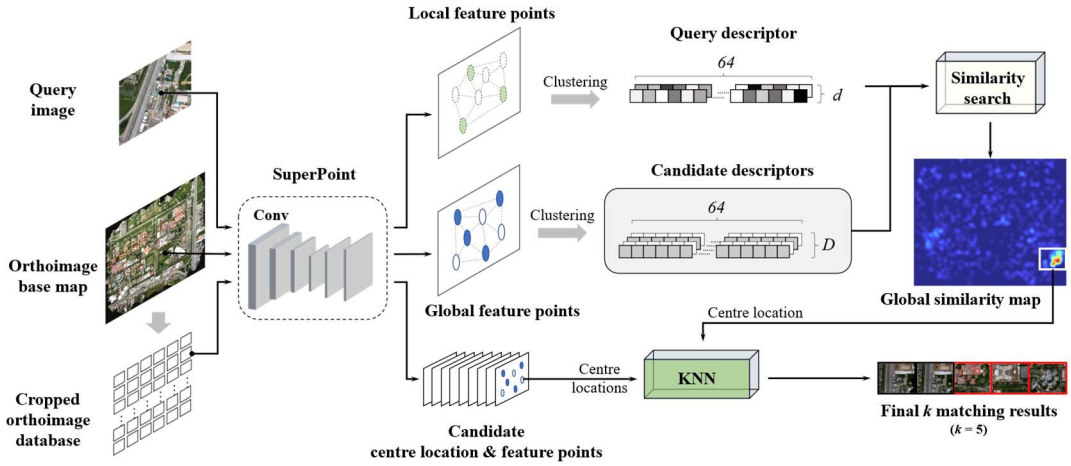


Figure 3.4 Framework of feature-based matching for cross-view image retrieval.

After the extraction of global feature points and descriptors from the orthoimage base map, principal component analysis is applied to reduce the redundancy of the spatial and geometric dimensionalities of features. Then, clustered feature descriptors are saved as an individual file in the pre-built database for further image matching and retrieval. The KNN algorithm is adopted owing to its ability to efficiently identify the *k* most similar or nearest vectors in the database for the query vector *a* using a Euclidean distance metric. Given a query vector of the feature point descriptor $x_i \in \mathbb{Q}^d$ and the database of vector collection $y_i \in \mathbb{C}^d$, we conduct the following search:

$$L_p(x_i, y_i) = k \cdot argmin \left| \left(\sum_{l=1}^{n} \left| x_i^{(n)} - y_i^{(n)} \right|^p \right)^{\frac{1}{p}} \right| \tag{3.1}$$

where $i \in \{1, \ldots, l\}$ indicates the number of vectors in the database, and n is the vector dimensionality. The Euclidean distance with $p = 2$ is denoted as the L2 distance. This metric has been mentioned earlier and is commonly used as a similarity measure in several applications, including image retrieval. This distance metric is often preferred owing to its attractive linear algebra properties, making it well-suited for tasks that

involve the learning of multiple embedded vectors. Specifically, the L2 distance is optimised by design, allowing for efficient computations and effective similarity comparisons between feature vectors (Muja et al., 2014). Its mathematical properties make it a popular choice in various similarity search tasks, including image matching and retrieval, owing to its effectiveness in capturing the pairwise distance between feature descriptors. The smallest L2 distance is collected via *k*-selection. For an array a_i, *k*-selection identifies the k lowest valued elements $a_{s_j}\left(s_j \in \{1, \dots, l\},\ j \in \{0, \dots, k\}\right), a_{s_j}$ indicate the elements from the input array s_j. Because each image contains n feature point descriptors, a batch similarity search is performed to identify the k most similar results through a comparison of the n feature point descriptors over the m descriptors from the database. Batching for *k*-selection entails selecting $n \times k$ elements and indices from n separate arrays.

The KNN similarity search results in a k set of vectors containing n feature points descriptors. We determine the location of n feature points according to the saved feature points $\mathbb{R}$ extracted by SuperPoint in previous steps. Then, a similarity map is generated following the feature point distribution. The similarity value is expressed as D_i ($i \in \{0, \dots, n\}$), and the centre location (x_D, y_D) of the maximum feature point similarity is calculated as

$$x_D, y_D = loc(argmax|D_i|) \tag{3.2}$$

Once the centre of maximum feature point density is located, the pixel coordinates will be saved as an index to search for the closest centre coordinates of the cropped orthoimage tiles in the database using KNN. This step serves as a two-way verification and constraint for the previous feature point similarity search, enhancing the robustness of the final search result. The KNN algorithm allows for more flexibility when executed on multiple CPU threads or GPUs. It was adopted in our study following the approach proposed by Johnson et al. (2019).

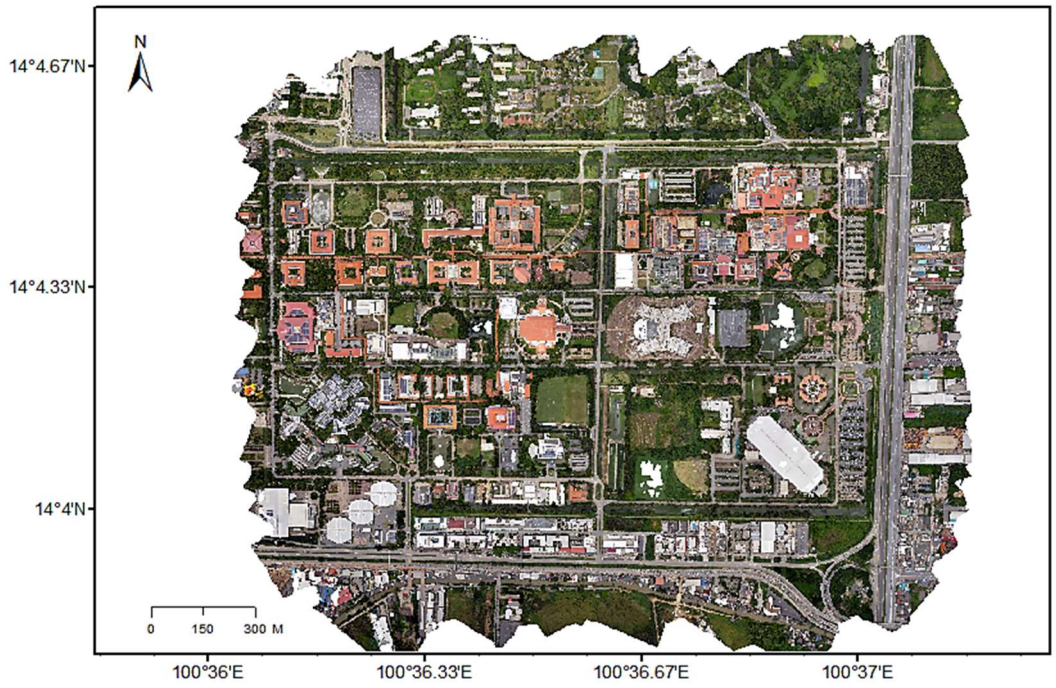
3.2.3 Experimental Analysis of Cross-View Image Matching and Retrieval

3.2.3.1 Pre-Built Database Construction

The orthoimage base map used for image matching was generated from an open-access aerial image dataset. The dataset comprised over 400 aerial photographs collected by an eBee X drone equipped with an Aeria X photogrammetry camera (senseFly, 2019). For further analysis, the orthoimage was cropped into multiple sections to create the image dataset. SuperPoint extracted the feature points and descriptors from the cropped image sections. The feature points and descriptors were represented as vector data and saved separately as files. The resulting vector database obtained from this process can be utilised for similarity search in image retrieval.

Figure 3.5 presents an overview of the orthoimage generated by the aerial images. The image size was 70,391 × 59,269 pixels, covering an area of ~0.033 km^2 with a 1 m resolution. The input to the framework was an orthoimage, from which global feature points and descriptors were extracted using SuperPoint and saved as a separate file for further image matching. Additionally, the orthoimage was cropped into several sections. The central location of each cropped tile on the orthoimage and the feature points and descriptors were saved for further image retrieval. The size of the cropping window was represented as (W_d, H_d). The cropping window was moved along the long side of the orthophoto, with the movement step determined by the end lap and side lap selections. The step was inspired by the flight planning of aerial photogrammetry, which emphasises the importance of image overlap during the capture of consecutive photos along and adjacent to a flight strip. As noted in the Chapter 2, ‘end lap’ and ‘side lap’ refer to the overlap between consecutive photos captured by the camera along and adjacent to a flight strip, respectively. Generally, the end lap and side lap in aerial photogrammetry are set to 60% (Wolf et al., 2014). The present study employed a cropping strategy to produce pre-built database, with each cropped image piece having 80% overlap on both the end and side laps. The image tiles in the pre-built database were achieved through the cropping of the orthoimage into smaller pieces, and

consecutive photos were captured according to the overlapping requirement. This approach was adopted to enhance the quality and completeness of the pre-built database.



(a)

(b)

Figure 3.5 (a) Overview of the orthoimage base map for constructing the database for image retrieval, and (b) thumbnails and examples of cropped tiles in the database

3.2.3.2 Performances of Cross-View Image Matching and Retrieval

The orthoimage was cropped into over 900 sections, and each section was passed to SuperPoint for feature point and descriptor extraction. The centre location of each image section was also saved during this processing. As shown in Figure 3.4, the aerial image was first extracted using SuperPoint, with *d* number of feature points and descriptors. The *k* sets from the global feature descriptor most similar to the query features were found via a similarity search, and each set contained *d* vectors of feature descriptors. In the experiment, we set *k* to 5 to obtain the top five sets of search results most similar to the query vector. Then, the maximum density value and corresponding image coordinates were found on the density map, which was generated according to the search results. The KNN algorithm was applied again to find the cropped tiles in the pre-built database.

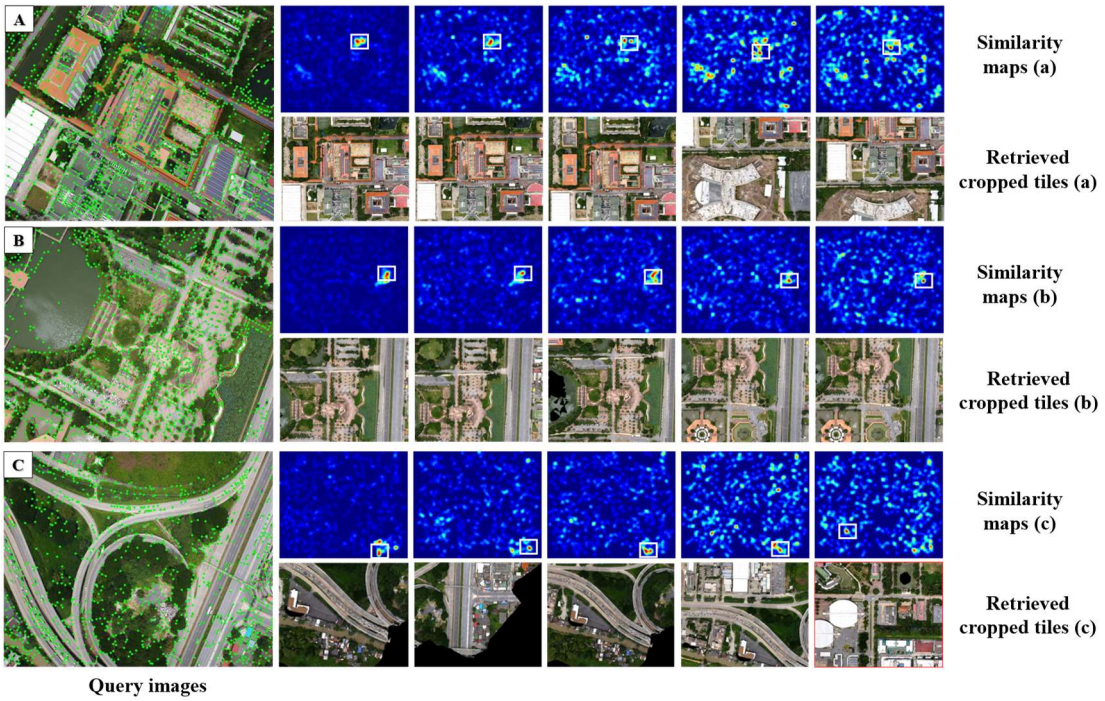


Figure 3.6 Experimental result of feature-based image matching and retrieval. (a), (b) and (c) are three example query images with different landscapes. The top five similarity maps of each query image and corresponding cropped orthoimage tiles were retrieved from the pre-built database

Figure 3.6 demonstrates our proposed approach for the matching and retrieval of the cropped tiles of the aerial image in the pre-built database. Figure 3.6(a) shows an aerial

image of rich-texture areas, such as buildings, roads, and green belts. The top half of Figure 3.6(a) presents similarity maps of the top five most similar feature points, and the corresponding nearest cropped tiles are shown below. Because features in this region are easily extracted, the abundance of features in this region results in relatively reliable search results. In contrast, the query image in Figure 3.6(b) contains areas with less texture (e.g., trees and lakes) than in Figure 3.6(a). However, the search results still enable the successful retrieval of the location of the query aerial image. In contrast, Figure 3.6(c) shows a query image with similar content to Figure 3.6(b) but with even less textured areas. Extracting features from this region is challenging, and the descriptors in such areas, such as trees and green belts, are rather similar to the global feature descriptor. Hence, the search results may result in incorrect image retrieval, as demonstrated by the last image retrieval results at the bottom of Figure 3.6(c).

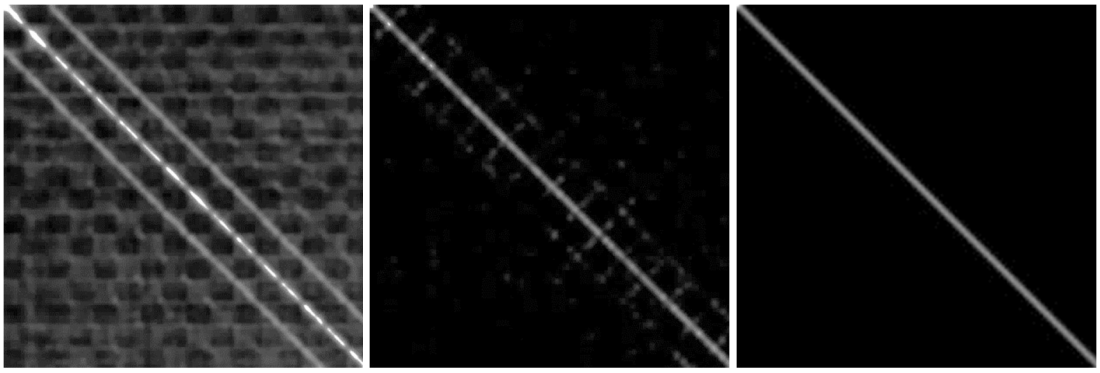
Table 3.2 Accuracy comparison between our methods and other methods

Methods	**R@1**			**R@5**		
	+ve	−ve	Precision	+ve	−ve	Precision
VGG-16	203	236	46.2%	315	124	71.8%
SuperPoint	225	214	51.3%	282	147	64.2%
Our method	263	176	**59.9%**	320	119	**72.9%**

To comprehensively evaluate the performance and accuracy of our proposed method, we analysed the number of positive (+ve) and negative (−ve) results for the top one and top five search results. Furthermore, we compared the similarity search and image retrieval results obtained using feature points and descriptors from VGG-16 and SuperPoint with the results obtained using feature points and descriptors from our method. The pre-built database consisted of 921 cropped image sections and 439 aerial images used in our evaluation and experiment. As shown in Table 3.2, our method successfully retrieved 253 images in the top one result, with a precision of ~60%, demonstrating its effectiveness compared with other methods. VGG-16 exhibited the worst performance in both the top one and top five search results. The proposed method exhibited the most robust performance in the top five search results. The favourable performance and robustness of the proposed approach demonstrate its potential in real-time localisation and navigation applications. Moreover, our evaluation results

demonstrate the potential of our proposed approach for practical aerial image retrieval tasks, providing a solid basis for further research and development in this area.

Using only aerial images, we conducted a comprehensive evaluation to assess the accuracy and reliability of the proposed approach for image retrieval. Different methods were adopted to retrieve aerial images from a dataset of over 400 aerial images. The evaluation was performed using a confusion matrix, which is commonly used for evaluating the accuracy and reliability of a model or algorithm in machine learning and data analysis. The confusion matrix typically consists of rows and columns representing the sample and actual values (ground truth). In our study, the rows of the confusion matrix represent the query image index, while the columns represent the index of the aerial images in the dataset as the reference. By analysing the confusion matrix, we comprehensively evaluated the performance of our approach and its ability to accurately retrieve aerial images from the dataset. The element (i, j) in the confusion matrix represents the similarity value between reference image i and image retrieval result j.



(a) VGG (b) SurperPoint (c) Our method

Figure 3.7 Comparison of confusion matrix between our method and others for similarity searching

Figure 3.7 shows the three confusion matrices for image retrieval. Each square matrix had dimensions of 439 × 439, where 439 represents the total number of aerial images in the dataset. The diagonal cells from top-left to bottom-right represent the correctly retrieved query image. The decimal proportion value indicates the similarity to the ground truth (the query image in the dataset). The off-diagonal cells represent the

mismatches. Figure 3.7(a) presents the confusion matrix obtained using the VGG-16 model. This matrix is used for extracting feature points and descriptors in the context of aerial image retrieval from the dataset. The descriptors extracted from VGG-16 were used for similarity search to retrieve images. The results indicate that the VGG-16 feature descriptor approach exhibited relatively high mismatches, suggesting its limitations for aerial image retrieval. The image retrieval based on only feature descriptors extracted by SuperPoint was evaluated using a confusion matrix (Figure 3.7b). The results indicate that the SuperPoint feature descriptor improved robustness in similarity search compared with the VGG-16 feature descriptor. Figure 3.7(c) displays the confusion matrix obtained through our approach after feature point similarity evaluation and density comparison. Our approach showed a higher precision than the other two methods, with fewer mismatches in the off-diagonal cells. These findings suggest that our approach yielded more robust and reliable image retrieval results than VGG-16 and SuperPoint feature descriptors.

3.3 Camera Pose Determination by the Integration of VO and Space Resection

VO is a computer vision-based technique that enables a machine or robot to estimate its position and orientation in the environment by analysing visual information from a camera or multiple cameras. It is a critical technology for navigation and positioning in various applications, including autonomous vehicles, drones, robotics, and augmented reality. VO relies on extracting visual features, such as keypoints or landmarks, from consecutive images or video frames and then tracking their motion over time to estimate the relative camera motion. VO involves analysing the changes in the visual features to estimate the camera pose (position and orientation) in 3D space, usually in relative frames. However, traditional VO methods often struggle under low-illumination conditions, fast motion, and large camera rotations. To overcome these challenges, a state-of-the-art deep learning algorithm has been introduced as a powerful technology for enhancing monocular VO.

Space resection is a fundamental technique used in photogrammetry to determine the absolute camera pose in a 3D space. It involves estimating the camera position and

orientation relative to a known coordinate system. Space resection can accurately calculate the camera pose by analysing the correspondences between 2D image points and their corresponding 3D world points. First, a set of known 3D points is selected, and then the corresponding 2D projections in the image are identified. The camera exterior orientation parameters can be solved using the collinearity equation with the camera interior orientation parameters. The reprojection error and the camera pose are minimised and optimised through the iterative adjustment of the camera position and orientation until the projected 3D points are aligned with their corresponding 2D points in the image. This enables the conversion of image coordinates to world coordinates, facilitating the accurate mapping of the camera position in space.

In the proposed approach, VO was used for the relative pose estimation of aerial image series, and space resection was used to determine the absolute camera pose of the keyframes (e.g., the turning point of the flying path) of the aerial images and transfer the VO results to absolute scales. The concept of the proposed approach is illustrated in Figure 3.8.

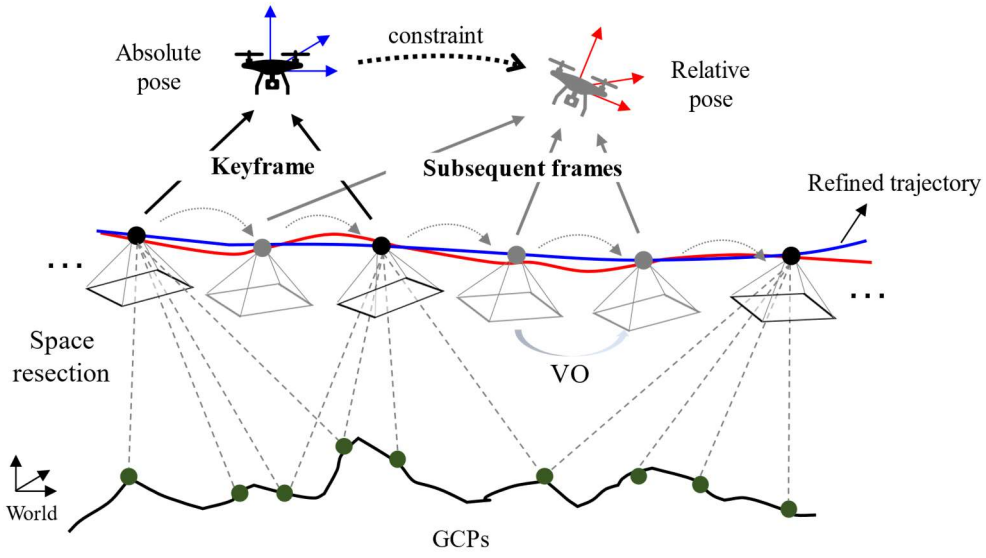


Figure 3.8 Overview of the integration of VO and space resection for camera pose determination. The absolute pose of the keyframe obtained via space resection is used as a constraint on the relative pose of the subsequent frames estimated via VO, resulting in a refined trajectory.

3.3.1 Feature-Based VO

Feature-based VO methods, which rely on the detection and matching of features, such as corners or keypoints, between consecutive images to estimate camera motion, are commonly used in navigation and positioning (Aqel et al., 2016, Nistér et al., 2004). However, these traditional feature-based VO algorithms have several limitations. They are often sensitive to lighting conditions, as changes in illumination can adversely affect feature quality and repeatability, leading to inaccurate feature detection and matching, which can result in poor motion estimation. Additionally, these methods may have limited robustness to motion blur and occlusions, as fast motion, motion blur, and occlusions can limit the accuracy of feature tracking across consecutive frames. Traditional feature-based methods may struggle to effectively handle these situations, resulting in degraded motion-estimation accuracy. Furthermore, traditional feature-based methods may face difficulties in accurately estimating motion in large camera-rotation scenarios. Large camera rotations can cause changes in the appearance and geometry of the scene, leading to feature mismatches and inaccurate motion estimates.

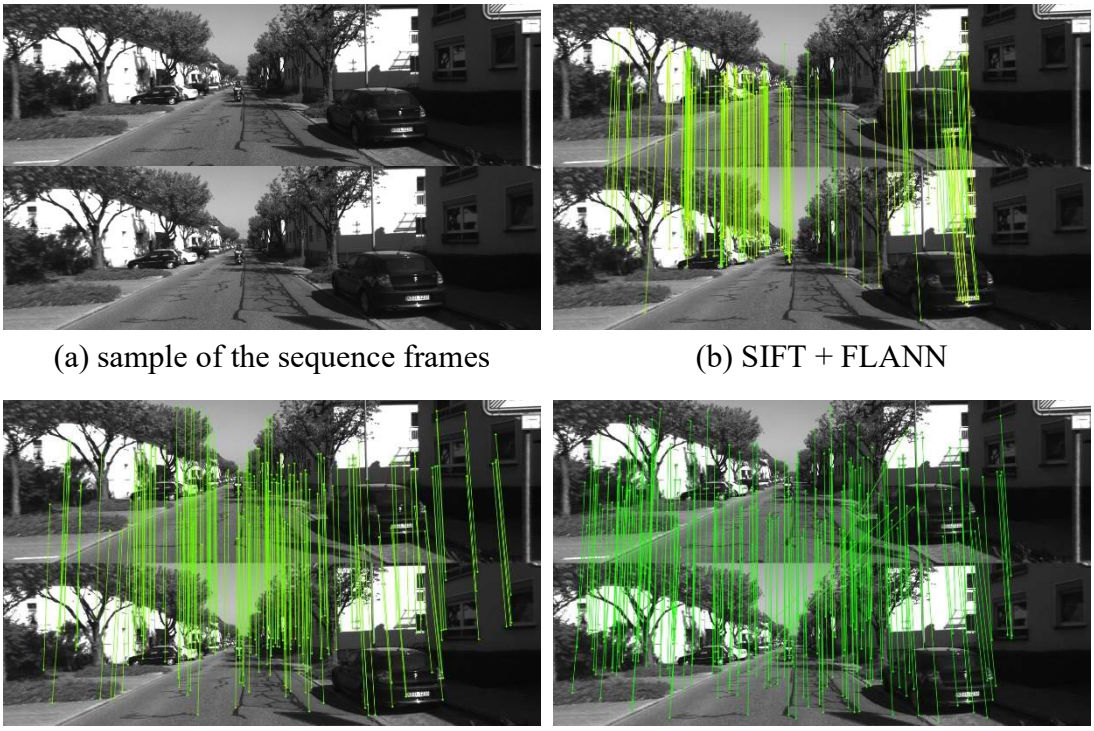
In contrast, deep learning-based VO algorithms such as SuperPoint and SuperGlue (Sarlin et al., 2020) offer several advantages in navigation and positioning, such as their robustness to varying lighting conditions. The state-of-the-art deep learning-based methods can learn robust features from images with varying illumination conditions, enabling accurate feature matching even in low-light or varying-lighting environments. Moreover, these methods can capture complex and discriminative features from images. They can learn sophisticated features, leading to more accurate and reliable feature matching, even in challenging scenarios with motion blur, occlusions, or large camera rotations. Additionally, state-of-the-art methods such as SuperPoint and SuperGlue perform end-to-end feature extraction and matching while considering the global context, which can help improve the overall pose estimation accuracy. These advantages endow the state-of-the-art VO algorithms with improved performance and reliability in motion estimation tasks, making them promising alternatives to traditional feature-based methods.

3.3.1.1 Accuracy Evaluation of VO Methods

The performances of various VO algorithms on the KITTI odometry benchmark dataset were evaluated using the average distance error and the relative distance error (RDE; Geiger et al., 2013). The KITTI odometry benchmark dataset is widely used for evaluating the accuracy, robustness, and real-time performance of VO algorithms for monocular cameras. The dataset serves as a benchmark for comparing different algorithms and assessing their performances in challenging driving scenarios, such as unfavourable lighting conditions, unfavourable weather conditions, and complex scenes. The KITTI odometry benchmark dataset includes 22 monocular camera sequences with over 4,000 frames captured from a vehicle driving in urban and highway environments. The sequences cover ~39 km and include diverse scenes such as urban streets, residential areas, highways, and tunnels. The sequences were captured at a frame rate of ~10 Hz, with a resolution of 1241 × 376 pixels. The ground truth poses of the camera are provided in the dataset, which allows for the evaluation of the accuracy of VO algorithms. The ground truth poses were obtained using a high-precision laser-based Velodyne HDL-64E LIDAR sensor and a high-accuracy GPS/INS system. The ground truth poses are provided as 3D translation vectors and 3D rotation matrices. Figure 3.7 illustrates the experimental results of the traditional and deep learning-based feature detection and matching methods.

Figure 3.9(a) shows the original two consecutive frames from one sequence of the KITTI benchmark dataset. To demonstrate the limitations of the traditional feature-based methods, Figure 3.9(b) presents the feature-matching results obtained using the SIFT and FLANN (fast library for approximate nearest neighbours) algorithms for feature detection and matching, respectively. Despite its popularity, the traditional SIFT + FLANN algorithm may have limitations, such as reduced feature extraction and matching robustness under varying conditions. Figure 3.9(c) shows the results obtained using the SuperPoint deep learning-based algorithm for feature detection while maintaining the same feature-matching method as in Figure 3.9(b). SuperPoint, specifically designed to capture complex and discriminative features, showed improved feature extraction and matching robustness. Furthermore, Figure 3.9(d) presents the results of SuperPoint + SuperGlue, in which SuperPoint is used for feature detection and SuperGlue, another deep learning-based method, is used for feature matching. The

use of SuperPoint + SuperGlue demonstrates the potential of end-to-end deep learning-based algorithms for achieving improved feature-matching performance.

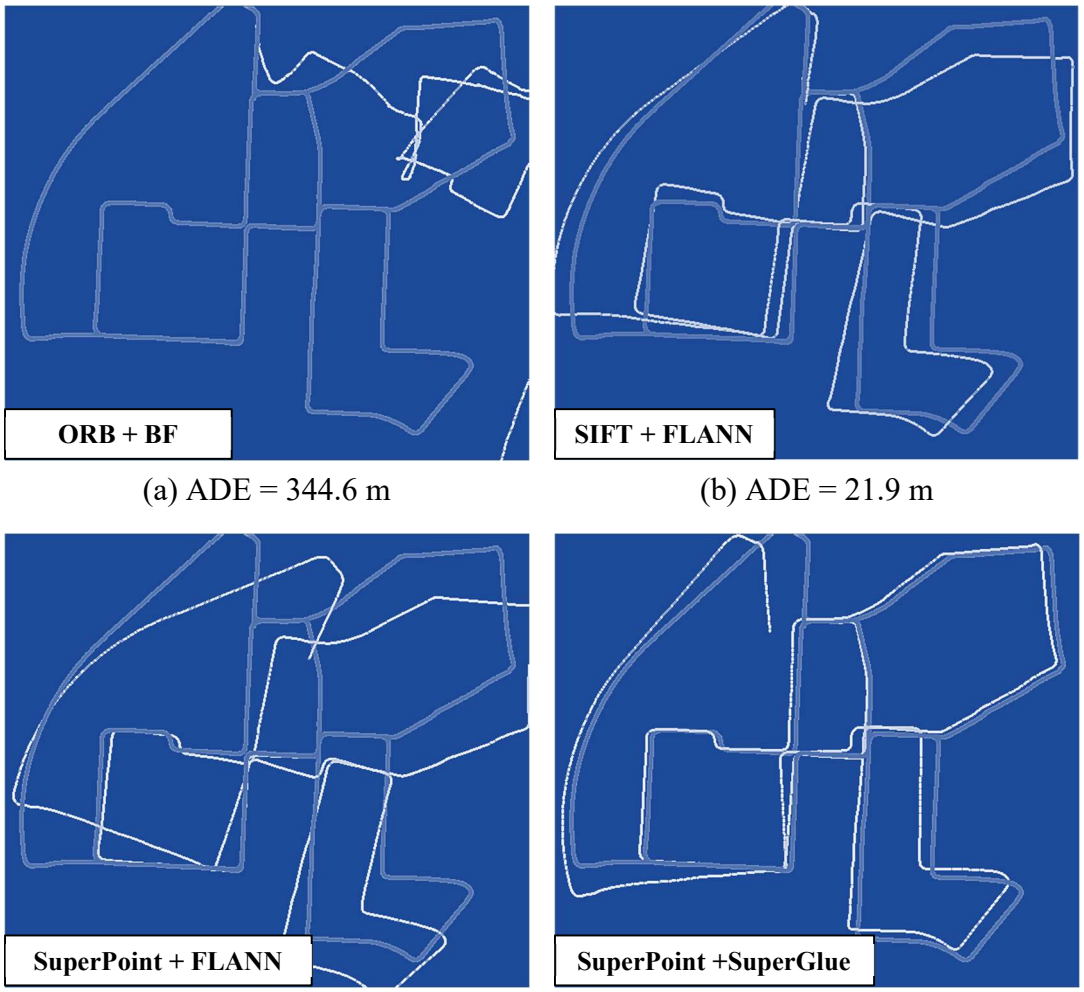


(a) sample of the sequence frames (b) SIFT + FLANN

(c) SuperPoint + FLANN (d) SuperPoint + SuperGlue

Figure 3.9 Experimental results of different feature detection and matching methods.

To further evaluate and compare the accuracies of traditional and state-of-the-art VO methods, the methods were applied to all frames in the KITTI benchmark dataset, and the average distance error and the RDE were calculated. The absolute distance error (ADE) measures the absolute difference in distance between the estimated camera trajectory and the ground truth trajectory. A lower average distance error indicates higher accuracy in camera motion estimation. The RDE measures the relative difference in distance between the estimated camera trajectory and the ground truth trajectory at each time step and frame. A lower RDE indicates better consistency in relative motion estimation between consecutive frames. Figure 3.10 compares the experimental results of the estimated camera trajectory (shallow line) with the ground truth trajectory (dark line).



(a) ADE = 344.6 m (b) ADE = 21.9 m

(c) ADE = 37.6 m (d) ADE = 15.5 m

Figure 3.10 Experimental results of estimated camera trajectories obtained via different methods compared with the ground truth trajectory

The camera trajectory estimates obtained using SuperPoint + SuperGlue (Figure 3.10d) closely reproduced the ground truth trajectory, indicating the superior performance of deep learning-based methods. In contrast, when combined both the traditional feature point extraction algorithms (Figures 3.10b) and the deep learning-based feature point extraction algorithms with FLANN for feature point matching (Figures 3.10c), the results deviate moderately from the ground truth. The estimated camera trajectory obtained using ORB + BF showed a significant deviation from the ground truth (Figure 3.10a). The statistics of each approach for camera trajectory estimation are presented in Figures 3.11 and 3.12. The ADE measures the cumulative error over time, while the RDE evaluates the accuracy of camera pose estimation between two consecutive frames.

The results in Figure 3.11 show that the SuperPoint + SuperGlue algorithm exhibited the lowest ADE, with an average error of 15.5 m, while the ORB + BF algorithm exhibited the highest ADE, with an average error of 344.6 m. The SIFT + FLANN algorithm exhibited an ADE of 21.9 m, and the SuperPoint + FLANN algorithms exhibited an average ADE of ~37.6 m. Both algorithms exhibited similar pattern offsets in their trajectories, possibly because both algorithms were used with the same matching algorithm. As shown in Figure 3.12, ORB + BF exhibited a substantially higher RDE (0.748 m on average) than the other algorithms, while SIFT + FLANN exhibited the lowest RDE (0.085 m on average). The SuperPoint + FLANN and SuperPoint + SuperGlue algorithms exhibited an average RDE of 0.177 and 0.103 m, respectively. These results suggest that the SuperPoint + SuperGlue algorithm outperformed the other algorithms in overall accuracy, while the ORB + BF algorithm exhibited the highest error.

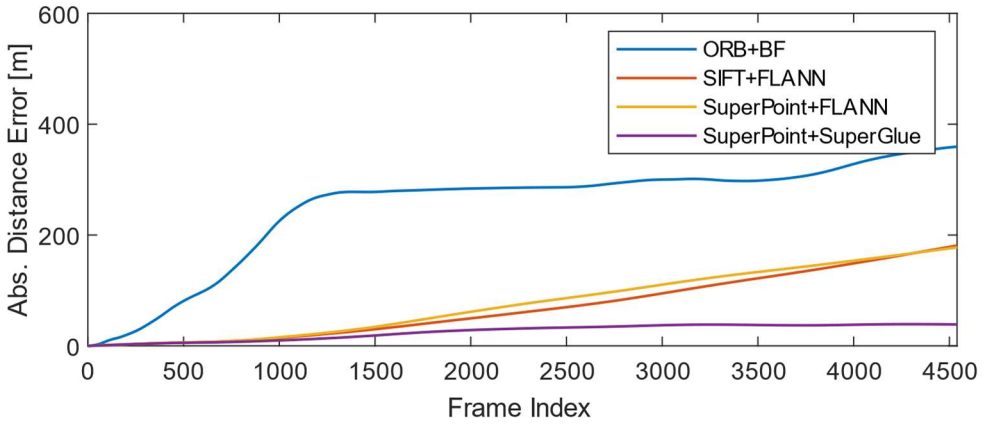


Figure 3.11 ADE between the estimated camera trajectory and the ground truth

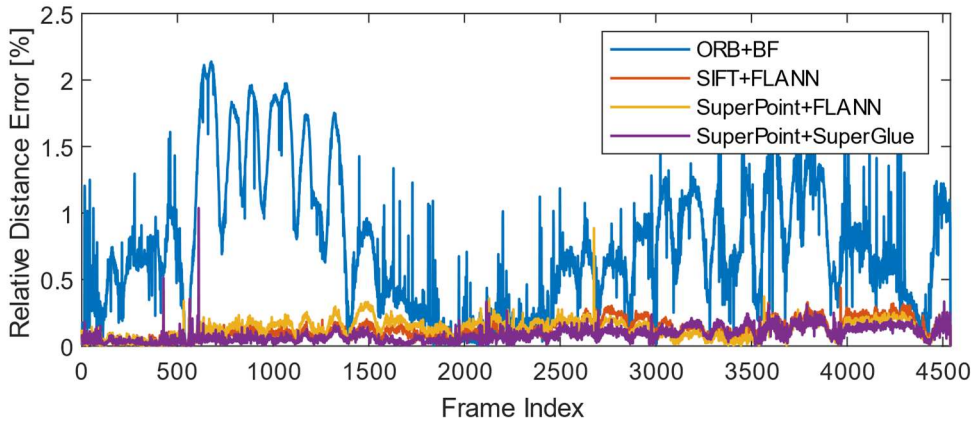


Figure 3.12 RDE between the estimated camera trajectory and the ground truth

3.3.1.2 Efficiency Evaluation of VO Methods

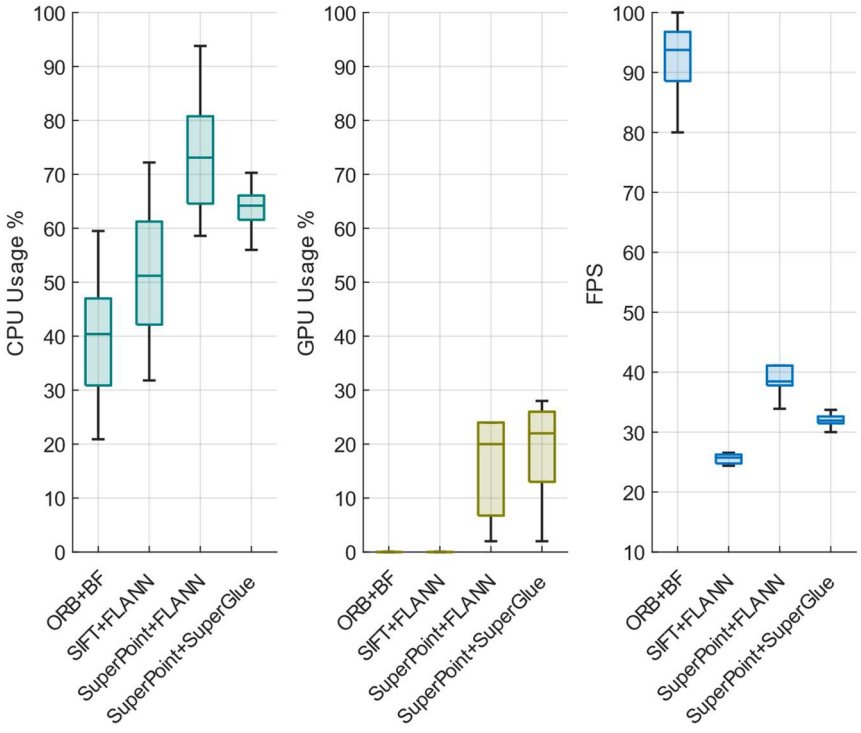
According to the above analysis, SuperPoint and SuperGlue demonstrated high accuracy and precision in feature matching, which are crucial for accurate motion estimation. SuperGlue, a deep learning-based feature-matching algorithm, further enhances the accuracy and robustness of feature matching compared with traditional methods such as BF or FLANN.

To evaluate the efficiency of different feature detection and matching algorithms, we considered three aspects: CPU usage, GPU usage, and frames per second (FPS). CPU usage refers to the amount of processing power required by the algorithm to run on the CPU. A higher CPU usage indicates that the algorithm is computationally intensive and may not be suitable for real-time applications and low-power devices. GPU usage refers to the amount of processing power required by the algorithm to run on a GPU. The speed of many feature detection and matching algorithms can be enhanced by running them on a GPU rather than a CPU. FPS refers to the number of frames per second that the algorithm can process. A higher FPS indicates that the algorithm can process more frames in real time and is more suitable for applications requiring real-time processing. According to these criteria, the efficiencies of the traditional and state-of-the-art VO methods were evaluated on a PC with an Intel Xeon E5-2603 v4 CPU and an NVIDIA GeForce RTX 2080Ti GPU.

Figure 3.13 presents the efficiency evaluation results of each VO method on the KITTI benchmark dataset. ORB is a feature descriptor algorithm that primarily runs on a CPU. Among the traditional VO methods, the ORB + BF algorithm is CPU-based. It runs entirely on the CPU, with an average usage of 40% CPU resource, and does not require GPU resource. Because ORB is a relatively lightweight feature detection algorithm and the BF algorithm used for matching is relatively simple, the efficiency of ORB + BF exceeded 90 FPS. SIFT is a computationally expensive feature detector. It involves multiple steps, including scale-space extrema detection, keypoint localisation, orientation assignment, and descriptor computation. These steps can be computationally intensive and may require significant CPU usage. The FLANN matching algorithm is more complex than BF and may require more CPU resources. Overall, the CPU usage of SIFT + FLANN (52% on average) was higher than that of

ORB + BF. SIFT can utilise the GPU for certain computations, but the CPU typically handles most of its workload. FLANN does not have GPU implementations. Therefore, its GPU usage should also be negligible. Figure 3.11(b) shows that SIFT + FLANN exhibited a lower FPS (27 on average) than ORB + BF, owing to the computational complexity of the former.

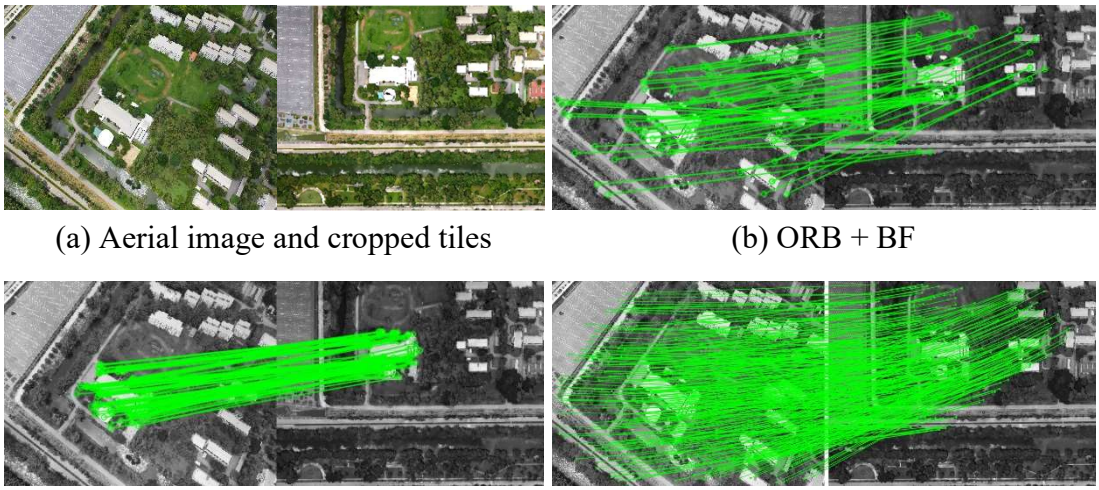
Among the state-of-the-art VO methods, SuperPoint + FLANN was less efficient than ORB + BF and SIFT + FLANN, as it requires a GPU for feature extraction. FLANN consumed ~74% of CPU resources for loading CNNs and feature matching, and SuperPoint consumed ~25% of GPU resources for feature extraction. SuperPoint + FLANN exhibited a higher FPS (~40 FPS for the entire evaluation) than SIFT + FLANN. However, both SuperPoint and SuperGlue require GPU for feature extraction and matching. They featured the highest GPU usage among all of the methods, and CPU usage was only for CNN loading. SuperPoint + SuperGlue exhibited a moderate FPS: ~33.



(a) CPU usage (b) GPU usage (c) FPS

Figure 3.13 Efficiency evaluation for each VO method.

The above VO methods were applied to our aerial image and pre-built database to evaluate the effectiveness of SuperPoint and SuperGlue on real-time motion estimation. We compared the methods with traditional feature detection and matching methods.



(a) Aerial image and cropped tiles (b) ORB + BF

(c) SIFT + FLANN (d) SuperPoint + SuperGlue

Figure 3.14 Experiment of feature matching with different methods

Table 3.3 Comparison of correct matches and execution times for each feature-matching method

Methods		**Number of Feature Points**		**Correct Matches**	**Execution Time (ms)**		
Detection	*Matching*	*Aerial Images*	*Cropped Tiles*		*Detection*	*Matching*	*Overall*
ORB	BF	460	463	105	101	8	**109**
SIFT	FLANN	922	887	58	101	137	238
SuperPoint	SuperGlue	917	1187	337	**20**	**102**	122

Figure 3.14 presents the experimental results of the traditional and deep learning-based feature detection and matching methods on aerial images and the corresponding cropped tiles from the orthoimage. Figure 3.14(a) displays the original aerial image and cropped image tiles used in the experiment. SuperPoint + SuperGlue exhibited a larger number of correct matches (Figure 3.14d) than ORB + BF and SIFT + FLANN (Figures 3.14b and c). These results are further substantiated by the data presented in Table 3.3. The table shows that the execution time for SuperPoint feature detection was only ~20

ms, resulting in over 900 detected feature points. The execution time for SuperGlue feature matching was 102.432 ms, with 337 correct matches. In contrast, the execution time for BF matching was only 8 ms, but ORB consumed much more time for feature detection. The overall execution time for SuperPoint and SuperGlue was only ~122 ms per frame, which is significantly shorter than those of SIFT + FLANN (238 ms) and ORB + BF (109 ms). Although ORB + BF is a fast algorithm for feature extraction and matching, using SuperPoint + SuperGlue for these tasks can result in even faster and more accurate performance, as discussed in the previous section on balancing accuracy and efficiency. These findings highlight the computational efficiency and effectiveness of SuperPoint and SuperGlue in real-time aerial image processing.

3.3.2 Space Resection for Camera Pose Determination of Keyframes

In visual navigation and positioning, space resection is used to determine the absolute camera pose of the keyframes in real time according to the detected ground control points (GCPs). In space resection, the camera position and orientation are computed using collinearity equations that relate the image coordinates of the GCPs to their known ground coordinates. As introduced in Chapter 2, the exterior orientation parameters of the camera position (X_s, Y_s, Z_s) and orientation (φ, ω, κ) in the scene can be solved using Eq. (3.3):

$$\begin{aligned} x - x_0 &= -f\frac{a_1(X - X_s) + b_1(Y - Y_s) + c_1(Z - Z_s)}{a_3(X - X_s) + b_3(Y - Y_s) + c_3(Z - Z_s)} \\ y - y_0 &= -f\frac{a_2(X - X_s) + b_2(Y - Y_s) + c_2(Z - Z_s)}{a_3(X - X_s) + b_3(Y - Y_s) + c_3(Z - Z_s)} \end{aligned} \tag{3.3}$$

where a_1, a_2, a_3, b_1, b_2, b_3, c_1, c_2, and c_3 are the elements of the rotation matrix consisting of (φ, ω, κ). (x, y) is the image coordinates of the GCPs (X, Y, Z), and (x_0, y_0, f) is the interior orientation of the camera. The space resection accuracy depends on the accuracy of the GCPs, the camera calibration quality, and the image quality. The GCPs must be accurately surveyed and measured to ensure that their known ground coordinates are precise. The camera must also be calibrated to correct for lens distortion and other factors that can affect the image coordinates.

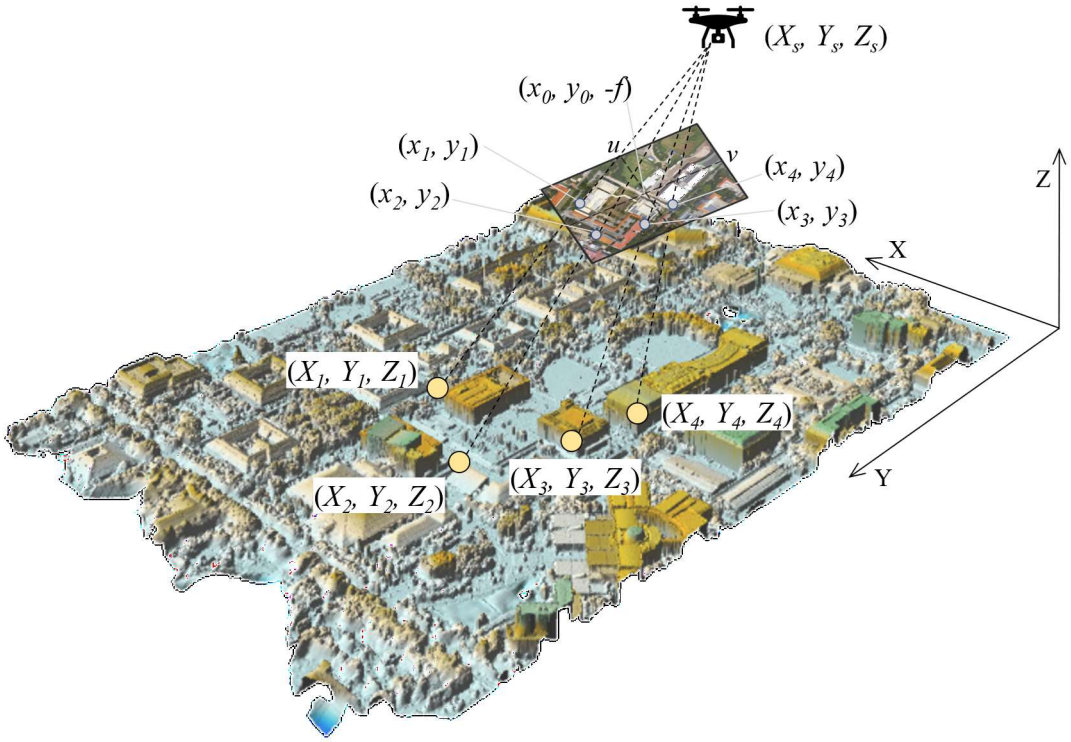


Figure 3.15 Geometry of space resection

In this study, GCPs in real-word coordinates were found on the orthoimage and its corresponding DSM. The geometry of space resection used in this study is given in Figure 3.15. During the processing of the pre-built data, the DSM and the orthoimage were simultaneously generated using all of the aerial images. After the feature points on the aerial image are matched with the corresponding points on the orthoimage, the spatial coordinates of the feature points can be obtained from the DSM according to the image coordinates. Figure 3.16 provides an overview of the DSM generated from the aerial image dataset.

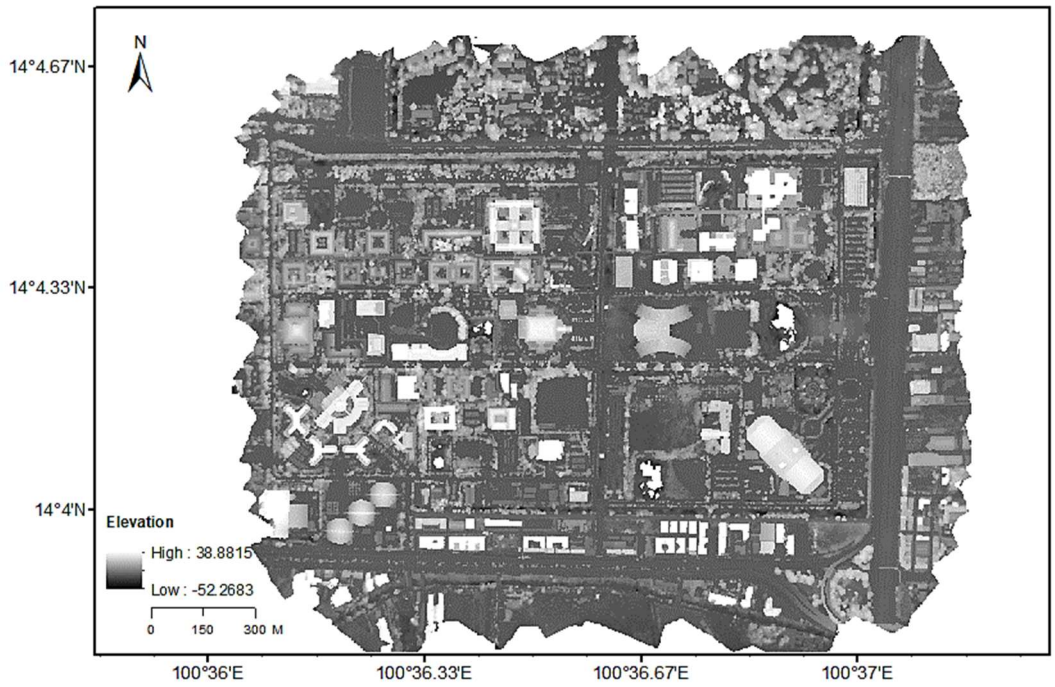
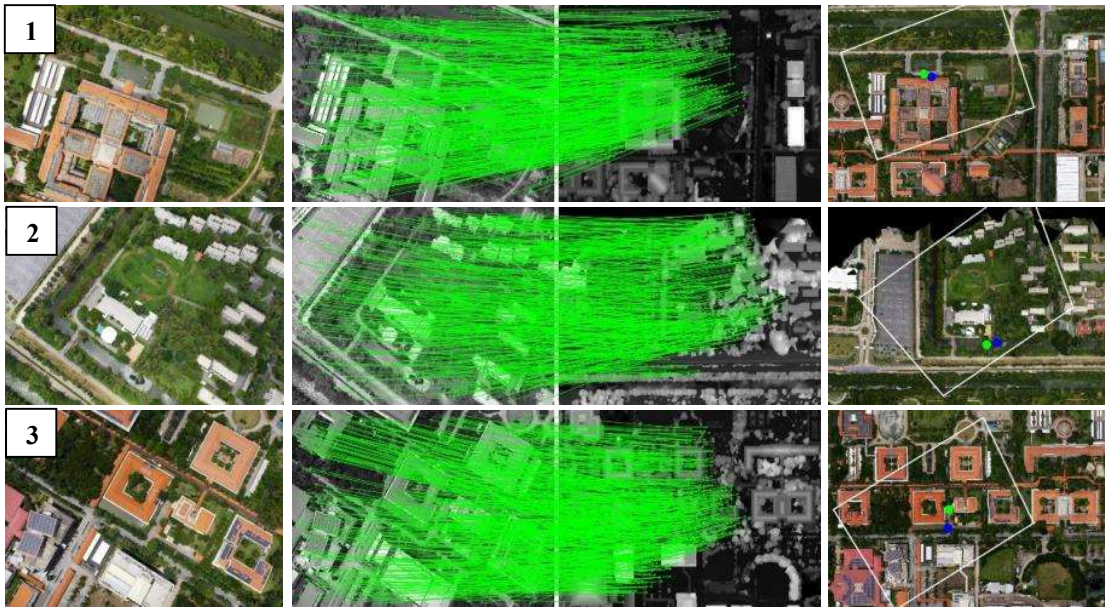


Figure 3.16 Overview of the DSM in the pre-built database

The DSM used in this study had the same image size as the orthoimage (70,391 × 59,269 pixels) and the same resolution of 1 m. The georeferenced DSM contained all spatial information, including elevation, which is essential for accurate space resection. Once the feature points on the aerial image are matched with the orthoimage, the image coordinates of the feature points can be used to locate the corresponding real-world coordinates on the DSM. These real-world coordinates can then be used as GCPs for space resection, to estimate the UAV camera position and orientation. This estimation can be achieved using Eq. (3.3), which considers the image coordinates of the feature points and the corresponding real-world coordinates on the DSM. The camera position estimation results are shown in Figure 3.17.



(a) (b) (c)

Figure 3.17 Experiment of space resection using matched GCPs on DSM to estimate camera position and orientation: (a) aerial image sample; (b) matching feature points from (a) to DSM to obtain real-world coordinates of GCPs; (c) estimated camera positions (green dots) and orientations (white polygons). The blue dots are the actual camera positions for reference.

In this study, aerial images with varying surface structures such as buildings, roads, trees, and other features were selected to perform space resection and estimate the position and orientation of the camera. The original aerial images captured by the UAV camera are displayed in Figure 3.17(a), while Figure 3.17(b) shows the correct matches between the aerial images and the DSM cropped tiles. The real-world coordinates, obtained from the DSM cropped tile in WGS84 Mercator coordinates, were used as GCPs for space resection. The estimated camera position (blue dot) for each aerial image is visualised in Figure 3.17(c) and compared with the ground truth (green dot) on the corresponding orthoimage cropped tiles. The estimated orientation is visualised via homography using a white contour. The accuracy of the estimated camera position and its deviation from the actual position are presented in Table 3.4. The average RDE between the actual camera location obtained via GPS and the estimation results calculated through the space resection of the above three images was 15.8 m. The error may include other inevitable errors; for example, the ground truth from GPS is the location of the antenna on the UAV and not the real camera spatial coordinates.

Therefore, the position and pose of the camera sensor estimated via space resection from the feature points of the image plane will feature a slight deviation from the GPS location obtained using the UAV positioning system.

Table 3.4 Accuracy evaluation of the experiment depicted in Figure 3.17

Image	Ground Truth (m)			Results from Our Method (m)			ADE (m)
	Lat.	*Lon.*	*Height*	*Lat.*	*Lon.*	*Height*	
(1)	11199634.5	1572373.7	258.6	11199628.1	1572379.7	261.82	9.3
(2)	11199414.5	1572566.3	254.4	11199409.4	1572561.8	261.02	9.5
(3)	11199189.2	1572119.6	256.3	11199191.1	1572122.8	263.45	8.0

3.3.3 Integration of VO and Space Resection for Continuous Camera Pose Determination

The previous section presents VO as a method for estimating the position and orientation of a camera in real time. However, the resulting estimates were in relative camera coordinates, not absolute real-world coordinates. To address this limitation, space resection was used to calculate the absolute pose of the camera in real-world coordinates using known locations of features in the image and the corresponding GCPs. Collinearity equations were solved using image coordinates of the GCPs, to derive the absolute camera pose in real-world coordinates. This approach is beneficial for environments where GPS signals are weak or unavailable, such as indoor environments, urban canyons, or areas with dense foliage. Combining these two methods allows for the real-time estimation of camera position and orientation, regardless of external conditions.

Another limitation of VO is its over-reliance on camera images and features extracted from the image to estimate the camera pose. In such environments, the features may be difficult to detect or track, leading to errors in position estimation. Moreover, VO is prone to cumulative errors, in which the estimated position and orientation drift away from the actual position and orientation over time. To overcome these limitations, space

resection can be integrated with VO. Because space resection relies on GCPs to estimate pose using the collinearity equation, it can be used to provide a global control for transferring the VO results to the global absolute scale and correct the local segments in which VO fails or largely deviates from the real situation. The workflow of our proposed method for integrating VO and space resection is given in Figure 3.18.

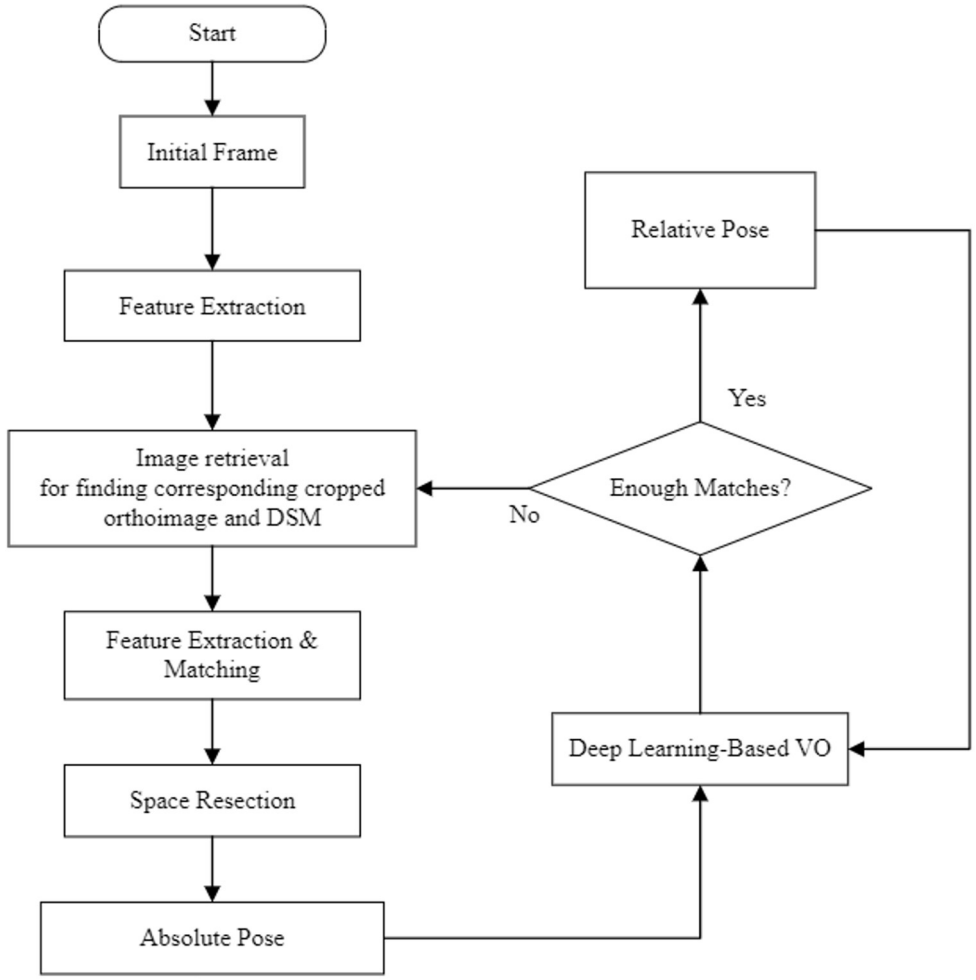


Figure 3.18 Overview of the integrated VO and space resection for camera pose determination.

The initial camera pose was calculated via space resection based on the collinearity equation using the first several frames of aerial images. SuperPoint extracts features $f(I_0)$ from aerial images I_0 and then passes them to the cross-view image matching approach for identifying the corresponding cropped orthoimage C_o and DSM C_d tiles in the pre-built database. The initial camera position R_0 and orientation t_0 were calculated using the features from aerial images $f(I_t)$ and GCPs from DSM cropped tiles.

The next step is a loop in which the camera position R_t and orientation t_t are estimated in real time via VO, according to the consecutive frames of aerial images I_t captured by the camera. The number of correct matches of feature points between consecutive frames was calculated to determine whether space resection was needed to estimate the camera position and orientation again. Space resection was applied to refine the estimated camera position and orientation if the number of correct matches fell below a certain threshold. After the refined camera position and orientation were obtained via space resection, VO was used for the subsequent frames. The following pseudocode provides a more detailed description of our proposed method.

Algorithm 1: Pseudocode for integrated VO

Input: frame $\boldsymbol{I_t}$, global features $\boldsymbol{f_g}$

Output: absolute orientation and position $[\mathbf{R}|\mathbf{t}]$

if $t = 0$ **then**

  $\boldsymbol{f(I_t)} \leftarrow \boldsymbol{SuperPoint(I_t)}$

  $\boldsymbol{C_o, C_d} \leftarrow \boldsymbol{Image\ Retriveal(f(I_t), f_g)}$

  $\boldsymbol{GCPs} \leftarrow \boldsymbol{Feature\ Matching(f(C_o), f(C_d))}$

  $[\mathbf{R_t}|\mathbf{t_t}] \leftarrow \boldsymbol{Space\ Resection(f(I_t), GCPs)}$

else loop

  $[\mathbf{R_t}|\mathbf{t_t}] \leftarrow \boldsymbol{VO}([\boldsymbol{f(I_t)}|\boldsymbol{f(I_{t-1})}], [\mathbf{R_{t-1}}|\mathbf{t_{t-1}}])$

  if $\boldsymbol{length(matches([f(I_t)|f(I_{t-1})]) < threshold)}$ **then**

    $[\mathbf{R_t}|\mathbf{t_t}] \leftarrow \boldsymbol{repeat\ step\ 3-5}$

In our integrated approach, space resection is adopted to provide an initial estimation of the camera position and orientation, and then VO is used for real-time estimation. The number of correct matches of feature points is used as a threshold to determine when to switch back to space resection for refinement. This approach can provide more accurate and reliable visual navigation and positioning by monitoring the number of correct matches and dynamically adjusting between the two methods.

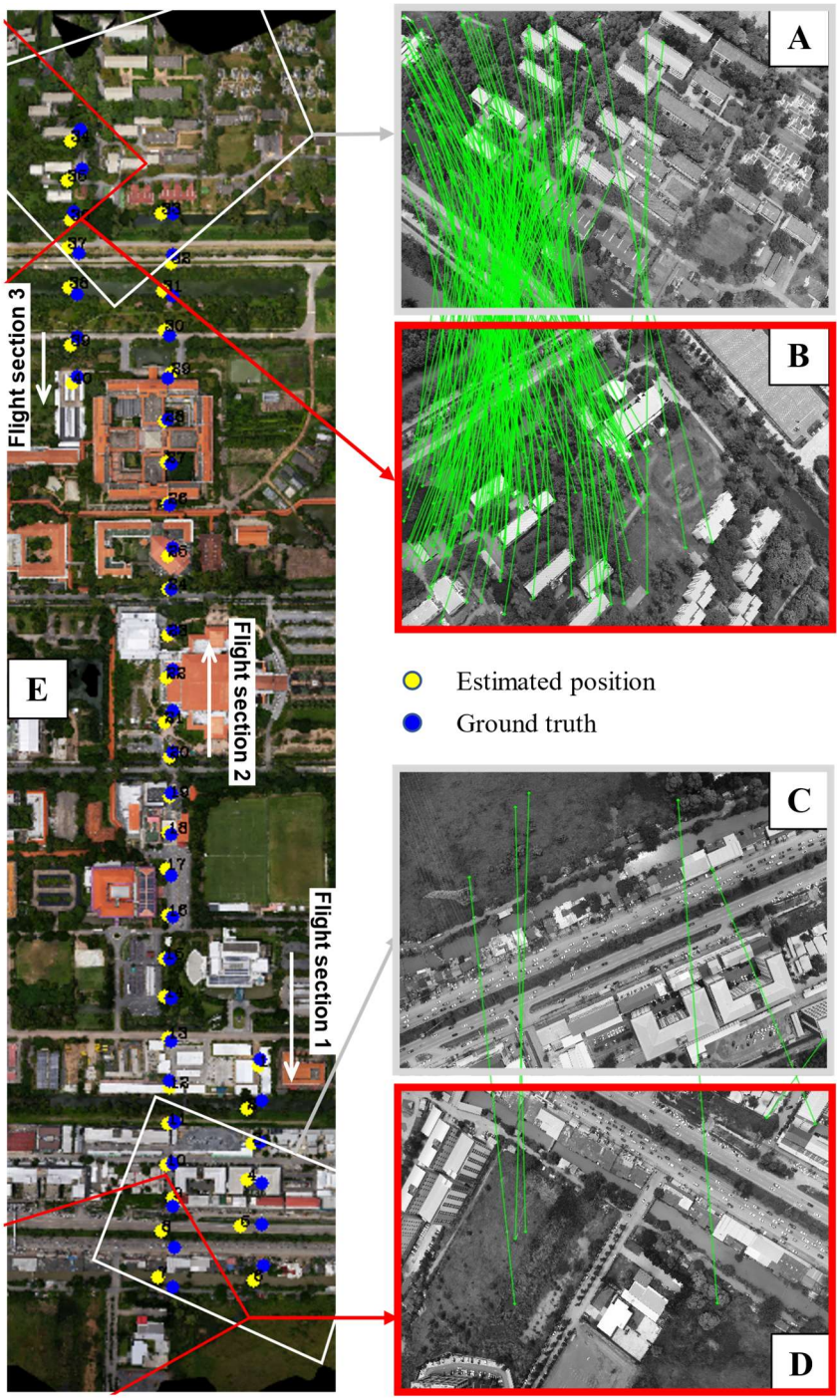


Figure 3.19 Experiment on integrating VO and space resection. The two frames depict the use of space resection to determine the camera pose of keyframes in situations where the UAV makes turns during the flight.

The experimental results of our VO–space resection integration approach for estimating the camera position and orientation in real-world coordinates are shown in Figure 3.19. Yellow dots represent the estimated results, while blue dots indicate the ground truth. In situations in which the correct matches between two consecutive frames are not

sufficient to estimate the camera's relative pose through VO, space resection determines the camera's absolute position and orientation as a constraint, enabling trajectory refinement. This situation typically occurs when the UAV finishes photographing along a planned flight route section and then turns around to start another planned flight section.

To demonstrate the integrated strategy, Figures 3.19(a) and (b) show a pair of consecutive frames captured by the UAV, while Figures 3.19(c) and (d) show the last image from the first section and the first image from the second section, respectively. When the features on these two consecutive frames are not sufficient for the VO algorithm to estimate the camera position and orientation, the features on both images are matched with the corresponding features on the cropped tile from the orthoimage and the DSM for space resection. The estimated camera position and orientation are shown as a yellow dot in Figure 3.19(e). The corresponding homography represented by a red and white polygon illustrates the estimated orientation. Similarly, Figures 3.19(a) and (b) show the last image of the second flight section and the first image of the third section, respectively. Although many features matched between the two images, the mismatches were significant; consequently, the estimated camera position and orientation exhibited significant deviation from the results based on the image shown in Figure 3.19(a). Therefore, space resection was applied again to estimate the camera position and orientation, which were then used to refine the camera pose. The refined results were used as the initial start point for the third flight route.

Integrating VO and space resection is a promising approach for the real-time estimation of the absolute camera pose in real-world coordinates. This method provides a reliable and accurate solution for mapping applications, particularly in scenarios with weak or unavailable GPS signals. The fusion of these two methods leverages the strengths of both techniques, allowing for precise localisation even in complex environments. Additionally, the fusion reduces the error accumulation associated with the standalone methods, resulting in more robust and accurate mapping results. This integrated method has many potential applications; for example, it can be applied in indoor environments, urban canyons, or areas with heavy foliage, where traditional localisation methods may not work effectively.

3.4 Implementation and Evaluation

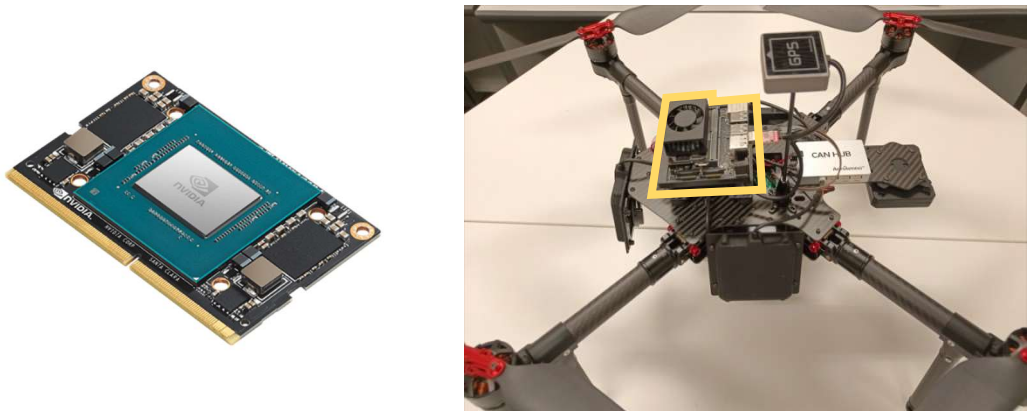
The onboard computer of a UAV serves as the central processing units that controls and manages the various components of the UAV, including the navigation and positioning systems. In particular, the computer is essential for the visual navigation and positioning of UAVs, as it utilises data from onboard sensors, such as cameras, to estimate the UAV position and orientation in real time. The onboard computer requires powerful image processing capabilities to extract features from the images captured by the onboard cameras. Recently, there have been significant advancements in the development of CPUs and GPUs for onboard computers in UAVs, with manufacturers designing specialised hardware to meet the specific needs of UAV applications. For example, Intel (Intel Corporation 2018) and ARM (Arm Limited 2023) have developed CPUs specifically for use in UAVs. These CPUs are designed to be small, lightweight, and energy-efficient, providing high-performance processing power. GPUs are also becoming increasingly crucial for onboard computers in UAVs, as they can increase the speed of complex image processing and machine learning algorithms, which is critical for object recognition and tracking tasks. NVIDIA is a leading manufacturer of GPUs for UAVs, with products such as the Jetson TX2 (NVIDIA 2017) and the Jetson Xavier (NVIDIA 2018), which offer high-performance computing in a compact form. These advancements in CPU and GPU technology enable UAVs to perform increasingly complex tasks, such as visual navigation and UAV positioning, with high accuracy and efficiency.

In this study, we implemented our proposed approach to estimate camera position and orientation on an onboard computer with a CPU and a GPU. The following sections introduce the overall design and evaluation results of our approach.

3.4.1 Onboard Platform and Algorithm Deployment

The NVIDIA Jetson Xavier NX is a high-performance system-on-module explicitly designed for use in embedded artificial intelligence applications, including UAVs. It features a 6-core NVIDIA Carmel ARM v8.2 64-bit CPU, a 384-core NVIDIA Volta GPU, and 8 GB of LPDDR4 RAM. In addition to its processing power, the Jetson

Xavier NX system supports multiple cameras, high-speed I/O, and hardware-accelerated video encoding and decoding.



(a) NVIDIA Jetson Xavier NX (b) Onboard computer installation

Figure 3.20 Schematic of onboard computer installation and assembling

An instance of NVIDIA Jetson Xavier NX chips is demonstrated in Figure 3.20(a), and Figure 3.20(b) illustrates the installation and assembling of the Jetson Xavier NX system on a UAV. The compact size, low power consumption, and high performance of the module make it an excellent choice for use in UAVs. The module's capacity to handle complex tasks makes it a perfect fit for visual navigation and positioning applications. Another significant feature of the module is its ability to perform multi-threaded processing using the CPU and GPU, allowing for the efficient processing of complex algorithms while minimising power consumption. For example, the CPU can be used for background tasks, while the GPU is used for real-time image processing and machine learning.

The allocation of the algorithms in our approach is shown in Figure 3.21. As the CPU is proficient in logic processing, we allocated VO and space resection algorithms to two threads for efficient parallel processing. The implementation of SuperPoint and SuperGlue algorithms relied on deep learning frameworks, and the Jetson Xavier NX GPU supported TensorRT (NVIDIA 2021) to accelerate graphic processing using deep learning algorithms.

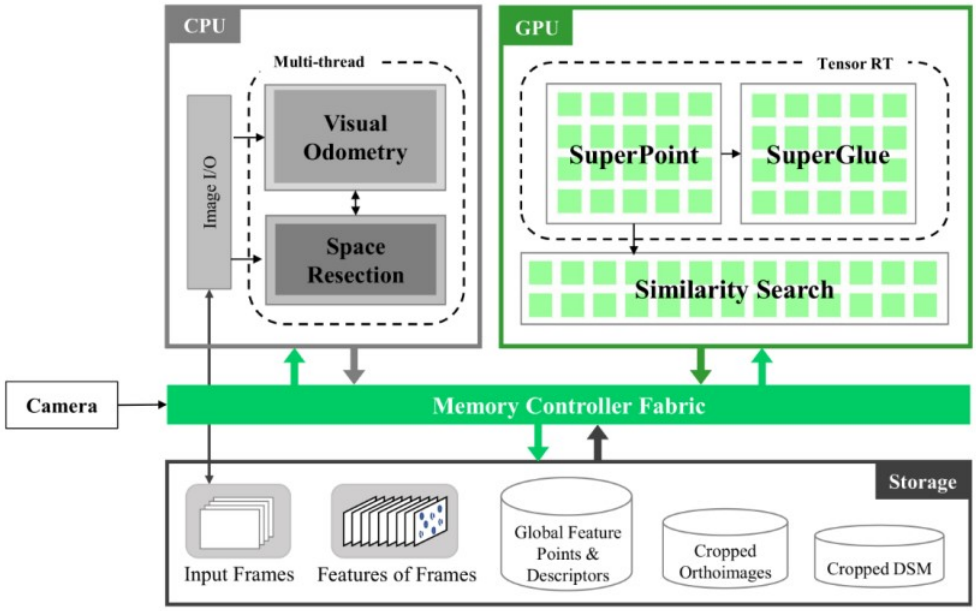


Figure 3.21 Algorithm deployment on GPU and CPU

The Jetson Xavier NX GPU has 384 CUDA cores and Tensor cores specifically designed for deep learning tasks. These cores enable the Jetson Xavier NX module to perform real-time inference based on deep neural networks, making it ideal for object detection and application tracking. TensorRT is an NVIDIA software library that optimises deep learning models for deployment on NVIDIA GPUs. It combines graph optimisation and layer fusion to optimise the computation graph of the neural network, resulting in shorter inference times and lower memory requirements. Furthermore, by allocating the VO and space resection algorithms to two separate threads, we leveraged the multi-threaded processing capabilities of the Jetson Xavier NX CPU, so that the tasks could be performed in parallel, significantly improving the system's overall performance for real-time processing.

The memory controller fabric (MCF) plays a crucial role in optimising the performance of the NVIDIA Jetson Xavier NX system. Its primary function is to manage memory access and bandwidth between various processing elements on the chip. The MCF facilitates efficient and rapid data transfer between the CPU, GPU, and other processing units. For instance, when image data are received from a camera, the USB controller first processes the data, which are then connected to the MCF. The MCF manages data transfer between the USB controller and the appropriate processing unit, such as the CPU or the GPU. If image processing is assigned to the CPU, the MCF ensures that the

image data are transferred from the memory to the CPU, which then processes the image data as necessary. If image processing is assigned to the GPU, the MCF transfers the image data from the memory to the GPU's GDDR6 memory. The GPU then performs the required image processing and machine learning tasks on the data. Moreover, the MCF dynamically allocates memory resources between the CPU and the GPU as needed, depending on the processing requirements of the image processing algorithm. This helps optimise the system's performance, ensuring that the system operates efficiently and accurately.

3.4.2 Evaluation with Aerial Images and Pre-built Database

In the experiment, the camera position and orientation were estimated through our approach using 439 aerial images. The algorithm was implemented on the Jetson Xavier NX system following the deployment shown in Figure 3.22. The resulting estimations and ground truth are presented in Figure 3.22. The camera position and orientation were obtained in WGS84 Mercator coordinates, with latitude, longitude, and height in metres. The estimated position was plotted on the orthoimage with a resolution of 1 m after coordination transformation from WGS84 Mercator to image coordinates. The root mean square error (RMSE) values in the horizontal (longitude and latitude) and vertical (height) directions were calculated (Table 3.5). The overall RMSE was 4.7 m, and the average execution time of our approach was 897.39 ms.

Table 3.5 Evaluation of trajectory estimation accuracy

	ADE (m)			**Rotation Error (°)**
	Horizontal	*Vertical*	*Overall*	
Mean	17.17	8.09	22.17	0.51
RMSE	4.14	14.24	4.7	0.33

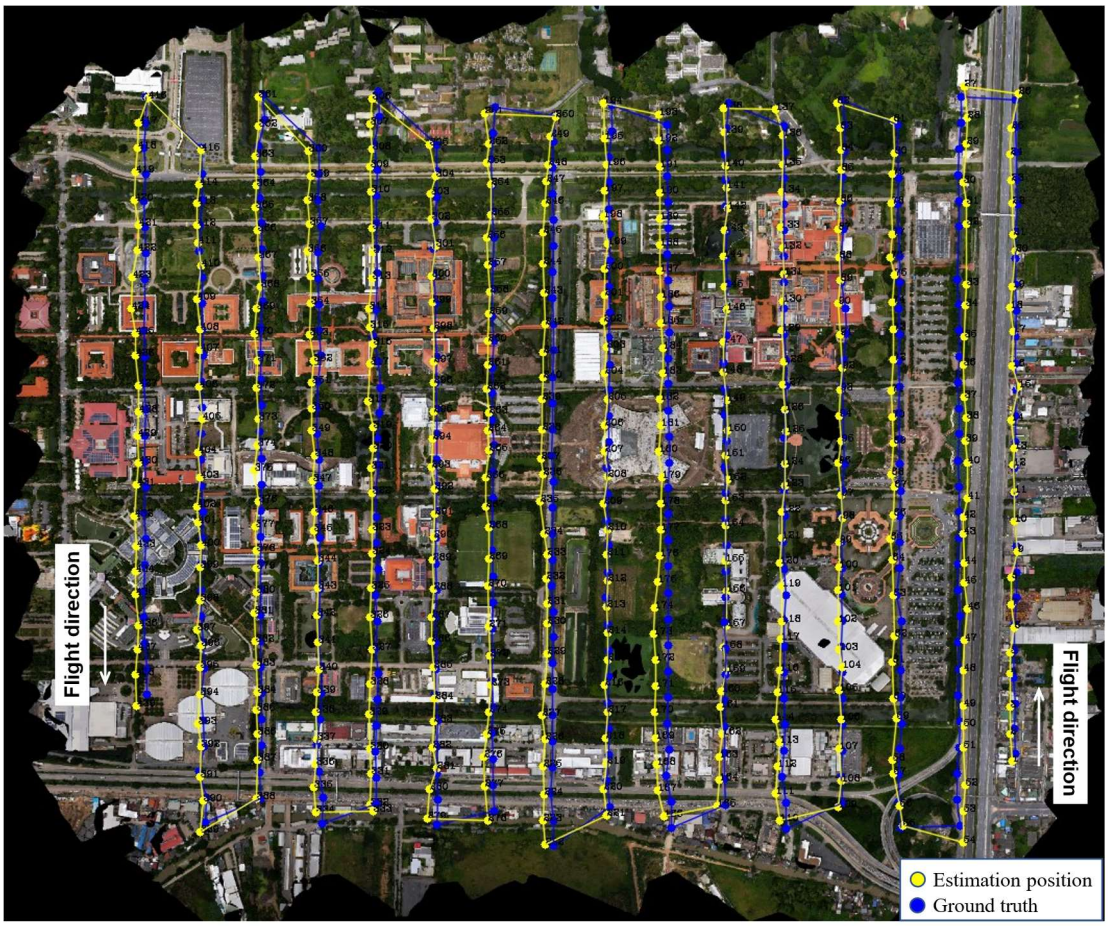
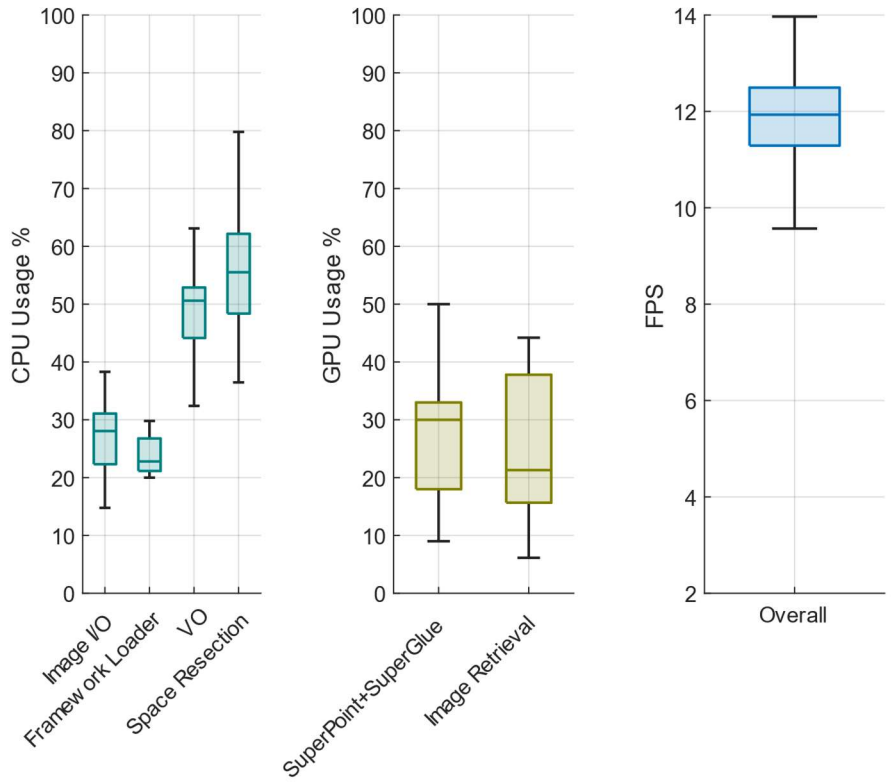


Figure 3.22 Comparison of trajectories estimated using our approach and the ground truth

Our approach achieved an RMSE of 4.7 m in ADE and 0.33° in rotation error. The execution time of the proposed approach includes several steps, namely the reading and writing of image data in memory, VO and space resection in the CPU, and the implementation of deep learning algorithms in the GPU. However, owing to the limited arithmetic power of the Jetson Xavier NX's camera ARM v8 processor and the complexity of the space resection algorithm, the input images were resized to 640 × 480 pixels to fully utilise the CPU and GPU resources. The CPU usage and GPU usage are given in Figure 3.23.



(a) CPU usage (b) GPU usage (c) FPS

Figure 3.23 The utilisation of hardware resources for each algorithm and the overall FPS of integrated VO

VO and space resection utilised most of the CPU resources for complex calculation, while the remaining CPU resources were allocated to image reading and writing and the loading of deep learning frameworks. Feature extraction and image retrieval consume a large amount of GPU resources, occasionally resulting in full utilisation, particularly when both algorithms are running simultaneously. The FPS fluctuated between 9.5 and 14, with an average of 12, depending on the execution time of each algorithm running in the CPU and the GPU. Our experiments showed that our approach can achieve near-real-time efficiency, demonstrating its potential for achieving real-time localisation. The results indicate that our proposed method can be optimised for improved performance; for example, the space resection algorithm and the utilisation of the hardware resources, such as the Tensor cores of the GPU, can be optimised.

Integrating VO and space resection techniques has immense potential for real-time UAV position estimation. With the increasing use of UAVs in diverse industries, this

integration can help overcome the challenges limiting the precise and reliable real-time estimation of UAV position and orientation, enabling more successful UAV applications. For instance, UAVs can be used for monitoring, inspection, search and rescue, and mapping applications. However, accurately estimating UAV position and orientation in real time is challenging. Integrating VO and space resection techniques offers a solution and could significantly enhance the accuracy and reliability of UAV applications.

Chapter 4 Real-Time Dense Image Matching Based on GPU Acceleration

4.1 Overview of Approaches

Dense disparity estimation is a challenging task owing to the high level of ambiguity often associated with real-world scenarios. Dense image matching algorithms, such as the SGM algorithm (Hirschmuller *et al.,* 2008), have been extensively used in various applications. Methods combining SGM with different types of local similarity metrics are insensitive to various types of noise and interference (e.g., illumination), efficiently estimate disparity on large untextured areas, and can produce favourable matching results (Feng *et al.,* 2019, Sinha *et al.,* 2014, Spangenberg *et al.,* 2014).

This section focuses on the development and implementation of a parallel-architecture SGM algorithm for real-time dense image matching on photogrammetric applications. A parallel-architecture method for accelerating dense image matching is proposed. This method can improve the efficiency of dense image matching in real-time scenarios. Its effectiveness is experimentally demonstrated in two applications. The first application involves the generation of real-time disparity maps using ground images obtained from a stereo camera. The second application involves real-time dense image matching to generate disparity maps using aerial images captured by UAVs.

This section includes the following:

(1) The SGM framework for dense image matching is introduced in Section 4.2. Issues related to processing efficiency (e.g., the matching cost [MC] and the selection of similarity measures) are discussed.

(2) The implementation of a parallel-architecture SGM with enhanced computational efficiency is presented. The parallel architecture significantly improves the overall processing efficiency of the algorithm and endows it with real-time processing applicability.

(3) To evaluate the real-time processing efficiency of the parallel-architecture SGM, we first used images taken by a stereo camera to assess the processing efficiency of real-time depth map generation and then tested the algorithm using large-scale aerial images collected from a UAV platform. The results of the real-time processing of the depth maps and the evaluation results are presented, and the results are analysed and discussed in Section 4.4.

4.2 SGM-Based Dense Image Matching and Efficiency Considerations

In Chapter 2, the state-of-the-art dense image matching algorithms are reviewed in terms of their strengths and weaknesses, and their performances in different scenarios are evaluated and compared. Our analysis and previous studies (Hermann *et al.,* 2011, Hirschmuller, 2005, Stentoumis *et al.,* 2015) indicate that SGM can generate robust matching results in stereo-scope scenarios.

SGM was first proposed by Hirschmuller (2005). It is a pixel-wise matching algorithm that combines the benefits of both global and local matching techniques. This dense image matching algorithm operates on a pair of images with known internal and external orientations and a defined epipolar geometry, meaning that the corresponding points are situated on the same horizontal line in the image. The objective of the algorithm is to minimise a global smoothness constraint by combining the MCs along independent one-dimensional paths across the image.

Scharstein *et al.* (2002) adopted a scanline approach to calculate a single global MC for each image line. This method was prone to streaking effects, as the optimal solution of each scan was not connected to the neighbouring scans. The SGM algorithm overcomes this limitation by symmetrically computing the pixel MC through multiple paths in the image. Given a known disparity value, the MCs obtained from each path are aggregated for each pixel and disparity value. The SGM algorithm then selects the pixel matching solution with the lowest cost, often through dynamic programming. This unique image matching approach leads to a more robust solution, eliminating the streaking effects present in previous methods. The MC vector is a 3D structure in which the first two

dimensions represent the pixels of the reference image, and the third dimension represents the pixels of the target image.

$$
\begin{aligned}
L'_r(p,d) = C(p,d) + \min\,(&L_r(p-r,d),\\
&L_r(p-r,d-1)+P_1,\\
&L_r(p-r,d+1)+P_1,\\
&\min_I L_r(p-r,i)+P_2) - \min_K L_r(p-r,k)
\end{aligned}
\tag{4.1}
$$

The cost $L'_r(p,d)$ of the pixel p at disparity d along the path direction r is defined in Eq. 4.1 as in Hirschmuller (2005). $C(p,d)$ represents the similarity cost between the pixels. The second part of the equation evaluates the regularity of the disparity field by introducing a penalty term P_1 to account for small changes. P_2 accounts for more significant changes in disparity relative to the previous point in the evaluated matching path. P_1 and P_2 allow for the description of curved surfaces and the preservation of disparity discontinuities, respectively. The last term of the equation plays a crucial role in mitigating the accumulated cost along the path. The subtraction of the minimum path cost of the preceding pixel from the overall cost allows for reducing the overall cost and ensures that the final result is minimised.

The SGM algorithm performs the minimisation operation via dynamic programming (Van Meerbergen *et al.,* 2002). To avoid streaking effects, SGM computes the optimisation by symmetrically combining multiple individual paths from all directions in the image. The algorithm generates the final disparity map by summing the costs of all paths r and identifying the disparity with the minimum cost for each pixel p in the image. The cost aggregation is expressed in Eq. 4.2. The minimum position is calculated by fitting a quadratic curve through the cost values of the neighbours' pixels for sub-pixel estimation of the final disparity solution.

$$
S(p,d) = \sum_r L_r(p,d) \tag{4.2}
$$

4.2.1 Matching Costs and Similarity Measurements

Area-based matching methods are fundamental techniques for identifying corresponding pixels. However, it is assumed that all pixels within a correlation window possess equivalent depth values, which is not necessarily valid in the presence of depth discontinuities or substantial perspective changes between matching images. Utilising small templates in the matching process may result in noisy and low-precision outcomes. In contrast, larger templates can lead to smoother results but also violate the constant-depth hypothesis, causing a loss of information on the shape details of small objects. The size of the correlation window influences the accuracy and completeness of the results and matching efficiency. While small correlation windows improve the level of object details, they may also provide an unreliable disparity estimation owing to the insufficient coverage of intensity variations. Conversely, a large window size hinders the ability of the matching algorithm to estimate sudden depth changes, leading to erroneous matching pairs and the generation of smoother surfaces (Kanade *et al.*, 1995). Image correlation in computer vision research is characterised by swiftness and low demand for runtime and memory occupancy; thus, it tends to be more widely utilised as a matching technique than alternatives such as least squares matching (Gruen *et al.*, 1988). Common parametric correlation measures used in photogrammetry and computer vision include the sum of SAD and NCC.

The SAD is calculated as the summation of the absolute differences between each pixel in an original image and the corresponding pixel in the matched image within a search window. In contrast, the sum of squared differences (SSD) is calculated as the summation of the squares of the differences between the same pixels. The summations are optimised via the winner-take-all (WTA) strategy (Kanade *et al.*, 1995). The SAD and SSD are expressed as

$$SAD = \sum_{i}\sum_{j}|f(i,j) - g(i,j)| \tag{4.3}$$

$$SSD = \sum_{I}\sum_{J}\big(f(i,j) - g(i,j)\big)^2 \tag{4.4}$$

where the window f is centred in the (x, y) position on the master image, and the corresponding same-size window on the slave image g is shifted by $(\Delta x, \Delta y)$.

The NCC is more complex than both the SAD and SSD; however, it is invariant to linear transformations in the image amplitude. Normalising feature vectors to unit length allows the similarity measure between the features to become independent of radiometric changes (Yoo *et al.*, 2009). The NCC identifies matches of a reference template $f(j, i)$ of size $m \times n$ in a scene image $g(x, y)$ of size $M \times N$, and is defined as

$$\rho(i,j) = \frac{\sum_i \sum_j \left[\left(f(j,i) - \bar{f}\right) \cdot \left(g(j+\Delta x, i+\Delta y) - \bar{g}\right)\right]}{\sqrt{\sum_i \sum_j \left[\left(f(j,i) - \bar{f}\right)^2 \cdot \left(g(j+\Delta x, i+\Delta y) - \bar{g}\right)^2\right]}} \tag{4.5}$$

where $\bar{f}$ and $\bar{g}$ represent the corresponding sample means. A unitary value of the NCC coefficient indicates a perfect matching window.

4.2.2 Census Transform

The census transform (CT) (Zabih *et al.*, 1994) is an area-based solution to the problem of correspondence between images. The CT is a non-parametric description of the local spatial structure. It compares the intensity values of each pixel within window W with that of the central pixel according to the Hamming distance. The intensity comparison between the master and central slave pixel p of the window returns a Boolean value of 1 if the pixel intensity is less than the intensity of the central pixel, and 0 otherwise; that is,

$$R(p) = \otimes_{p'} \xi(I(p'), I(p)) \qquad \begin{matrix} \xi(i,j) = 1, i < j \\ \xi(i,j) = 0, i > j \end{matrix} \tag{4.6}$$

where $\otimes$ represents the concatenation and $p' \in W$. According to Hirschmuller *et al.* (2008), both hierarchical mutual information and CT features provide similarly high-quality results, with CT being less computationally demanding. However, recent advancements in cost functions based on neural networks have been demonstrated to outperform CT (Zbontar *et al.*, 2015) but increase computational requirements.

4.2.3 Efficiency Considerations

The accuracy and computational efficiency of real-time dense image matching using SGM depend on several critical factors, including the choice of similarity measure, adaptive window size, and search range. The similarity measure is critical in establishing correspondences between the left- and right-view images and estimating the disparity. The adaptive window size is used to limit the search space for matching pixels, which helps to reduce the number of computations required. Additionally, the search range defines the minimum and maximum disparity values. Increasing the maximum disparity value allows the algorithm to estimate disparities over a wider range of pixel distances but increases the computational requirements and the noise levels in the resulting disparity map.

Image similarity measures also ensure that the disparities can be accurately estimated in the presence of noise, occlusions, and other image artefacts. Among the widely used image similarity measures, the SAD is computationally efficient and straightforward, making it suitable for images with small changes in content, such as close-range images. The SSD and NCC are more robust to illumination and contrast variations, making them more accurate than the SAD for images with large changes in content, such as large-scale images. Owing to their complex calculation steps, the SSD and NCC require more computational resources than the SAD. Once the MC volume is created, it is processed via cost-volume smoothing (using a series of filters) through a left–right consistency check to reduce the number of false matches. The left–right consistency check ensures that the disparity value for a pixel in one image is the same as that for the corresponding pixel in the other image. Finally, the optimal path is computed according to the cost volume, which provides the correspondences between the pixels of the two images. The optimal path is calculated using a dynamic programming algorithm such as the WTA strategy or graph cutting. The WTA strategy is a simple approach for computing the optimal path (Scharstein *et al.* 2002). It involves finding the minimum value of the cost volume along the disparity dimension for each reference image pixel. The algorithm finds the minimum cost with overall disparities for each pixel of the reference image, and assigns the corresponding disparity value to the pixel.

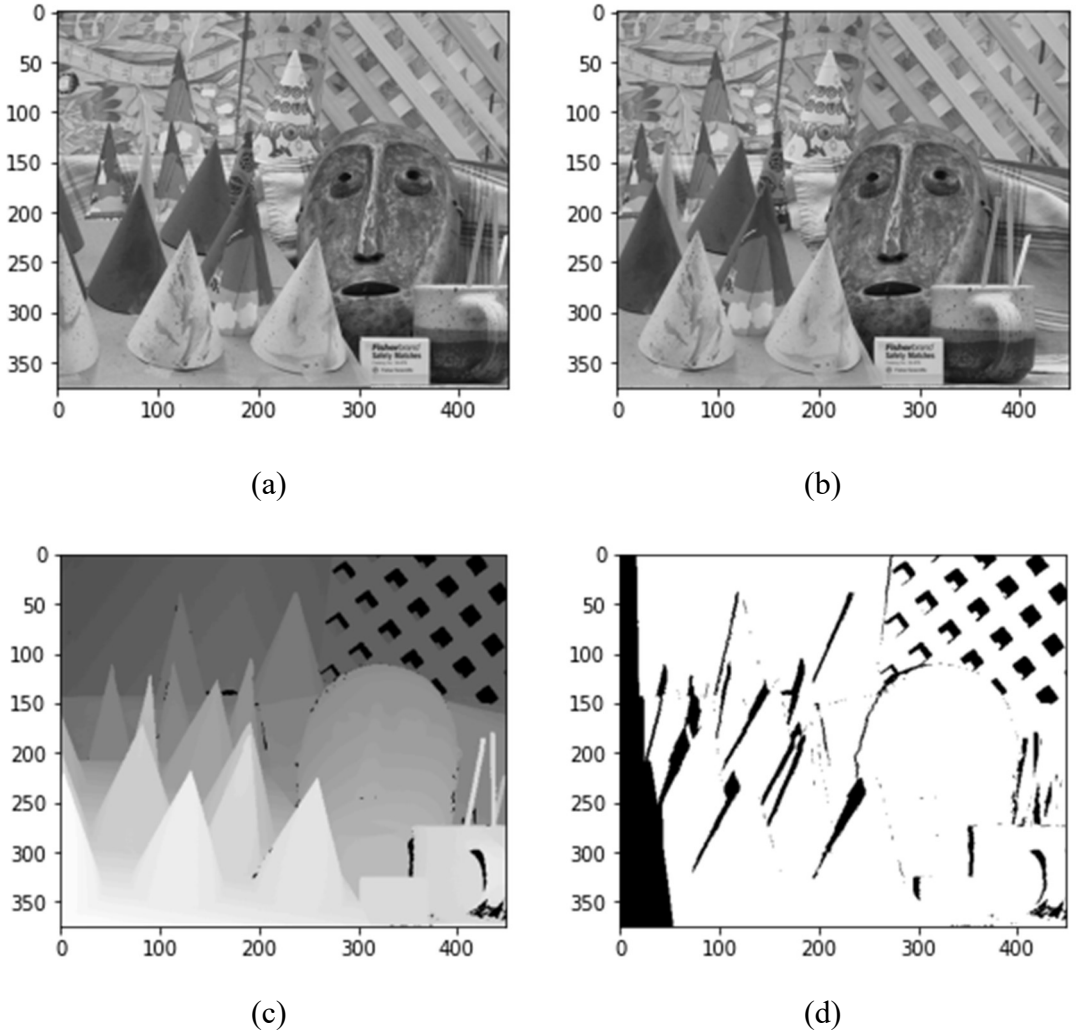
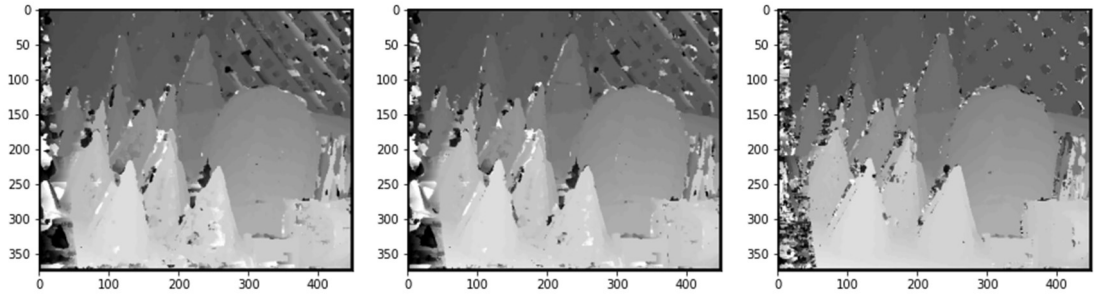


Figure 4.1 Dataset for similarity measurement evaluation: (a) left-view image; (b) right-view image; (c) ground-truth disparities; (d) invalid disparity mask.

The accuracy and computational efficiency of various similarity measures combined with WTA were assessed using the standard benchmark dataset for stereo vision proposed by Scharstein *et al.* (2003). The benchmark dataset included a pair of left- and right-view images, each with a 450 × 375 pixels resolution (Figure 4.1). Ground-truth disparities were provided for accuracy evaluation, and a grey-level mask was used to indicate pixels with invalid disparities; the invalid disparities were encoded as 0 and 1, respectively. The similarity measures were evaluated through the comparison of the estimated disparities with the ground-truth disparities, and the provided mask was used to exclude pixels with invalid disparities.



(a) (b) (c)

Figure 4.2 Disparity results obtained using different similarity measures combined with WTA: (a) SAD with WTA; (b) SSD with WTA; (c) NCC with WTA.

Table 4.1 Accuracy and efficiency results of different similarity measures

	SAD	**SSD**	**NCC**
Accuracy (%)	86.4	88.1	91.1
Processing time (s)	1.76	2.28	4.13

The disparity results obtained using various similarity measures are shown in Figure 4.2. The accuracy was evaluated using the following equation:

$$AccX = \frac{\sum_i^n \sum_j^n d(i,j)}{\sum_i^m \sum_j^m g(i,j)} \times mask \tag{4.7}$$

where $d(i, j)$ represents the disparity generated using various similarity measures, and $g(i, j)$ is the corresponding ground-truth disparity. The mask was applied to filter out all unknown disparities, and the accuracy was calculated using valid pixels. Compared with the other similarity measures, the NCC provided more robust disparity results, with >90% accuracy. As indicated in Table 4.1, the NCC provided high-precision disparity; however, the computational cost of ~4 s per pair of stereo images was higher than those of the other two measures. The SAD and SSD provided similar accuracies in the disparity results: 86.4% and 88.1%, respectively, but the processing time of the SAD was 1.76 s, much less than that of the SSD. Because the processing efficiency and simplicity of the algorithm are essential for real-time stereo matching, the SAD is preferred for similarity measures, as it requires less computation time than the SSD or NCC and is suitable for real-time stereo matching.

To achieve real-time stereo matching, it is necessary to carefully choose the adaptive window size for the search space of each pixel and the search range between a pair of stereo images to limit computational cost. The evaluation of the window sizes and the minimum and maximum disparities adopted in SGM was assessed using the same dataset. In the experiment, the minimum variance was set to 0 to reduce the influence of redundant variables on the evaluation results. The accuracy and efficiency of disparity generation under different window sizes and a search range of diverse maximum disparities are shown in Figure 4.3. The results in Figure 4.3(a) were obtained using a window size of 5 × 5 pixels. A larger maximum disparity led to a more robust disparity estimation. At a maximum disparity of over 64 pixels, the disparity generation accuracy became stable and remained above 90%. Similar results were observed at various window sizes of 9 × 9, 13 × 13, and 21 × 21 pixels (Figures 4.3b, c, and d).

Figure 4.3 also indicates that the processing time for disparity estimation increased with increasing window radius under a constant maximum disparity. For instance, the time for disparity generation under a maximum disparity of 64 pixels and a window size of 5 × 5 pixels was ~1.5 s (Figure 4.3a). However, disparity estimation under a window size of 21 × 21 pixels took ~13 s (Figure 4.3d).

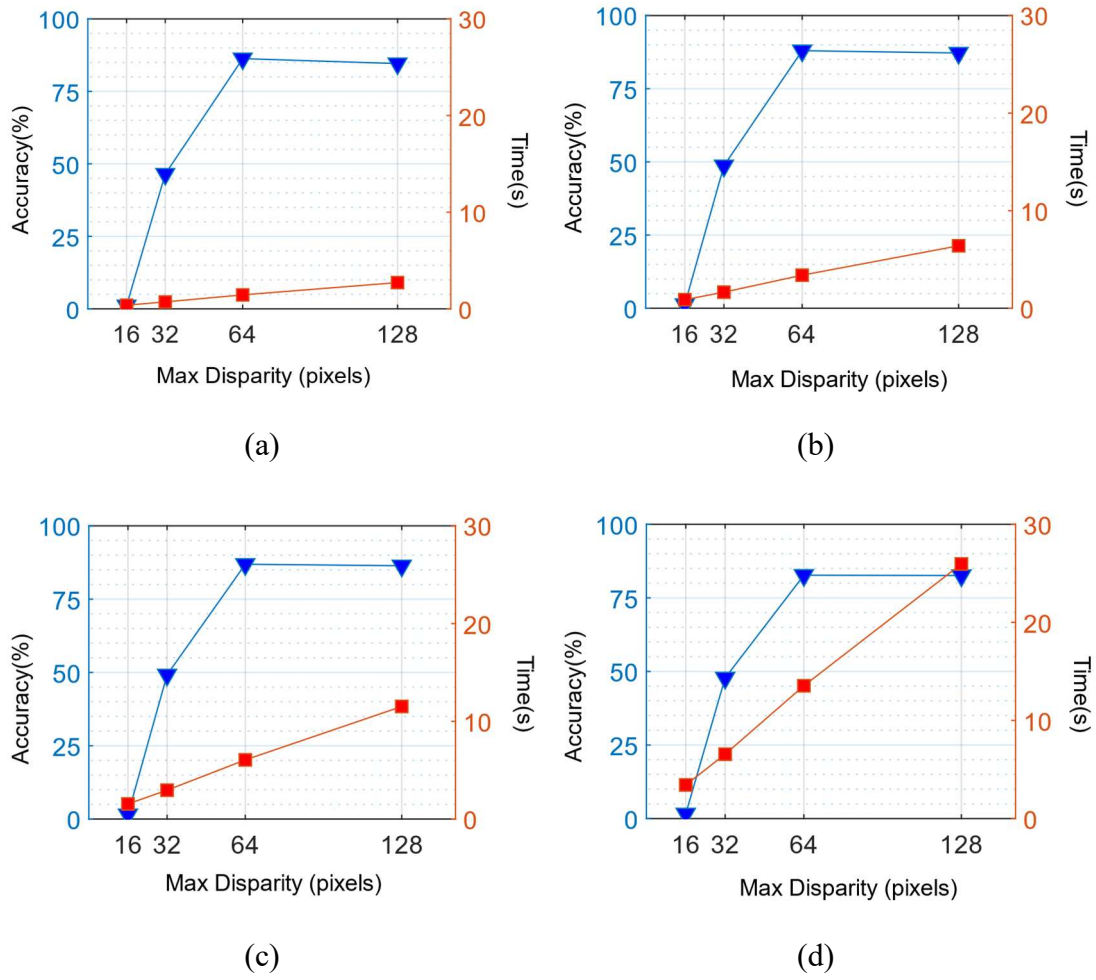


Figure 4.3 Disparity estimation accuracy and processing time evaluation results under the same maximum disparity and different window sizes of (a) 5 × 5, (b) 9 × 9, (c) 13 × 13, and (d) 21 × 21 pixels.

In conclusion, to achieve real-time dense image matching using SGM, the following are recommended: 1) The SAD should be used as the similarity measure for close-range images owing to its lower complexity and more robust disparity estimation ability compared with other similarity measures. 2) An adaptive window size of 5 × 5 pixels and a search range with 64 pixels as the maximum disparity should be adopted, as these parameters provided accurate and fast disparity estimation in the conducted experiment. 3) The smoothing of MC for each path should be considered as a filtering procedure to reduce the number of false matches. 4) Dynamic programming techniques such as WTA should be employed to calculate the optimal path using MC, thereby determining the relationship between the pixels of the two images.

74.3 GPU-Accelerated Dense Image Matching

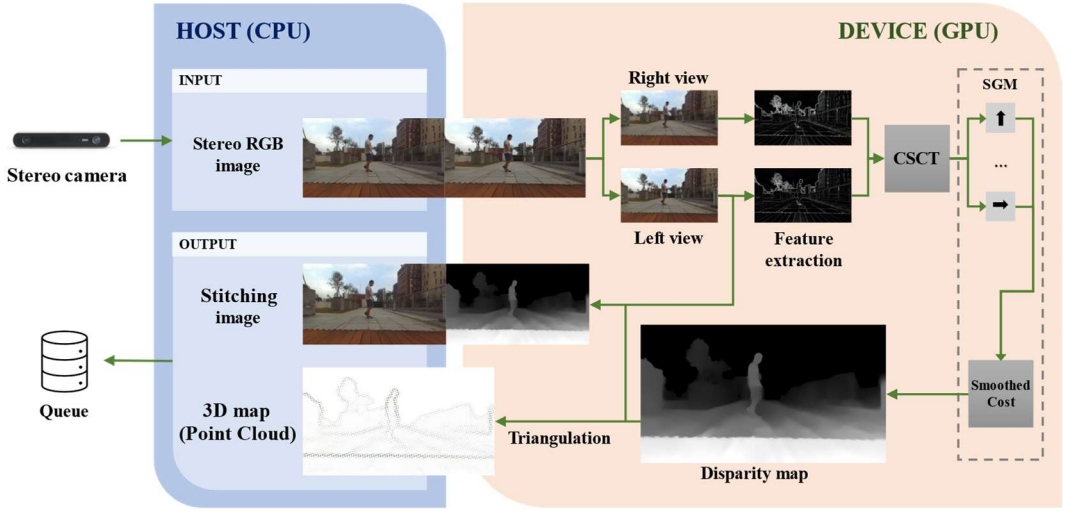


Figure 4.4 GPU-accelerated procedure of dense image matching and 3D map generation.

A GPU-accelerated SGM method was developed to obtain a disparity map for real-time stereo image estimation. As shown in Figure 4.4, each frame of the stereo images in the rectified pipeline was captured by preliminary calibrated cameras as a side-by-side (SBS) image and saved in the host memory. The GPU device copied this image from the host memory space and split it into left-view and right-view images in preparation for dense image matching by SGM. The CT features were extracted from the two images and used for a similarity comparison to generate a local-matching cost for each pixel and potential disparity. SGM was then used to aggregate a smoothing cost that considers the similarity of the neighbouring points and disparities along different paths to reduce errors. In this system, the number of paths was set to four to reduce computational consumption while ensuring the quality and effectiveness of real-time processing. The disparity of each pixel was computed, and a 3 × 3 median filter was applied to remove outliers. The resulting disparity image was copied back to the local host memory and stitched with the left-view image to form a new image array, which was then saved in the queue for visualisation.

4.3.1 GPU Architecture and Performance

GPUs are massively parallel architectures containing tens of streaming multiprocessors (SMs), and vector computer operations are highly utilised and pipelined in SMs to optimise computational efficiency. The compute unified device architecture (CUDA) programming model (NVIDIA *et al.,* 2020) allows for defining a massive number of threads deployed in SMs of the same program code. SGM was coded using a two-level identifier in CUDA to specialise each thread for disparity estimation. The code in this study was deployed following the method proposed by Hernamdez-Juarez *et al.* (2016).

The CUDA programming model provides a platform for executing parallel programs in a GPU environment. It enables the creation of numerous concurrent execution instances, commonly referred to as threads, which run the same program code. The threads are differentiated according to their unique two-level identifier, <*ThrId, CTAid*>, which serves as a specialisation mechanism for assigning particular data and functions to each thread. A cooperative thread array (CTA) is a group of threads that simultaneously execute the same *CTAid* within the same SMs and can share a fast, limited memory space. *Warps* are groups of threads with consecutive *ThrIds* within the same CTA, and they are compiled by a compiler into vector instructions, allowing for the execution of the threads in a lockstep synchronous manner. Warps belonging to the same CTA can be synchronised according to explicit barrier instruction. Each thread has its own private local memory space, commonly assigned to registers by the compiler. Additionally, a large space of global memory is accessible to all execution instances, providing a shared public space for data and functions. The global memory is mapped into a large-capacity device memory with a long latency and is optimised using a two-level hierarchy of cache memories.

The parallelisation scheme of an algorithm and the data layout determine the available parallelism at the instruction and thread levels, which is crucial for achieving the total resource usage and the memory access pattern. To achieve efficient memory performance, the GPU requires that the set of addresses generated by a warp correspond to consecutive positions that can be coalesced into a single, wider memory transaction. As the device memory bandwidth can be a performance bottleneck, an efficient CUDA

code should be used to promote data reuse on shared memory and registers. The design of the CUDA programming model allows for the efficient use of the GPU resources through the creation of numerous concurrent execution instances and the utilisation of a fast but limited memory space of the SMs. The use of warps and CTAs allows for efficient thread execution and synchronisation and enhances memory performance by promoting data reuse and coalescing memory transactions. CUDA provides a powerful and efficient platform for parallel computing on GPUs.

4.3.2 GPU-Based Centre-Symmetric CT and Matching Cost Computation

As mentioned in the previous section, CT is a technique for encoding the similarities between the values of pixels in a window around a central pixel. A global two-dimensional energy minimisation problem involving non-unique or wrong correspondences caused by low texture and ambiguity can feature consistency constraints. SGM approximates the global solution by solving a one-dimensional minimisation problem along several independent paths across the image (Hirschmuller *et al.,* 2008). There are typically four or eight paths. An eight-path direction was used in this work (Figure 4.5). For each path direction, image point, and disparity, SGM calculates a cost by considering the cost of neighbouring points and disparities. The other path directions along the diagonal paths between pixel P and P' are not immediately available for cost calculation, resulting in complex memory access patterns. The number of paths used in the SGM algorithm plays a crucial role in determining the final outcome. It affects both the quality and accuracy of the results. A higher number of paths will result in a more accurate solution but increases computational time. A lower number of paths will result in faster computation but reduces accuracy.

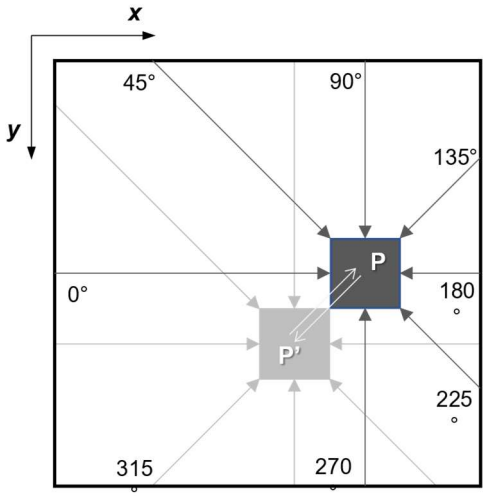


Figure 4.5 Orientations of eight paths for pixel *P*, shown in black.

After careful examination, a centre-symmetric CT (CSCT) configuration with a 9 × 7 window size was selected, providing a compact representation while maintaining similar accuracy (Spangenberg *et al.*, 2013). The similarity between two pixels was calculated using the Hamming distance of the CSCT bit-vector features. This feature is robust in outdoor environments with uncontrolled lighting and in the presence of calibration errors. The CT feature is invariant to local intensity changes and tolerant to outliers, as it compares the neighbouring pixels with each other. An incorrect value only modifies a single bit, which will not considerably affect the overall similarity score.

To accelerate the CSCT and MC computation using GPUs, CTA-parallel schemes are proposed in this work. The CSCT feature encodes the similarities between the values of pixels in a window around a central pixel. The CSCT feature uses a 9 × 7 window size and concatenates the comparisons of 31 pairs of pixels into a bit-vector feature. Eq. 4.8 defines the CSCT, where $\otimes$ is bit-wise concatenation; $I(x, y)$ is the value of pixel (x,y) in the input image; and $s(u,v)$ is 1 if $u \geq v$, or 0 otherwise. This bit-vector feature calculates the MC between a pixel (x, y) in the base image and each potentially corresponding pixel in the matched image at a specific disparity d. MC is calculated using Eq. 4.9, where $\oplus$ denotes bit-wise exclusive-or, and the bit count (B) is the number of bits set to 1.

$$CSCT_{9,7}(I,x,y) = \otimes \begin{cases} \otimes_{i=1}^{4} \otimes_{j=-3}^{3} s(I(x+i,y+j), I(x-i,y-j)) \\ \otimes_{j=1}^{3} s(I(x,y+j), I(x,y-j)) \end{cases}, \quad (4.8)$$

$$MC(x,y,d) = B(CSCT_{9,7}(I_{base},x,y) \oplus CSCT_{9,7}(I_{match},x-d.y). \quad (4.9)$$

Figures 4.6 and 4.7 show the pipeline of CSCT and the MC processing steps, respectively. In Figure 4.6, (*H, W*) denotes the dimensions of an input image. A 2D-tiled data access pattern using shared memory reduces the total number of global data accesses. To enhance the processing efficiency, computation and image input in the system are performed using the *32-bit integer* data type. To maintain data coherence and alignment for all threads in CUDA, a conversion from integer to *uchar4* data type (the native data type in the GPU) is conducted just before result writing. The cost values are only stored in the shared memory every four iterations, and the in-built packed *uchar4* data type is used to minimise memory bandwidth requirements. This leads to a compact data layout of the cost space, which is not significant if subsequent kernels employ the same parallelisation scheme and thus maintain a consistent data layout. This straightforward and embarrassingly parallel design allows each thread in the GPU to read its input values directly from the global memory, thereby optimising data reuse.

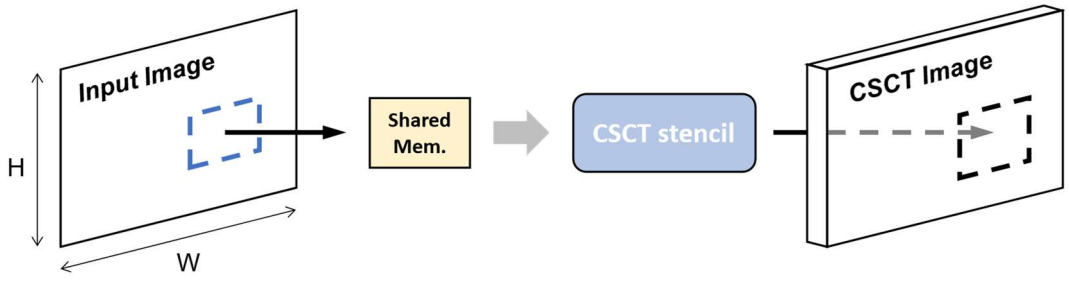


Figure 4.6 CSCT: 2D-tiled CTA-parallel scheme.

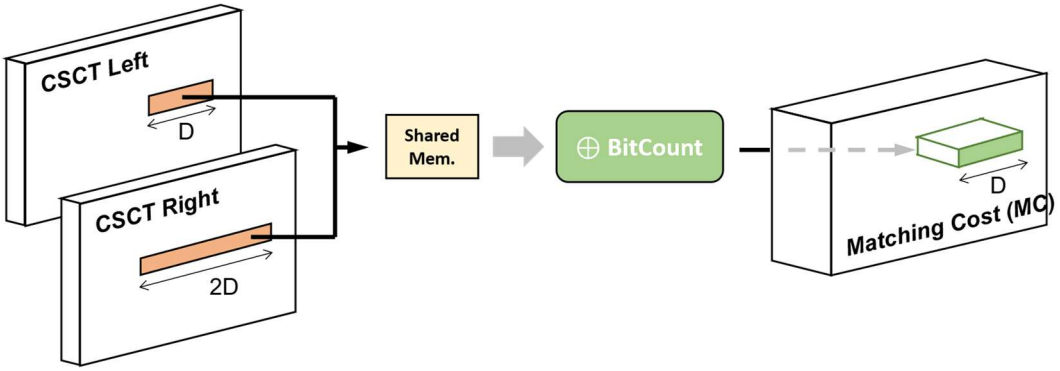


Figure 4.7 MC: 1D-tiled CTA-parallel scheme.

A 1D-tiled parallel scheme for MC calculation using Eq. 4.9 is shown in Figure 4.7. *D* represents the maximum disparity. Each thread synchronously calculates the disparity levels of a group of *D* neighbouring pixels in *D* threads. Pixels from the left and right CSCTs are aligned and coalesced over the *D* threads. Additionally, choosing *D* as a multiple of the warp size results in an always-aligned memory access. The MC calculation follows an optimisation approach similar to CSCT calculation by providing inherently aligned memory access, high data reuse, and efficient arithmetic pipeline usage.

$$L_r(x,y,d) = MC(x,y,d) + \min \begin{cases} L_r(x-r_x, y-r_y, d) \\ L_r(x-r_x, y-r_y, d-1) + P_1 \\ L_r(x-r_x, y-r_y, d+1) + P_1 \\ min_i L_r(x-r_x, y-r_y, i) + P_2 \end{cases} - min_k L_r(x-r_x, y-r_y, k), \quad (4.10)$$

$$D(x,y) = min_d \sum_r L_r(x,y,d). \quad (4.11)$$

SGM solves one-dimensional minimisation problems by considering different paths, represented by the vector $r = (r_x, r_y)$, and using a dynamic programming algorithm to find the optimal path. The method uses a matrix called L_r, which contains the smoothed aggregated costs for each path *r*. The smoothing costs are calculated as in Eq. 4.10, which has three terms. The first term is the original MC, denoted $MC(x, y, d)$, which is the cost of matching the current pixel to the corresponding pixel in the other image. The second term is the minimum cost of the disparities corresponding to the previous pixel $(x - r_x, y - r_y)$. This term includes penalties for small disparity changes P_1 and larger disparity discontinuities P_2. P_1 is designed to detect slanted and curved surfaces, as they are more likely to feature small disparity changes. P_2 smoothens the results and makes it more difficult for abrupt changes to occur. This is important as abrupt changes may result in false depth perception. The last term of Eq. 4.10 ensures the boundedness of the aggregated costs, which prevents unrealistic results and helps the algorithm to converge to the optimal solution. This term also limits error accumulation in the calculation, thereby increasing the accuracy of the final result. Eq. 4.10 demonstrates the use of the WTA strategy, in which the matrices L_r are added to obtain the final cost, and then the disparity corresponding to the minimum cost is selected.

Eq. 4.10 is used to calculate the costs of different paths in the SGM method, but it also creates a recurrent dependence that prevents the parallel processing of pixels in the same path direction. However, parallelism can still be exploited in other ways. Specifically, it can be exploited in the direction perpendicular to the path, in the disparity dimension, and for each of the computed path directions. Our proposed solution leverages all of the available parallelism by creating a CTA for each slice in the aggregated cost matrix along each particular path direction. This allows for the parallel processing of the slices in the disparity dimension and the parallel processing of each path direction. This means that instead of processing the pixels sequentially, our proposed strategy allows for the simultaneous processing of multiple pixels, resulting in a significant increase in computational efficiency.

In summary, to minimise the number of memory accesses during cost aggregation and disparity computation according to Eq. 4.11, a CTA-based parallel scheme is proposed. The algorithm uses a CTA-based parallel scheme in which each CTA thread first adds the costs corresponding to a given disparity level for all path directions. Then, the CTA threads cooperate to identify the disparity level with the minimum cost. This approach avoids the writing and reading of the final cost matrix and increases the computational speed. Additionally, it allows for parallel processing of the algorithm, enhancing its efficiency and suitability for real-time computer vision applications.

4.3.3 Optimisation of Disparity Map Generation and Parallel Computing

SGM uses penalty parameters to handle different situations, such as depth continuity and discontinuity. By adjusting these parameters, the algorithm can effectively handle depth discontinuities, reduce breakage, and provide good disparity smoothing. However, this method is computationally intensive, and if the input left and right image pairs from a stereo camera are not ideal, the final disparity map may contain small black squares. These squares may be due to the failure of the algorithm to match the corresponding pixels in the left and right images. This is usually caused by errors in the images, such as noise, blur, or a lack of texture, which makes it difficult for the algorithm to identify correspondences.

To address the issue of noise and artefacts in the disparity map, a simple peak filter was applied after the SGM cost aggregation step. The peak filter (Hirschmuller, 2007) operates by computing the maximum disparity value within a local window around each pixel in the disparity map. The resulting maximum disparity value is then assigned to the central pixel in the window, and this process is repeated for all pixels in the disparity map. Despite the operational simplicity and effectiveness of the process in reducing noise and artefacts in the disparity map, the peak filter features several limitations. For example, the filter may blur edges and details in the scene, particularly in regions with high contrast or texture. Moreover, the peak filter may be unable to handle occlusions and textureless regions as effectively as other smoothing filters, such as the weighted least squares (WLS) filter. These limitations suggest that further modification of the peak filter is necessary to improve its performance in challenging scenarios.

The WLS filter can effectively reduce noise and artefacts in the disparity maps without significantly blurring the edges or details of the scene. It achieves this by assigning higher weights to pixels with similar intensity values and lower weights to pixels with dissimilar intensity values. Thus, the WLS filter can preserve the sharpness and details of the scene while effectively reducing noise and smoothing the disparity maps. Moreover, the WLS filter can handle occlusions and textureless regions, which are common challenges in stereo image matching tasks. The filter can detect and handle these regions by assigning higher weights to pixels with reliable disparity values and lower weights to those with unreliable disparity values. This helps to minimise the effect of occlusions and textureless regions on the final disparity maps.

Figure 4.8 compares the experimental results of disparity maps generated using the peak filter and the WLS filter. WLS filtering was more robust and effective in improving the quality of the disparity maps. As shown in Figure 4.8(d), the peak values of disparity filtered out by the peak filter, indicated by the black gaps in Figure 4.8(c), were significantly reduced after the application of WLS filtering.

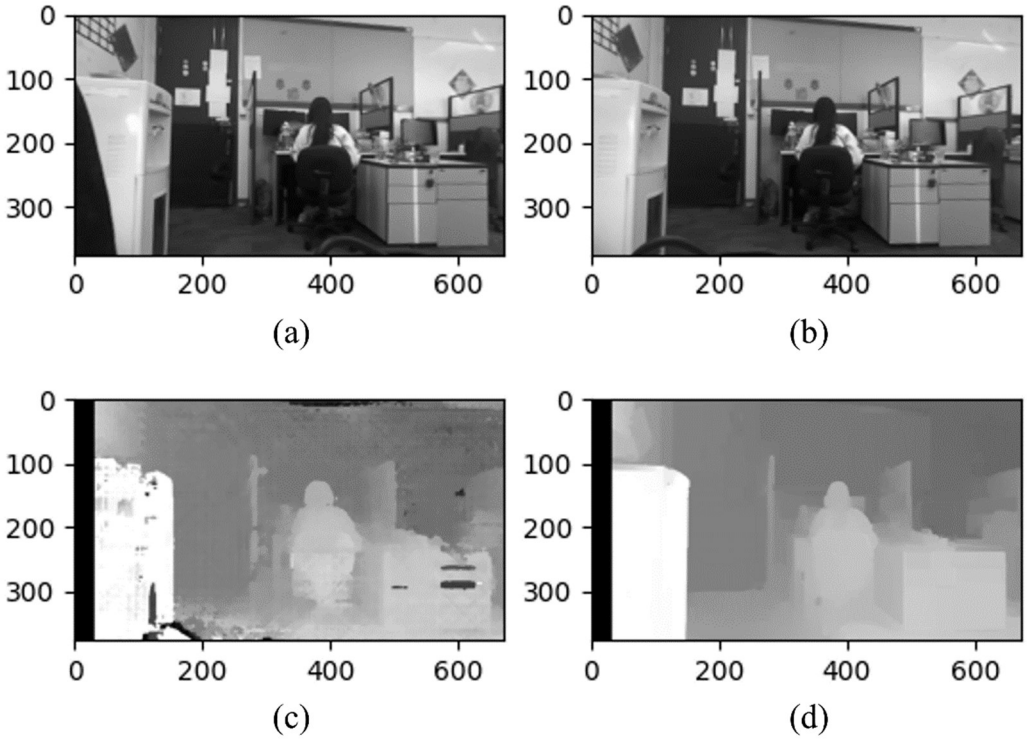


Figure 4.8 Comparison of disparity maps generated using the peak filter and the WLS filter: (a) left-view image and (b) right-view image; (c) disparity map after peak filtering; (d) disparity map after WLS filtering.

The deployment of the WLS filter as the smoothing filter will increase the amount of data accessed from memory and therefore the computational cost of SGM. To avoid this situation, we adopted the GPU parallel scheme optimisation procedure proposed by Hernandez-Juarez *et al.* (2016); however, the parallel scheme was modified such that a single warp now performs the task previously assigned to the CTA, and the modified method is referred to as CTA-to-warp conversion. This modification eliminates the need for expensive synchronisation operations, enables faster register-to-register communication through special shuffle instructions, and reduces the number of instructions required while increasing instruction-level parallelism. However, the strategy reduces thread-level parallelism. Overall, the CTA-to-warp conversion increases processing efficiency and is thus useful for real-time applications that require high performance.

4.4 Implementation and Evaluation

4.4.1 Hardware Configuration and Data Acquisition

(a) (b)

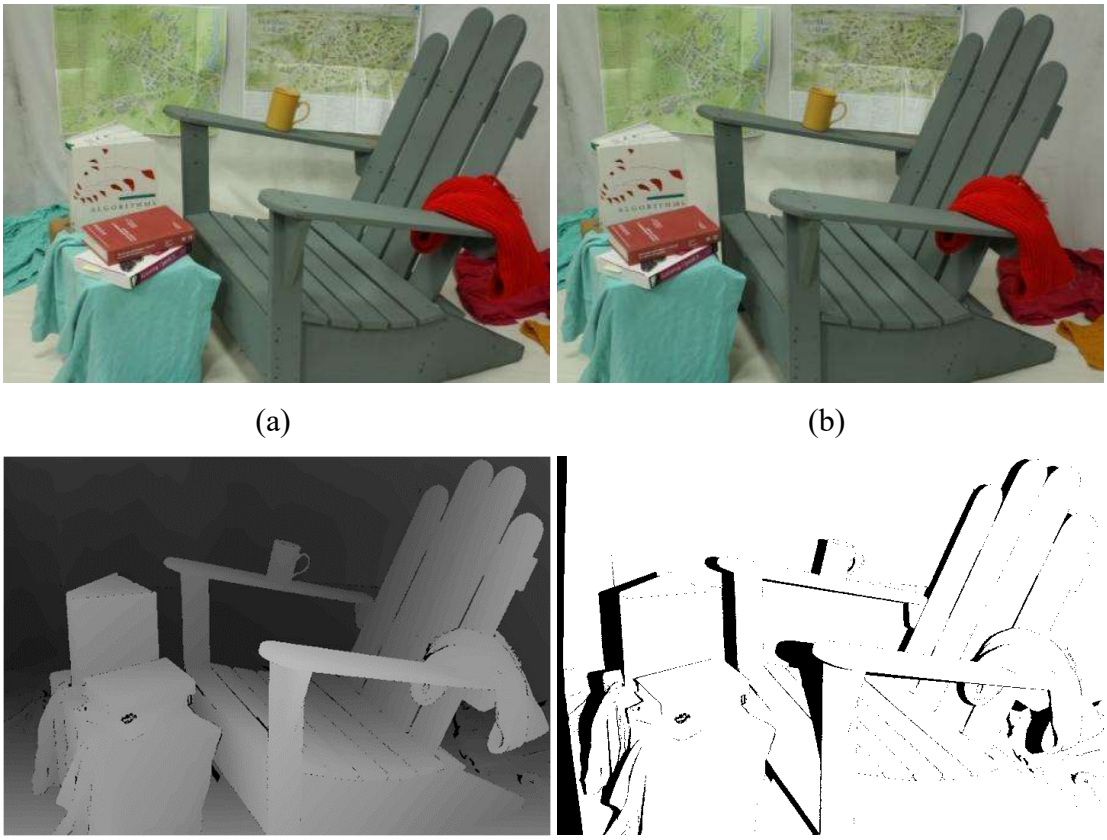
Figure 4.9 Types of cameras used in this research: (a) ZED camera by Stereolabs; (b) Aeria X by senseFly.

The first experiment on real-time dense image matching was related to human kinematics. In this experiment, a ZED camera was used (Figure 4.9a). The camera system included a stereo pair of RGB cameras of the same model on a mainboard. The baseline between the two cameras was 12 cm, and each camera had a horizontal field of view (FOV) of 90° and a vertical FOV of 60°. The left and right cameras had a focal length of 5.6 mm. The image resolution of each camera was 672 × 376 pixels, with a pixel size of 8 µm. The manufacturer had calibrated the camera system. The camera interior orientation parameters, including the focal length, the offset of the principal point, lens distortions, and a fundamental matrix defining the relative orientation of the stereo cameras, were provided. In the experiments, we used a local coordinate system, with the origin at the perspective centre of the left camera, the X-axis along the baseline, the Y-axis pointing downwards, and the Z-axis pointing to the range direction.

Another experiment on real-time dense image matching was conducted. The images were captured by an Aeria X camera mounted on the UAV (Figure 4.9b). The camera had a focal length of 19 mm, with an image resolution of 6000 × 4000 pixels. The FOV of the camera was 64° vertical and 90° horizontal. The camera was also calibrated by the manufacturer, and all of the interior orientation parameters were provided. All of the UAV images are accessible through the senseFly website (senseFly, 2019).

Both experiments were run on a computer with two NVIDIA RTX 2080Ti graphics cards, 64 GB RAM, and two 12-core CPUs. The real-time processing capability of the algorithm was evaluated through the evaluation and comparison of the processing efficiencies of the traditional SGM algorithm and our parallel-architecture acceleration method. The traditional SGM algorithm was implemented using OpenCV, and the evaluation was conducted using images captured by stereo cameras. During the evaluation, several frames were captured, and the disparity maps were obtained using our GPU-accelerated SGM and the stereo_SGBM function of OpenCV.

4.4.2 Evaluation of GPU-Accelerated SGM on Benchmark Dataset

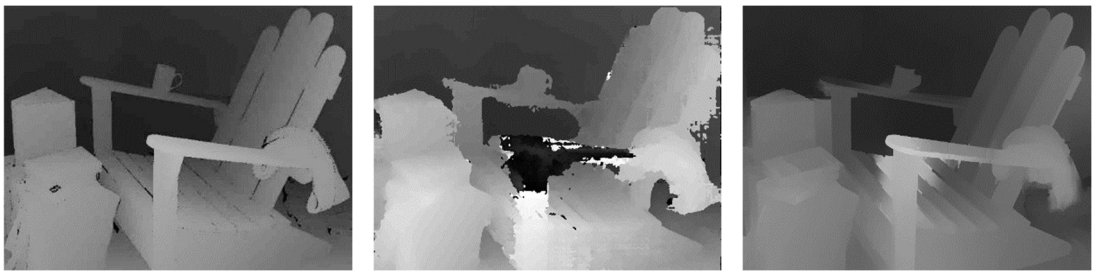


(a) (b)

(c) (d)

Figure 4.10 Evaluation dataset from the Middlebury stereo vision dataset: (a and b) left and right views of the dataset; (c) the ground-truth disparity; (d) mask of the valid disparity in (c).

The effectiveness of our GPU-accelerated SGM algorithm for disparity map generation was evaluated using a single stereo image pair with ground truth from the Middlebury stereo vision dataset (Scharstein *et al.*, 2014). The dataset (Figure 4.10) contained left-view and right-view images with a resolution of 497 × 720 pixels. The corresponding ground-truth disparity and mask of valid pixels were provided; they were similar to those of the test dataset in Section 4.3.3, and the latest version was used for stereo image matching evaluation.



(a) (b) (c)

Figure 4.11 Disparity results obtained using traditional and GPU-accelerated SGM: (a) ground-truth disparity; (b) disparity map obtained using traditional SGM; (c) disparity map obtained using GPU-accelerated SGM.

Table 4.2 Accuracy and efficiency evaluation results of traditional SGM and GPU-accelerated SGM

	Traditional SGM	**GPU-accelerated SGM**
Accuracy (%)	64.5	88.9
Processing time (ms)	282.1	18.3

The dataset shown in Figure 4.10 was used to assess the precision of a GPU-accelerated SGM algorithm and the effectiveness of a singular image pair. The objective was to determine the accuracy and performance of the algorithm through a comparison of the generated disparity maps with the ground-truth disparity. As stated in the previous section, the algorithm was evaluated using SAD as the similarity measure. A window size of 7 × 7 and a search range with a maximum disparity of 64 pixels was adopted. Figure 4.11(a) presents the ground-truth disparity of the stereo image pair, which was used as a benchmark for evaluating the accuracy of the disparity maps generated by the

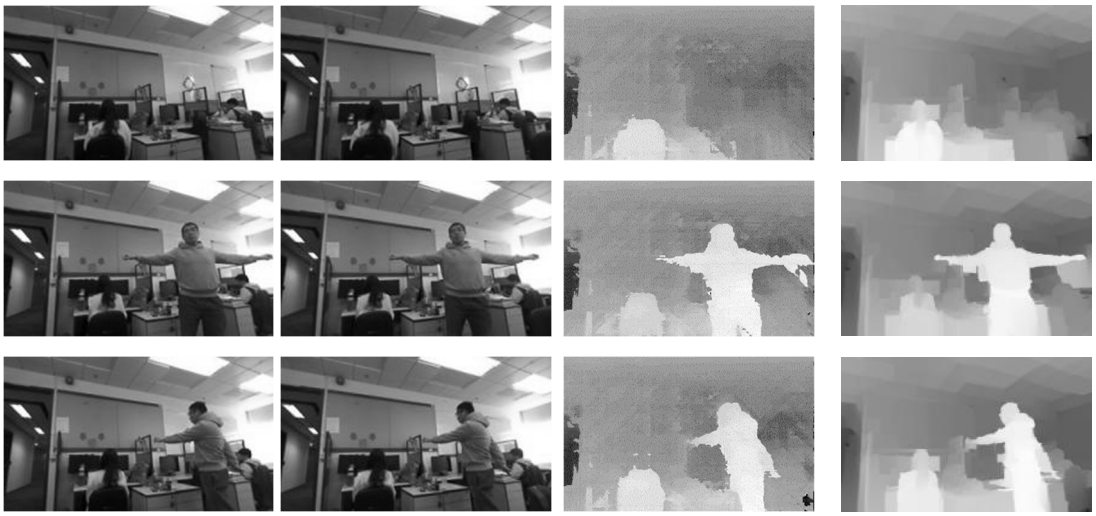
algorithm. Figures 4.11(b) and (c) depict the disparity maps generated by the traditional and GPU-accelerated SGM algorithms, respectively. The disparity maps were compared with the ground-truth disparity to evaluate the accuracy of the generated maps.

The chair rail in the middle of the left-view and right-view images featured a significant positional variation, leading to a large variation in the content of the corresponding pixels in the epipolar-plane stereo image. Figure 4.11(b) shows that the pixels in this area produced a mask with a distinct pattern. This non-uniformity in the image can lead to discontinuities in the estimated disparity, which may result in spurious matches or incorrect peak disparity values. Traditional SGM approaches with peak filters have been adopted to address this issue (Hirschmuller, 2007). However, as illustrated in Figure 4.11(b), this filtering approach may also remove valid disparity values, leading to a substantial amount of invalid disparity in the final result. The accuracy was assessed according to Equation 4.7. As shown in Table 4.2, disparity accuracy was significantly improved from 64.5% to 88.9% after the application of WLS filtering. The processing time of this single stereo image pair was considerably decreased to 18.3 ms. The results indicate that our approach can provide a robust disparity suitable for real-time dense image matching.

4.4.3 Evaluation of GPU-Accelerated SGM on Stereo Close-Range Images

To evaluate the application of our approach to close-range images, a stereo camera was used to capture SBS images featuring the human body at a resolution of 1344 × 376 within 600 s, and the effectiveness of the algorithm for real-time processing was evaluated. Figure 4.12 illustrates the results of real-time disparity map generation using the traditional SGM and GPU-accelerated SGM algorithms. The image obtained using the traditional SGM algorithm (Figure 4.12c) featured mismatches and noise consisting of occlusion and textureless regions caused by excessive sunlight from the window. In contrast, the disparity map obtained via our approach featured distinct contours of the human body, both in a sitting and standing position (Figure 4.12d). WLS helped minimise the effects of occlusions and textureless regions on the final disparity maps;

thus, our approach provided a disparity map that could delineate the contours of the object.



(a) (b) (c) (d)

Figure 4.12 Real-time disparity map generation results obtained using traditional and GPU-accelerated SGM: (a) the left-view images in greyscale; (b) the right-view images in greyscale; (c) disparity map obtained using traditional SGM; (d) disparity map obtained using GPU-accelerated SGM.

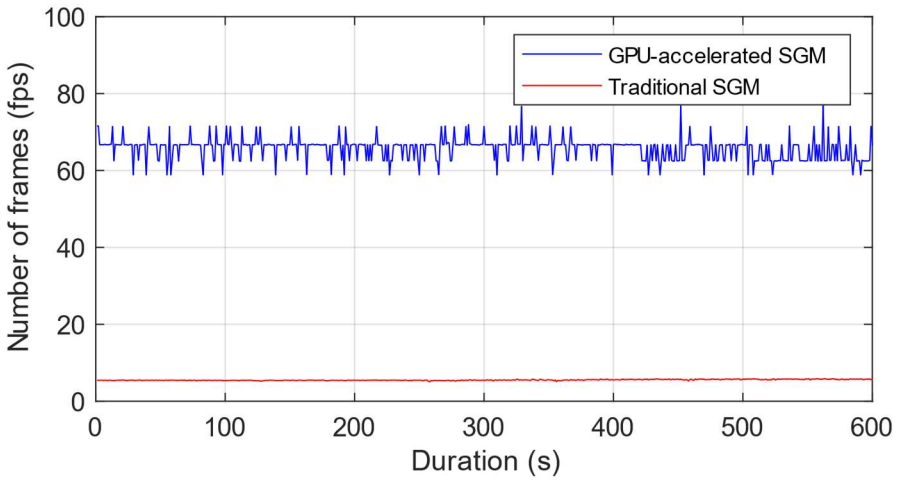


Figure 4.13 Comparison of the processing efficiency between traditional SGM and GPU-accelerated SGM on SBS images.

Table 4.3 Comparison of the real-time processing efficiencies of traditional and GPU-accelerated SGM on close-range images

Approaches	Real-time processing efficiency (fps)		
	Min	Max	Mean
Traditional SGM	1.29	5.92	5.24
GPU-accelerated SGM	58.82	76.92	65.5

Figure 4.12 depicts the real-time processing efficiency of disparity map generation from SBS images using the traditional SGM and GPU-accelerated SGM algorithms. The red line indicates the SBS image processing efficiency of the traditional SGM algorithm implemented with OpenCV, while the blue line represents the efficiency of our GPU-accelerated SGM algorithm. Figure 4.13 compares the real-time processing efficiencies of the traditional and GPU-accelerated SGM algorithms. The GPU-accelerated SGM algorithm maintained an average real-time processing efficiency of ~65.4 frames per second (fps) during the 10 min experimental record, while the traditional SGM algorithm maintained a speed of only ~5.2 fps. However, the graph also shows that the processing efficiency of the GPU-accelerated SGM algorithm fluctuated. As shown in Table 4.3, both the traditional and GPU-accelerated SGM algorithms varied in the interval between the maximum and minimum fps. This occurred owing to the fast movement of the individual being photographed, and a ghost effect occurred on the corresponding frames under unstable illumination, affecting the computational efficiency of matching and smoothing. The significant difference in processing efficiencies shows that parallel architecture-based image matching acceleration can significantly improve the real-time processing efficiency of the SGM algorithm.

4.4.4 Evaluation of GPU-Accelerated SGM on Aerial Images

The evaluation of the GPU-accelerated SGM algorithm on close-range images in the previous section shows that the computational efficiency of this method is significantly higher than that of the traditional SGM approach. Furthermore, we evaluated the GPU-accelerated SGM algorithm using images taken by a UAV. Chapter 2 introduces the background and purpose of our research on the real-time processing of aerial images

captured by the camera on UAVs during flight. Our goal is to achieve real-time processing of aerial images to generate a digital terrain model for the visual localisation of the UAV. However, the task is challenging owing to the high resolution of aerial images and the changing angle of the images caused by UAV oscillation during flight. The dense image matching algorithm must be fast and optimised for images captured from different angles to effectively handle a large amount of image data in real-time.

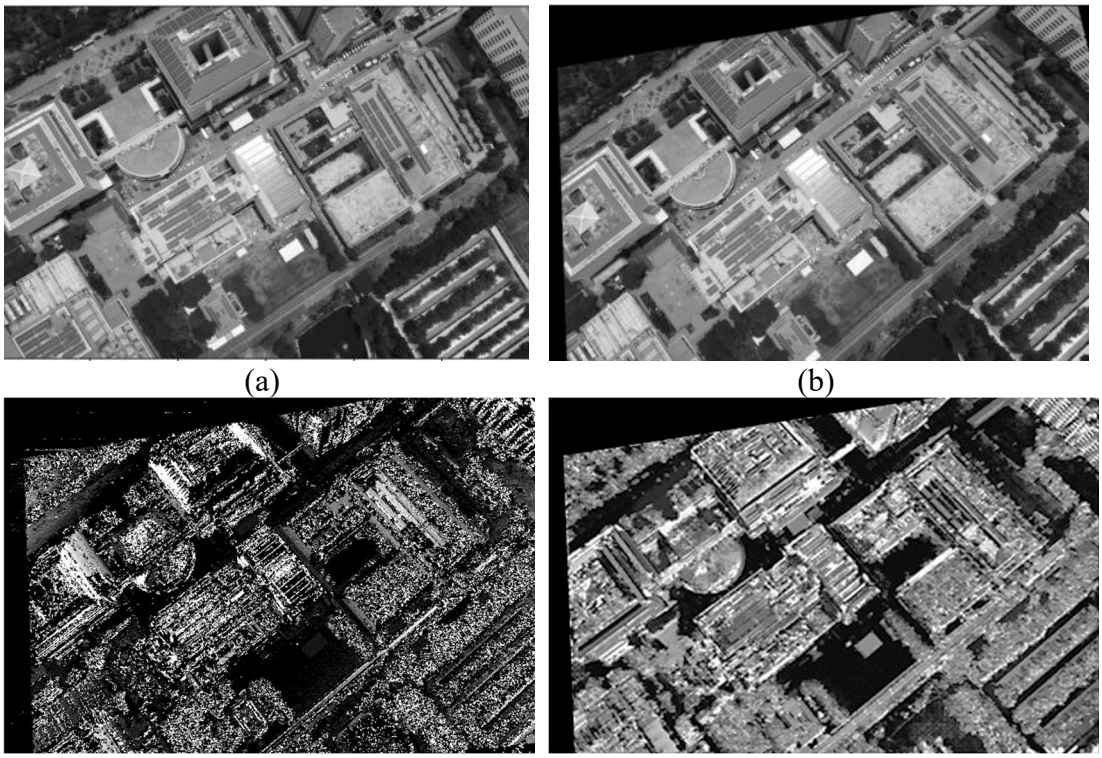


Figure 4.14 Disparity map generated by traditional and GPU-accelerated SGM: (a and b) two consecutive aerial images captured by UAV; (c) disparity map obtained using traditional SGM; (d) disparity map obtained using GPU-accelerated SGM.

Therefore, before a large amount of data is processed using an efficient dense image matching technique, epipolar calibration should be performed on every consecutive pair of images. Two aerial images captured from different viewpoints can be corrected via epipolar calibration, which ensures that the corresponding points lie on conjugate epipolar lines for dense image matching. Our GPU-accelerated SGM algorithm can then be applied to perform dense image matching to generate a more robust disparity map.

Figure 4.14 shows the results of disparity map generation using SGM from two consecutive aerial images captured at 6000 × 4000 resolution after epipolar calibration. Rotation and transformation matrices were first calculated for two consecutive aerial images to perform epipolar calibration. Subsequently, both the traditional and GPU-accelerated SGM algorithms were applied for dense image matching. The disparity map generated using traditional SGM (Figure 4.14c) featured a larger proportion of mismatches and noise than that generated using our GPU-accelerated SGM. The WLS filter reduced the noise in the results and allowed the object to retain a complete edge in the disparity map (Figure 4.14d).

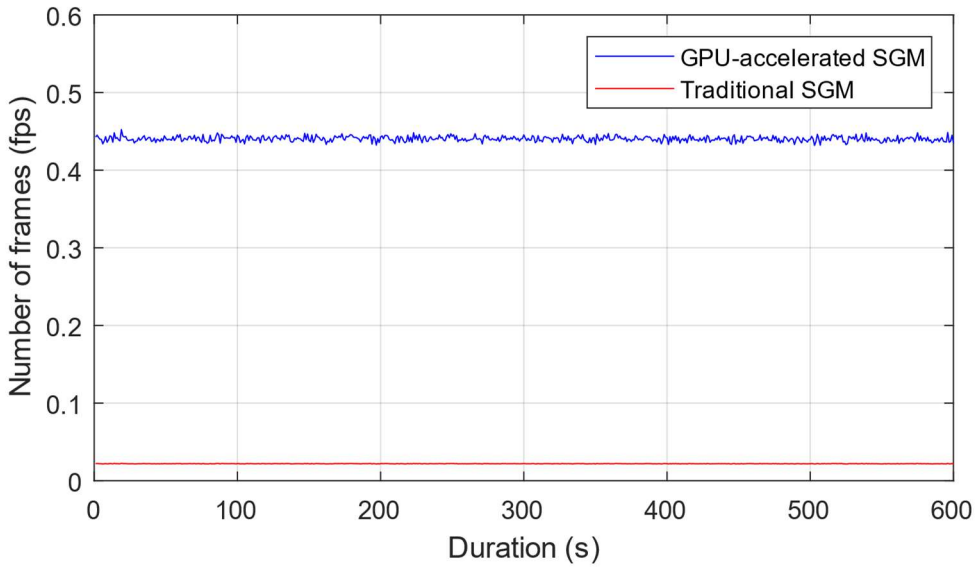


Figure 4.15 Comparison of the processing efficiency of traditional SGM and GPU-accelerated SGM on UAV images.

Table 4.4 Comparison of the real-time processing efficiencies of traditional and GPU-accelerated SGM on aerial images

Approaches	**Real-time processing efficiency (fps)**		
	Min	Max	Mean
Traditional SGM	0.021	0.023	0.022
GPU-accelerated SGM	0.427	0.452	0.440

The aerial images featured a significantly higher resolution than the SBS image captured by the stereo camera adopted in the previous experiment. As a result, the real-time processing efficiencies of both the traditional SGM and GPU-accelerated SGM algorithms were expected to be lower than the experimental results. The traditional SGM algorithm exhibited a low real-time processing efficiency of 0.022 fps for aerial images (Figure 4.13, red line). This corresponds to a processing time of ~45 s for a pair of images, making the algorithm unsuitable for real-time applications. The GPU-accelerated SGM significantly improved the processing efficiency, with a real-time processing efficiency of 0.44 fps (Figure 4.13, blue line). Our approach improved the processing time for a pair of aerial images by ~20 times more efficiently than the traditional SGM (Table 4.4). The GPU-accelerated SGM algorithm offers a promising solution for real-time UAV image processing, as it improves processing efficiency for dense image matching, according to the assessment results in Table 4.4.

Chapter 5 Real-Time 3D Data Generation and Applications

5.1 Triangulation for 3D Position Determination

Triangulation is a fundamental method for determining the precise position of an object or point in three-dimensional space. This process involves the measurement of angles formed between the object or point of interest and multiple reference points, thereby enabling accurate localisation. Triangulation, also known as space intersection, is widely applied in photogrammetry, with the fundamental geometric relation being the well-known collinearity equation (Wu, 2021). The collinearity equation, as introduced in Chapter 2, establishes a mathematical relationship between the 3D coordinates of a point, the 2D image coordinates, and camera parameters. This collinearity relationship is essential in determining the 3D coordinates of an object point based on its image coordinates and camera parameters.

Figure 5.1 illustrates the concept of stereo triangulation. Two cameras, referred to as a stereo pair, are used to capture images of the same scene from different viewpoints. Due to the offset viewpoints of the cameras, the captured images exhibit disparities, which indicate the horizontal shift between the corresponding points in the left and right images. The known baseline distance between the cameras and disparity information can be used to calculate the depth or distance of objects using triangulation principles (Hartley and Zisserman, 2003).

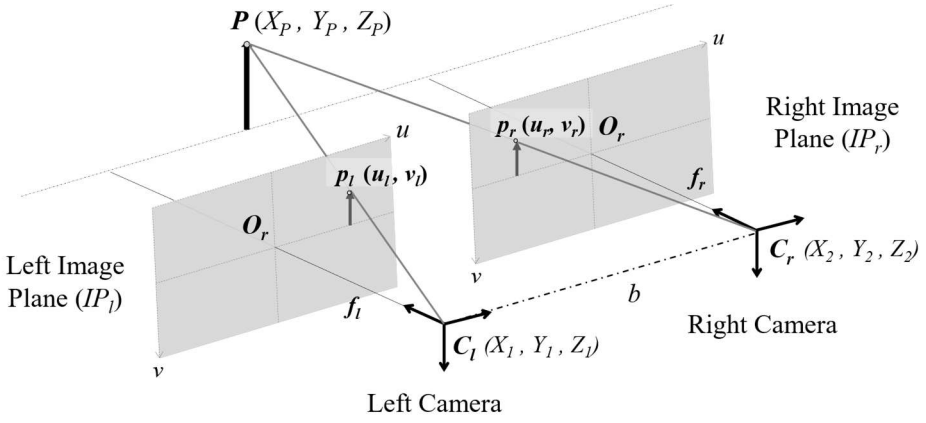


Figure 5.1 Stereo geometry for triangulation

In Figure 5.1, C_l and C_r are the centres of the left and right camera sensors, respectively; and IP_l and IP_r are the corresponding image planes. O_l and O_r on the image planes are the optical centres of the left and right cameras, respectively. Given any point $P(X_P, Y_P, Z_P)$ of the object in the real world, $p_l(u_l, v_l)$ and $p_r(u_r, v_r)$ are pixel-point representations of P in the IP_l and IP_r planes of the stereo cameras at sensors C_l and C_r, respectively. The baseline is the offset distance between the optical centres of the camera sensors C_l and C_r. The world point P is transformed from the pixel point using the interior orientation parameters and translation between two image planes. The relationship between these parameters in the homogeneous coordinate system can be expressed as follows:

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix} = W \cdot \begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix} \tag{5.1}$$

$$\begin{bmatrix} X_P \\ Y_P \\ Z_P \\ 1 \end{bmatrix} = Q \begin{bmatrix} u \\ v \\ d \\ 1 \end{bmatrix} \tag{5.2}$$

where R and T represent 3×3 rotation and 3×1 translation matrices, respectively; W is a projection matrix derived from the interior orientation parameters and translation matrix; Q is a re-projection matrix that enables the translation of pixel points to world points; and d is the disparity between the pixels in the left and right image planes. The

rotation and translation matrices can be obtained from preliminary calibrations, and Eq. 5.2 can be alternatively represented using intrinsic parameters, as follows:

$$\begin{bmatrix} X_P \\ Y_P \\ Z_P \end{bmatrix} = \begin{bmatrix} u - c_x \\ v - c_y \\ f_x \end{bmatrix} \cdot \frac{b}{d} \tag{5.3}$$

where b is the baseline of the camera pair, f is the camera focal length, (c_x, c_y) is the optical centre of the corresponding sensor, and d is the disparity value of any pixel point (u, v). The focal length of a single sensor is fixed. Hence, the distance Z_p of the world points depends solely on its disparity component, which is calculated for each point from the left-view to the right-view images, as described previously. In this manner, each pixel in the disparity map can be transformed to a 3D coordinate. The resulting 3D point cloud is typically represented in the camera coordinate system.

5.2 Real-Time Triangulation Based on GPU Acceleration

In recent years, the use of GPUs has extended beyond their original purpose of accelerating graphics rendering, and they have been applied as powerful acceleration tools in various domains. NVIDIA's CUDA Programming Model has significantly facilitated this transition, enabling developers to write general-purpose programs for GPUs using a language based on C/C++. GPUs offer several advantages for image processing and computer vision tasks due to their high memory bandwidth, efficient access to large image datasets, and ability to exploit data parallelism. The numerous cores on GPUs can be used to implement a divide-and-conquer approach, which is particularly beneficial for handling high-resolution images. The programming of GPUs is predominantly based on the single-instruction multiple-data (SIMD) model, in which multiple threads execute the same operations simultaneously on different data. Consequently, an algorithm must be well-suited for this SIMD computational model to effectively leverage the computational capabilities of GPUs.

This research introduces a GPU-based triangulation method that builds upon the cost function proposed by Recker et al. (2013). The proposed method is highly amenable to parallelisation due to the large number of independent tasks. By leveraging the

capabilities of the CUDA programming model, a parallelisation strategy is used for the GPU, in which one thread block is assigned to each track for processing. The triangulation algorithm exploits GPU properties derived from the L1 cost function and its gradients, as described in Section 5.2.1. The implementation of triangulation with parallelisation is discussed in Section 5.2.2.

5.2.1 Cost Function for Triangulation

The previous section outlines the process of triangulation, which involves solving the collinearity equation. The collinearity equation uses the geometric relationship between the camera, the object in space, and its corresponding 2D point on the image plane as parameters. Notably, each individual collinearity equation can serve as a cost function suitable for parallel computation on a GPU. This parallelisation approach enables the simultaneous processing of multiple collinearity equations. In this manner, the computational capabilities of the GPU can be leveraged to expedite triangulation.

A prospective 3D location p and its accompanying feature track t can be evaluated using an angular error measure based on the L1 triangulation cost function (Recker et al., 2013). The inputs to the cost function are a collection of feature tracks across N images and their corresponding 3 × 4 camera projection matrices P_i. The error for position p is calculated as follows:

$$f_{t\in T}(p) = \frac{\sum_{i\in I}(1 - \widehat{v_i} \cdot \widehat{w_{ti}})}{|| I ||} \tag{5.4}$$

Calculation of the error at position p involves several stages: First, a unit direction vector v_i is computed between the centre of each camera and the candidate position p. Subsequently, a second unit vector w_{ti} is determined, which originates from each camera centre C_i and passes through the 2D feature track t in each image plane. Because the feature track t may not precisely align with the projection of position p in each image plane, a non-zero angle typically exists between the potential direction vector v_i and vector w_{ti}. Finally, the cost function is computed as the mean of the dot products ($\widehat{v_i} \cdot \widehat{w_{ti}}$) across all cameras. Each dot product has a range of [-1, 1], but only the points in

front of the cameras are evaluated, limiting the range to [0, 1]. To maintain consistency with the L1 triangulation cost function, we use the same notation and define Eq. 5.4 as the cost function for evaluating the 3D position p pertaining to track t.

5.2.2 GPU-Based Implementation of Triangulation

To facilitate comprehension, we focus on the key features of CUDA that are relevant to our task. In CUDA, algorithms are referred to as kernels and are executed on parallel blocks with up to 1024 threads. The GPU allocates these blocks to its multiple streaming multiprocessors, each responsible for the synchronised execution of 32 thread groups, known as warps, under the control of an SIMD. The use of shared memory, a small memory space that facilitates efficient data sharing among threads within a block, is a crucial aspect of CUDA programming. Notably, shared memory offers considerably faster access than DRAM (global memory), which is located off-chip. Effective GPU programs must maximise the use of computational resources by launching numerous threads; minimising thread divergence (i.e., threads following different control flows) within warps; and strategically using the memory hierarchy to prioritise fast shared memory over global memory, when possible (Nickolls et al., 2008). These considerations are essential for the development of GPU applications that are both efficient and effective.

In this study, triangulation is implemented using a block-based method for processing tracks, inspired by the methodology proposed by Mak et al. (2014). This strategy maximises computational efficiency by implementing parallelism within each segment. Instead of assigning a single thread per track, a group of threads is dedicated for processing each recording. The individual threads within these segments compute the specific term associated with each track feature. The angles between ray-based terms are then aggregated using a parallel reduction technique to derive the overall gradient of the cost function. This block-based strategy enables the concurrent processing of multiple features within a single track, thereby maximising parallelism and facilitating efficient triangulation computation.

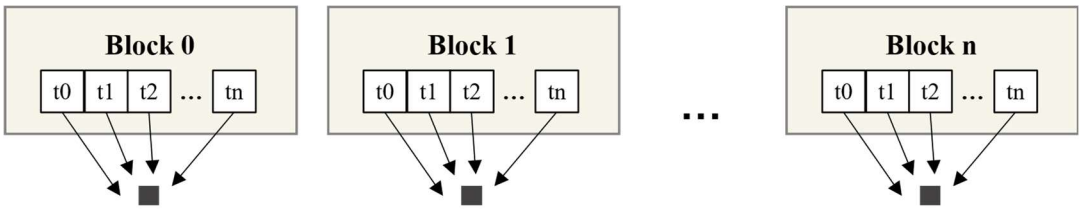
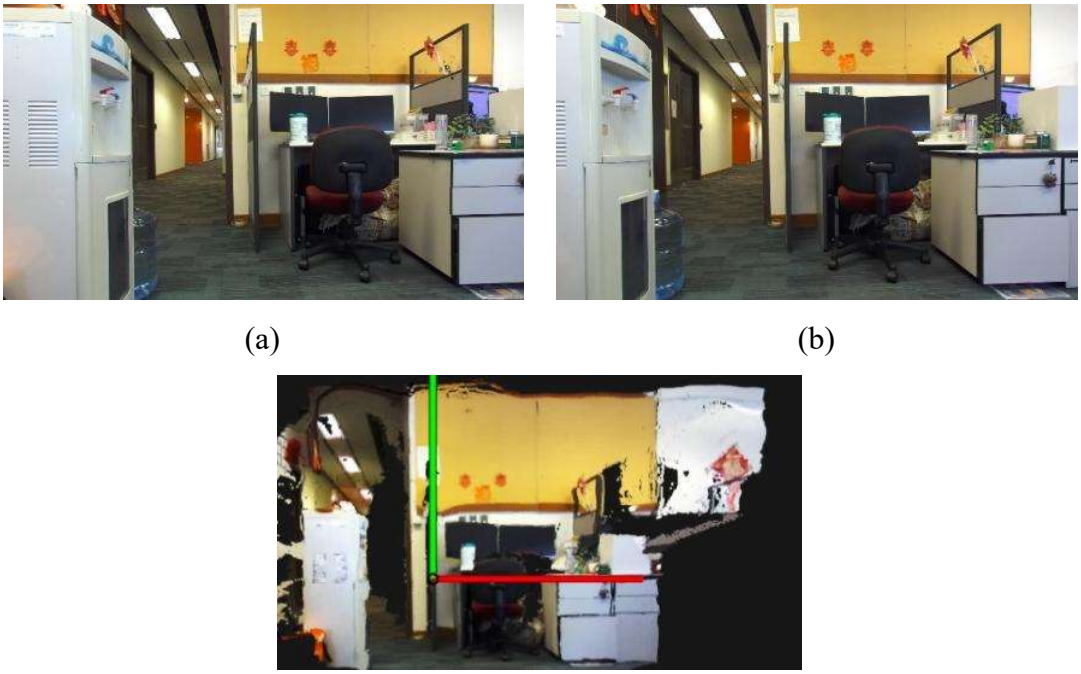


Figure 5.2 Concept of using one block per track for the multiple processes of triangulation. Each block consists of several tracks for solving collinearity equations.

Figure 5.2 illustrates the proposed implementation, tailored for datasets with long feature tracks. Each thread within a block is responsible for calculating the gradient for a particular feature, and a parallel sum reduction is performed to obtain the final gradient value for the entire track. This method exhibits enhanced performance when dealing with longer tracks, as the gradient computation workload varies with the track length, and multiple gradient calculations may be required until convergence. The utilisation of shared memory on GPU is a significant advantage of this approach. Modern GPUs provide thread blocks with limited access to shared memory. When one track is assigned to each thread, the shared memory may not be adequate for storing the data associated with all tracks within a block, even when sampling techniques are used. Assigning an entire block to a single track and integrating sampling can reduce the per-thread memory requirements. Consequently, the working set of track and camera data for a block can be accommodated in shared memory, effectively transforming it into a cache. In addition, the parallel sum reduction operation for gradient computation is also executed in shared memory, which facilitates the necessary inter-thread communication for the reduction process.

Figure 5.3 shows the results of an experiment involving GPU-based triangulation for 3D point cloud generation. The processing times and performance of the proposed GPU-based triangulation strategy were evaluated on a computer equipped with a 1.70 GHz Intel Xeon E5-2603 v4 CPU and an NVIDIA GeForce 2080Ti GPU.



(a) (b)

(c)

Figure 5.3 Experiment results of GPU-based triangulation. (a), (b) Inputs of stereo pair images (1920 × 1080 pixels). (c) Coloured point clouds from different views.

Figure 5.3 shows the result of implementing GPU-based triangulation for the generated 3D point clouds. The input for this process consists of a stereo pair of images (Figures 5.3 (a) and (b)), each sized 1920 × 1080 pixels. GPU-based triangulation generates a coloured point cloud (Figures 5.3 (c) and (d)). The processing pipeline involves the initial reading of the input images by the CPU, which transmits them to the GPU for triangulation. As shown in Figure 5.4, the average utilisation of CPU resources is 13%. The GPU helps to accelerate the triangulation process, with the average GPU usage being 19%. Eventually, the GPU-based triangulation generates a coloured point cloud.

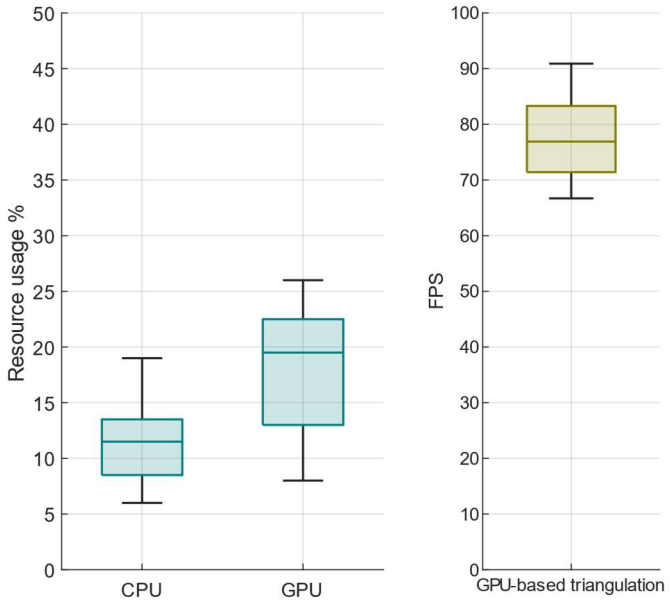


Figure 5.4 Resource usage and processing rate (fps) of GPU-based triangulation

5.3 Real-Time 3D Point Cloud Generation from Aerial Images

Real-time 3D point cloud generation is crucial for applications that require immediate visualisation, accurate object detection and tracking, and 3D mapping. These capabilities can enable real-time interaction with the 3D environment and enhance the efficiency and effectiveness of various systems and applications. In Section 5.1, triangulation and its algebraic representation are described. Specifically, Eqs. 5.1 to 5.3 can be used to triangulate 2D image points to 3D points in space coordinates. Chapter 3 describes the process for computing the camera pose for the current input frame in real-world space coordinates. This pose can be used as the exterior orientation to reconstruct 3D point clouds through GPU-accelerated triangulation, as discussed in Section 5.2. This section describes the strategy for real-time 3D point cloud generation.

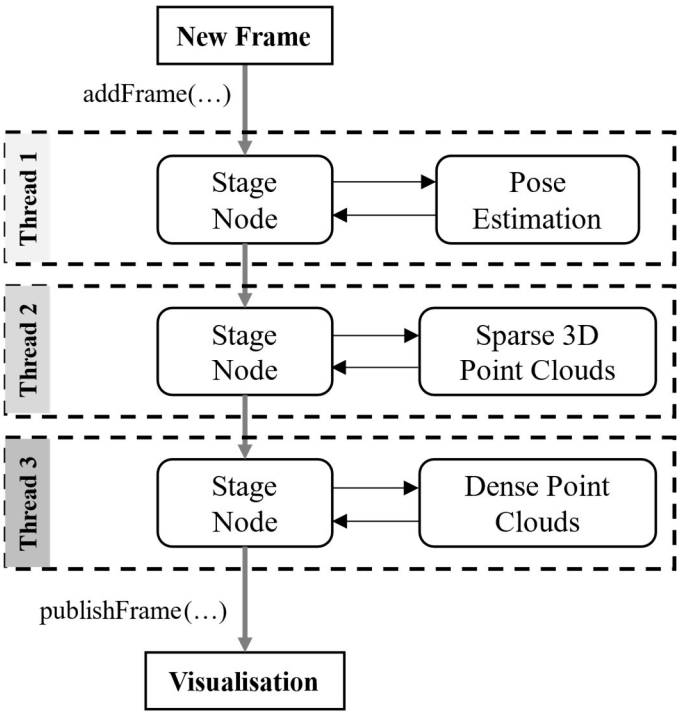


Figure 5.5 Framework of real-time 3D point cloud generation from aerial images

Figure 5.5 shows the framework for real-time 3D point cloud generation. The implementation over three different threads allows the parallel processing of incoming data. The first thread is responsible for camera pose estimation, following the method described in Chapter 3. After obtaining the camera poses of two consecutive frames, the frame is passed to the next thread to generate a sparse 3D point cloud through triangulation, following the approach described in Section 5.2. The third thread interpolates the sparse 3D point to construct a dense point cloud construction, as discussed throughout Chapter 5. The visualisation of the 3D point cloud is achieved using the Grid Map Core library (Fankhauser and Hutter, 2016). The transportation of each frame is realised using the robot operating system (ROS), which is commonly used to simulate and efficiently deploy real-time processing algorithms for robots. The ROS provides different nodes for the facile implementation of parallel processing and real-time visualisation of the final result.

5.3.1 3D Point Cloud Generation

The camera pose for the current input frame is obtained using the approach described in Chapter 3. Upon determining a keyframe based on a predefined threshold, the camera

pose is calculated through space resection according to the matches between two consecutive frames. This pose is used to refine the camera pose for the subsequent frames. The camera pose information is then used to generate 3D points through triangulation.

5.3.1.1 Local Optimisation for Generating Sparse 3D Points

In the process of extracting and matching features, 3D points must be rapidly generated by triangulation and added to the map to achieve real-time 3D mapping. In the proposed strategy, local optimisation is performed with a bag-of-words (BoW) vector for matching features on keyframes and ensuring their correspondence on neighbour subsequent frames. BoW involves the construction of a visual vocabulary by clustering the feature descriptors into a set of visual words. Figure 5.6 illustrates the BoW framework. The feature descriptors on each image are encoded to the nearest visual word, resulting in a histogram-like representation with visual word frequencies. The corresponding neighbouring frames of keyframes are matched by comparing the similarity of the visual words of the features in different image views. After verifying the correspondences, all of the features on the keyframes and corresponding neighbouring frames are triangulated to generate sparse 3D points. Fast BoW (FBoW, Gálvez-López and Tardos, 2012) is applied for local optimisation in this work.

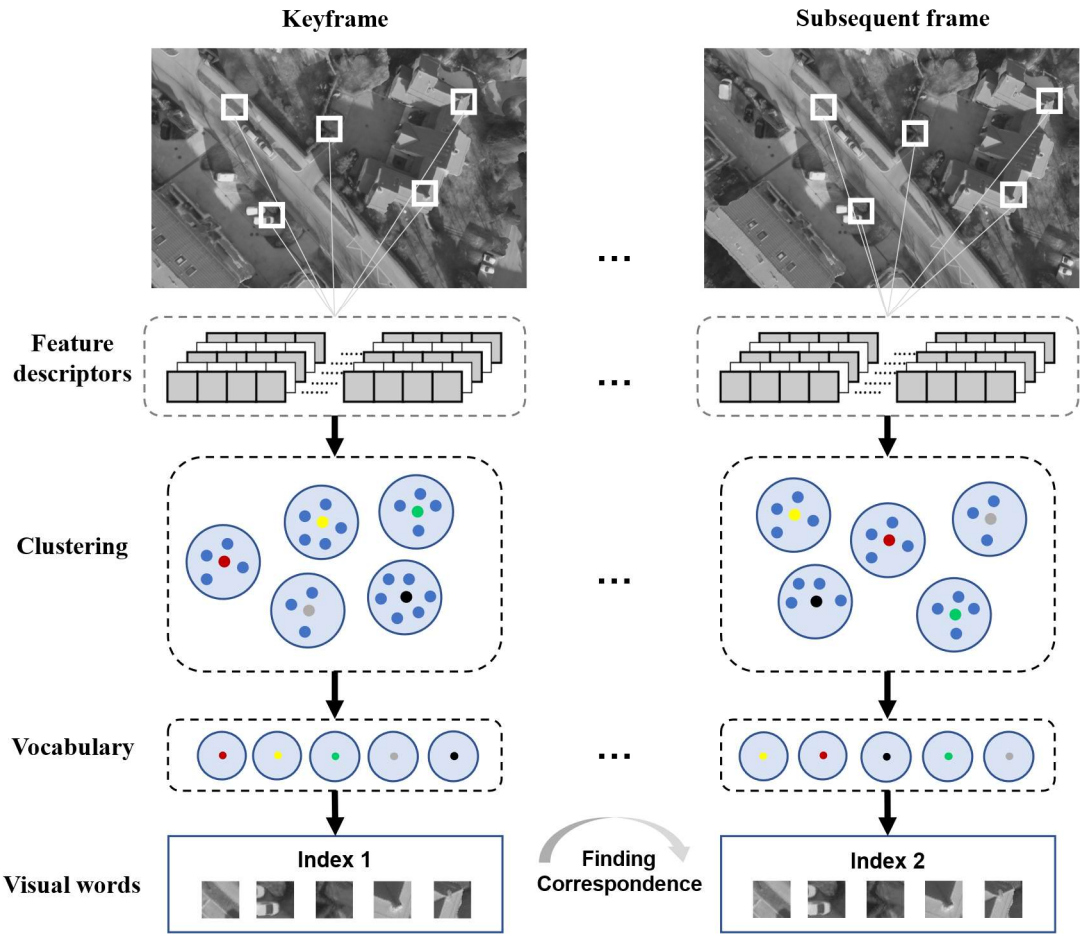


Figure 5.6 BoW framework for identifying the matching features on corresponding neighbouring frame by visual words.

5.3.1.2 Interpolation for Generating Dense Point Clouds

As discussed in the previous Section 5.3.1.1, local optimisation facilitates the identification of matching feature points between keyframes and adjacent frames. These feature points serve as the basis for triangulation, resulting in the generation of sparse point clouds. These sparse point clouds then serve as a reference for creating dense point clouds. Specifically, a sparse point cloud is transformed into a grid map while preserving the elevation information to generate a dense point cloud. This grid map representation allows for comprehensive and organised data storage. Further details regarding the grid map are provided in Section 5.3.1.3. To interpolate the sparse point cloud into a dense point cloud, a k-dimensional (k-d) tree is used to locate the nearest neighbouring feature points to the sparse points within the grid map. Finally, a dense point cloud with elevation information is generated and visualised in 3D.

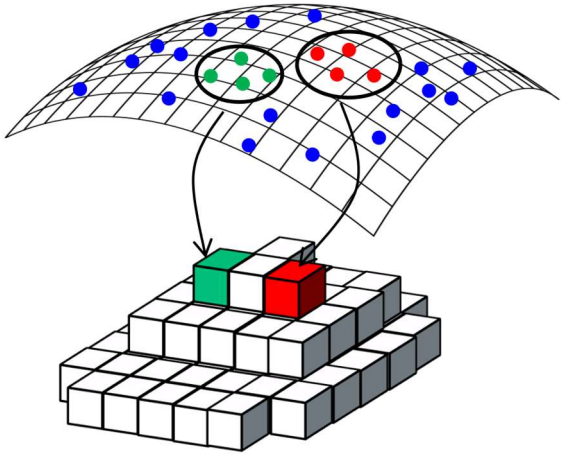


Figure 5.7 Interpolation of dense point clouds from sparse 3D points

Figure 5.7 illustrates the interpolation scheme for creating dense point clouds from sparse point clouds. The sparse point cloud serves as the input for dense point cloud generation. The algorithm involves a loop that iterates over all affected cells in the grid. The first step involves organising the *x*- and *y*-coordinates of the dense point cloud using a 2D binary k-d tree (Bentley, 1975). In the next step, this k-d tree returns a set of nearest points within the interpolation radius, which is determined by the average value of the distances from the nearest point according to the k-d tree. The inverse distance weighting (IDW) method is used to perform the interpolation. IDW computes the cell height by linearly combining the nearest neighbours with the weights determined using the inverse distance to the cell centre. Points closer to the cell centre are assigned higher weights, and thus, their influence on the interpolated height is more notable. The resulting interpolated height is assumed to be the ground sampling distance (GSD) for the creation of the grid map. Additionally, the region of interest (ROI) is determined by projecting the frame onto the reference plane. After adding the layers for 'elevation' and 'valid', a nearest neighbour search is conducted for each cell (X_{cell}, Y_{cell}) within the grid. The *z*-component is extracted from the dense point cloud for each identified neighbour, enabling the final interpolation of the cell height. These steps are detailed in Algorithm 5.1, with the implementation following the workflow presented by Hinzmann et al. (2018).

Algorithm 5.1: Pseudocode of dense point cloud creation

Input: ROI for the current frame, sparse point cloud

```
1:  KdTree kdtree = initKdTree(sparse point cloud)
2:  double resolution = estimateNearestPointDistance(kd tree, sparse point cloud)
3:  CvGridMap dense_map('elevation', 'valid')
4:     dense_map.setGeometry(ROI, GSD = resolution)
5:  for every cell in dense_map do
6:     Point query_point = (X_cell, Y_cell)
7:     vector<Point> neighbours = kdtree → findNearestNeighbours(query_point)
8:     if neighbours are found then
9:        dense_map.at(X_cell, Y_cell, 'elevation') = interpolateHeight(neighbours);
10:       dense_map.at(X_cell, Y_cell, 'valid')=true
11:    end if
12: end for
```

5.3.1.3 Real-Time Visualisation of the 3D Point Clouds

The implementation of point cloud storage and visualisation is based on an adaptation of the Grid Map Core library (Fankhauser and Hutter, 2016). The fundamental concept can be summarised as follows: A structured framework is established, based on an ROI specified by its coordinates (x, y), width, height, and a specified sampling resolution. As shown in Figure 5.8, the ROI consists of multiple data layers, each capturing specific information. Detailed information can be assigned to individual sample points within these layers. For instance, in a 3D grid map model, one layer stores the observed height of a point relative to a reference plane, while another layer stores the corresponding surface normal and texture. This multi-layered strategy allows the generated 3D grid map model to comprehensively represent the various attributes associated with the observed points within the defined ROI.

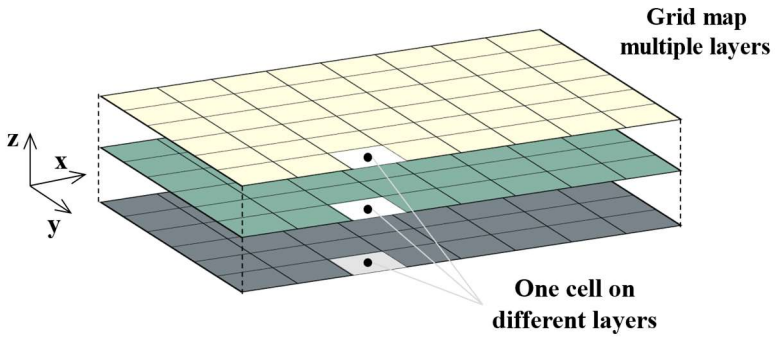


Figure 5.8 Example of a multi-layered grid map model

In this work, several modifications to the original Grid Map Core library are introduced. The original library was designed for a mobile robot, resulting in efficient dynamic map movement but limited incremental expansion capabilities. Although this design may be suitable for moving aerial robots, it is not suitable for creating a comprehensive global map. Furthermore, the original library represents all layers as Eigen matrices (Bates et al., 2013) with float values for cell storage. Although this framework can simplify mathematical operations involving layer manipulations, the implementation of computer vision tasks that rely on OpenCV becomes challenging (Bradski, 2000). To address these issues, a new framework, CvGridMap, is developed, which maintains the overall architecture of the original library but incorporates an OpenCV matrix-type backend. Moreover, the CvGridMap framework is extended to support dynamic map expansion and overlap computations, thereby enhancing the capabilities for the visualisation task considered in this study.

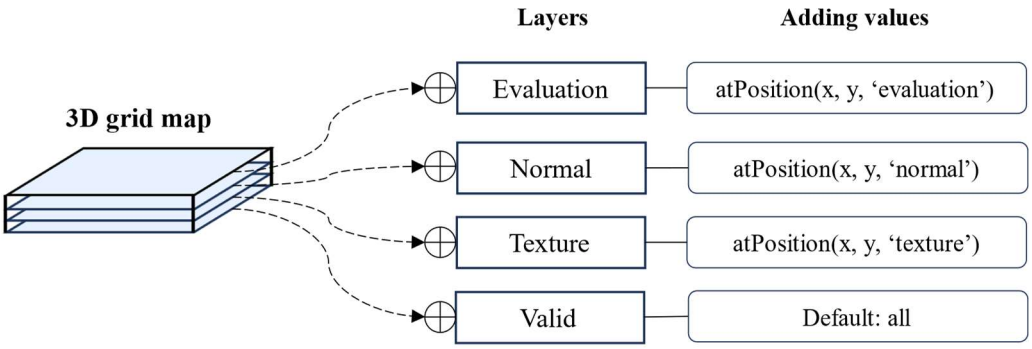


Figure 5.9 Initialisation of grid map for storing sparse point cloud information

Figure 5.9 illustrates the initialisation of the grid map, which involves the creation of four distinct layers. The elevation data within the map are determined through

triangulation, using the feature points matched between the keyframe and corresponding subsequent frame. The normal vector of each 3D point is computed based on its geometric relationship with the surrounding 3D points. Textures, represented using RGB colour, are obtained from the pixel points in the frame corresponding to the ROI. These layers collectively provide the necessary information for subsequent steps, such as the generation of dense point cloud maps and interpolation of additional points, guided by the validity indicators established in the initialisation phase.

In the final stage, the point cloud data acquired in the preceding steps are consolidated into a single scene to enable real-time visualisation. The point clouds in this stage are georeferenced, as the absolute orientation of the camera is used during triangulation. The first set of point cloud data is received and used for initialisation to construct the global map framework, which serves as the foundation for further data integration. Subsequently, new point cloud data are seamlessly fused with the existing global map using the 'map update' and 'map fusion' functions provided by the Grid Map Library. These functions facilitate the incorporation of new data while handling the overlap with the current global map, thereby ensuring the coherence and consistency of the overall map. Figure 5.10 demonstrates the procedure of point cloud fusion for real-time visualisation.

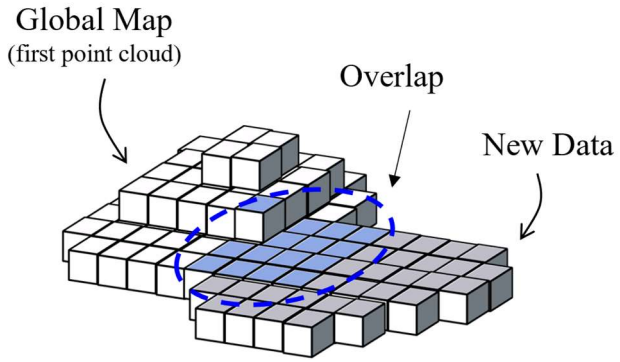


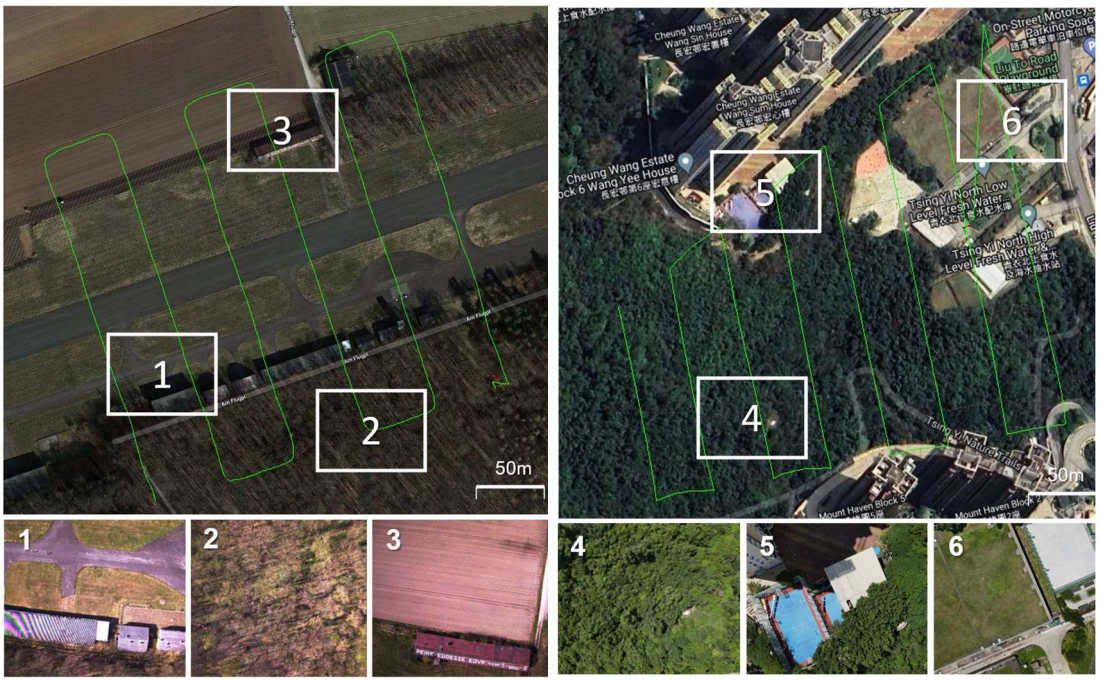
Figure 5.10 Point cloud fusion for real-time visualisation

5.3.2 Implementation and Experimental Evaluation

This section describes the implementation of techniques for the real-time generation of sparse 3D point clouds and interpolation of dense 3D point clouds. The proposed approach is validated using two aerial image datasets, referred to as Datasets 1 and 2. As described later in this section, the validation results indicate that the proposed approach can realise real-time 3D point cloud generation.

Dataset 1 is an openly accessible dataset consisting of 328 aerial images sized 2456 × 2054 pixels, which were captured by a UI-5280CP camera mounted on a customised UAV. Dataset 1 covers a land area of approximately 250 m^2 in Germany. The images for Dataset 1 were acquired with 99% end overlap and 50% side overlap. Dataset 2, a self-made made dataset, consists of 329 aerial images sized 4000 × 3000 pixels, which were captured using the camera on a DJI Mavic AIR 2 (Figure 5.11). Dataset 2 covers a land area of approximately 150 m × 200 m in Hong Kong. The images for Dataset 2 were captured with 80% end overlap and 60% side overlap. An overview of the coverage of both datasets is depicted in Figure 5.12. The images are downsampled to half of their original size to decrease the bandwidth and enhance the performance of triangulation and interpolation. The implementation and evaluation are performed on a computer with a 1.70 GHz Intel Xeon E5-2603 v4 CPU and an NVIDIA GeForce 2080Ti GPU.

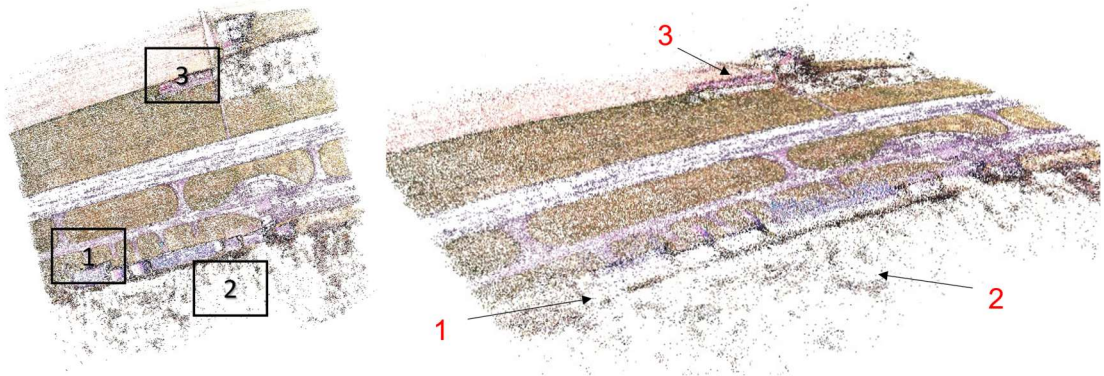
Figure 5.11 Collection of Dataset 2 by DJI Mavic AIR 2



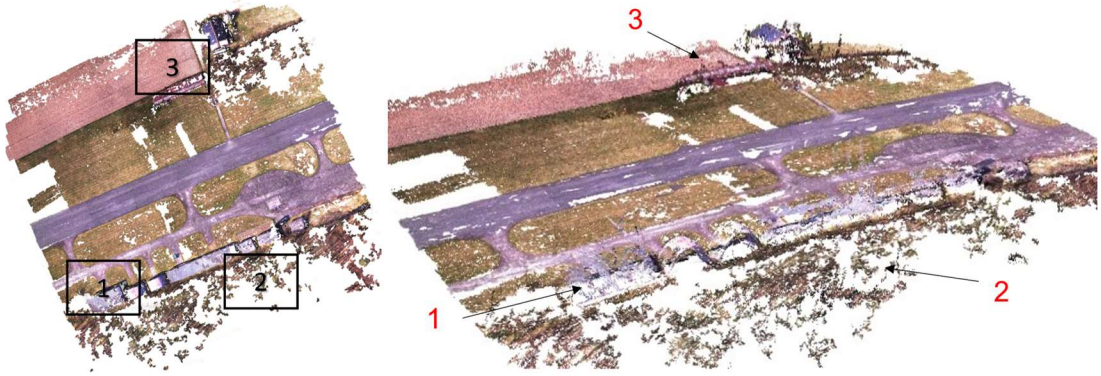
(a) Dataset 1 (b) Dataset 2

Figure 5.12 Overview of the coverage of experimental aerial image datasets

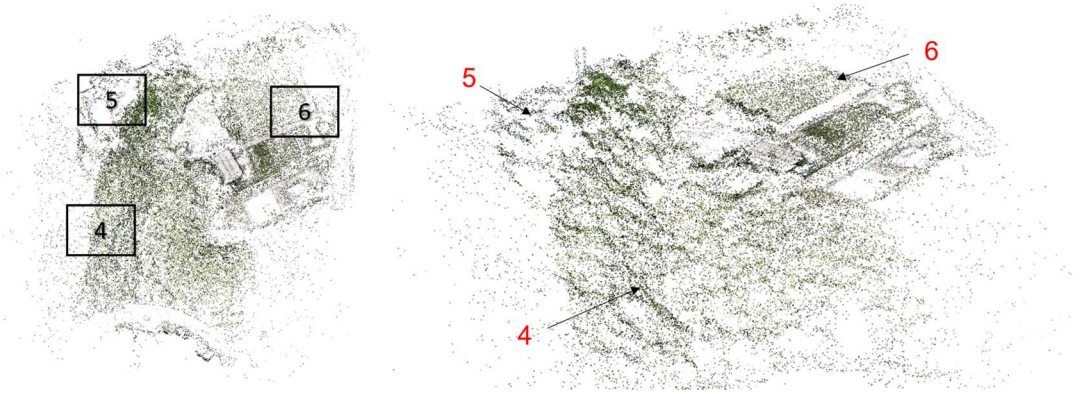
Both datasets present various challenges for visual algorithms, as illustrated in Figure 5.12. Figure 5.12 (a1) illustrates the regional aliasing effect on corrugated iron roofs, caused by the camera resolution. As shown in Figures 5.12 (a2) and (b4), the forest has a repetitive texture that renders feature extraction challenging. Feature extraction for the grassland area shown in Figures 5.12 (a3) and (b6) is challenging because of the homogeneous surface. The shadows in Figure 5.12 (b5) correspond to a low texture area from which the extraction of features is difficult.



(a) Results of sparse 3D point cloud generation from Dataset 1 from different views



(b) Dense point cloud interpolated from (a)



(c) Results of sparse 3D point cloud generation from Dataset 2 from different views

(d) Dense point cloud interpolated from (c)

Figure 5.13 Experimental results of sparse and dense point cloud generation.

To address the abovementioned challenges, the features are extracted using SuperPoint, as discussed in Chapter 3. SuperPoint, which is a deep learning method, can effectively extract features in textureless regions such as those shown in Figure 5.12. The sparse 3D point clouds generated by the proposed method for Datasets 1 and 2 are shown in Figures 5.13 (a) and (c), respectively. The corresponding dense point clouds obtained by interpolation based on the IDW method are shown in Figures 5.13 (b) and (d). In the interpolation process (Algorithm 5.1), the radius of the search neighbourhood is determined using the features in the k-d tree structure and sparse point cloud.

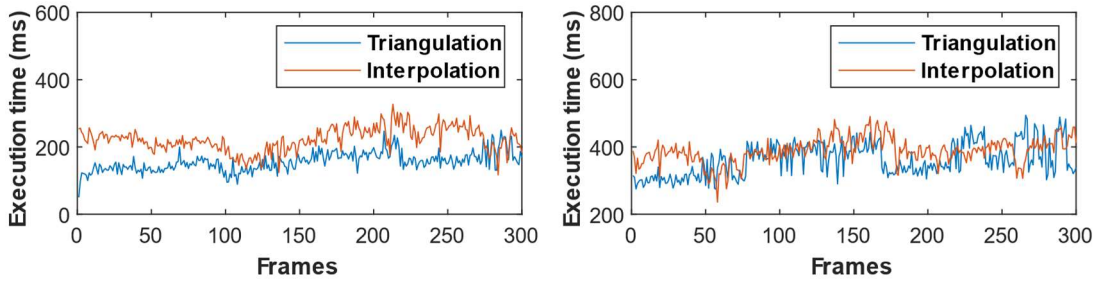
Figures 5.13 (a1), (a2), and (a3) illustrate the sparse point clouds for the regions with iron roofs (Figure 5.12 (a1)), southern forested area (Figure 5.12 (a2)), and grassland (Figure 5.12 (a3)), respectively. The sparse point clouds for the forest area and grassland shown in Figures 5.12 (b4) and (b6), respectively, are shown in Figures 5.13 (c4) and (c6). Notably, the shaded region in 5.13 (c5) indicates missing point clouds due to insufficient feature points. In the iron roof area shown in Figure 5.13 (b1), texture interference results in the inadequate representation of most textures within the dense point cloud. Similar observations can be made from Figures 5.13 (b2) and (d4), corresponding to the forest regions in Figures 5.12 (a2) and (b4), respectively. In contrast, in the grassland region shown in Figure 5.13 (b3), the feature points are evenly distributed, resulting in a high-quality dense point cloud. A similar outcome is observed in Figure 5.13 (d6): A high-quality dense point cloud is obtained for the grassland region in Dataset 2. Nevertheless, a dense point cloud cannot be accurately interpolated for the shadow regions (Figure 5.13 (d5)) due to the absence of features, as in the case of their sparse 3D point cloud.

Table 5.1 presents the statistics associated with the generation of sparse 3D point clouds and dense point clouds for Datasets 1 and 2. f_T denotes the features used in the triangulation process for constructing sparse 3D point clouds, and f_I denotes the features used during interpolation to generate dense point clouds. P_{TA} and P_{IA} represent the percentages of features used in triangulation and interpolation among all features, respectively. D_n is the nearest neighbour point distance (Algorithm 5.1).

Table 5.1 Statistics of sparse and dense point cloud generation

Dataset	Total images	Total features	Sparse 3D point clouds			Dense point clouds			
			f_T	P_{TA}	Points	D_n (pixels)	f_I	P_{IA}	Points
1	328	2,217,172	766,175	34.5%	185,666	20	1,870,473	84.4%	3,510,445
2	329	3,736,392	324,136	8.7%	51,702	162	2,983,361	80.1%	670,587

For Dataset 1, approximately 760,000 features (34.5% of all features) are selected for triangulation, resulting in the generation of a sparse 3D point cloud with 190,000 points. For dense point cloud generation, the neighbourhood search radius is 20 pixels, and approximately 1.9 million features (85.4% of all features) are used for interpolation. Dataset 2 consists of 329 aerial photographs, from which more than 3.7 million feature points are extracted. Among these features, approximately 320,000 (8.7% of all features) are used to generate sparse 3D point clouds via triangulation. The search radius is 162 pixels, and approximately 3 million features (80% of all features) are used to generate dense point clouds.



(a) Dataset 1 (b) Dataset 2

Figure 5.14 Execution time of triangulation and interpolation on the two datasets.

Figure 5.14 illustrates the efficacy of the triangulation and interpolation procedures for generating sparse and dense 3D point clouds from the two datasets. As shown in Figure 5.14(a), in the case of Dataset 1, the triangulation process exhibits a greater variation in performance than does interpolation. This discrepancy is attributable to the additional complexity involved in scene reconstruction, which relies on the textures present in the area. The average execution time values for triangulation and interpolation are 156 ms and 223 ms, respectively. In the processing of Dataset 2 (Figure 5.14(b)), sporadic high

peaks can be observed in the triangulation stage, whereas low peaks can be observed in the interpolation stage. This behaviour can be attributed to the presence of texture-rich regions within the dataset, owing to which more sparse point clouds are generated. Consequently, less time is required for the interpolation of dense point clouds. The average execution time values for triangulation and interpolation are 362 ms and 383 ms, respectively.

5.4 Real-Time Acquisition and Monitoring of 3D Human Body Kinematics

The real-time capture and analysis of human locomotion at a large scale are crucial for various applications, such as monitoring the actions of patients during physical rehabilitation (Karunarathne et al., 2014), ensuring worker safety in domains with industrial robots (Seo et al., 2015), analysing the movements of athletes (Gholami et al., 2019), and facilitating human–computer interactions in virtual reality (Jaimes and Sebe, 2007). Given that the success of these applications depends on the accurate extraction and analysis of 3D human body kinematics at a large scale, real-time photogrammetric systems with the corresponding capabilities have been extensively researched in recent years.

Real-time computations have become feasible with the advent of CPUs with multi-threaded capabilities and GPU-acceleration technologies. This work introduces a cost-effective photogrammetric system involving a stereo pair of RGB cameras. Using GPU-acceleration and multi-threading technologies, this system can extract and monitor 3D human body kinematics in real-time and on a large scale. Section 5.4.1 describes a strategy that combines a 2D human body skeleton extraction algorithm with the projection relationships between 2D and 3D spaces. Section 5.4.2 describes a kinematic model based on 3D body features. Section 5.4.3 presents the experimental details and discusses the results.

5.4.1 2D and 3D Human Body Feature Extraction

In recent years, the rapid advancements in GPU technology and multithreading-capable CPUs have led to the widespread use of deep learning approaches (Ranjan et al., 2017), such as mask regional-based convolutional neural networks (R-CNNs) (He et al., 2017), OpenPose (Cao et al., 2021), and regional multi-person pose estimation (RMPE) (Fang et al., 2017). These technologies and algorithms have facilitated the evaluation and extraction of 2D features of human postures in real-time. Fang et al. (2017) used the benchmark MPII human pose dataset to compare state-of-the-art human pose estimators based on the mean average precision (mAP) score, which indicates the accuracy of the estimation results. Table 5.2 provides an overview of these popular human pose estimators. According to Fang et al. (2017), deep-learning-based object-detection and pose-evaluation algorithms accurately obtained the 2D keypoints of human posture. Among the assessed algorithms, RMPE was the most reliable and accurate multi-person pose estimator, with an overall mPA of 80+ and a processing rate of 20+ frames per second (fps). The OpenPose algorithm had a mAP of approximately 70+ but a processing rate of only 10+ fps when implemented on the same platform (Cao et al., 2021). Overall, these deep learning approaches are highly efficient and accurate and thus suitable for real-time 2D human posture evaluation and feature extraction.

Table 5.2 Comparison of 2D human detection and tracking algorithms based on mAP scores

	Head	*Shoulder*	*Elbow*	*Wrist*	*Hip*	*Knee*	*Ankle*	*Total*
Fang et al. (RMPE)	88.4	**86.5**	**78.6**	**70.4**	**74.4**	**73.0**	**65.8**	**76.7**
Iqbal et al.	58.4	53.9	44.5	35.0	42.2	36.7	31.1	43.1
Insafutdinov et al. (DeeperCut)	78.4	72.5	60.2	51.0	57.2	52.0	45.4	59.5
Levinkov et al.	89.8	85.2	71.8	59.6	71.1	63.0	53.5	70.6
Insafutdinov et al. (ArtTrack)	88.8	87.0	75.9	64.9	74.2	68.8	60.5	74.3
Cao et al. (OpenPose)	91.2	87.6	77.7	66.8	75.4	68.9	61.7	75.6
Newell et al.	**92.1**	89.3	78.9	69.8	76.2	71.6	64.7	77.5

Considering its performance, the proposed system leverages the mature 2D body skeleton extraction algorithm, RMPE (also known as 'AlphaPose', Fang et al., 2017). The 2D body skeleton is converted to 3D body features based on the projection relationship between the 2D image space and 3D object space. RMPE is an open-source, CNN-based, single-person pose estimator method that can be applied to conventional pictorial structure models for posture estimation. RMPE is particularly suitable for real-time human body detection from RGB images. It yields a well-trained posture estimation model of the common objects in context (COCO) dataset (Fang et al., 2017), which is a benchmark dataset for training deep learning object detection algorithms and includes 17 key joint points representing human body parts (Figure 5.15). Table 5.3 lists the 17 key joint points and their corresponding human body parts.

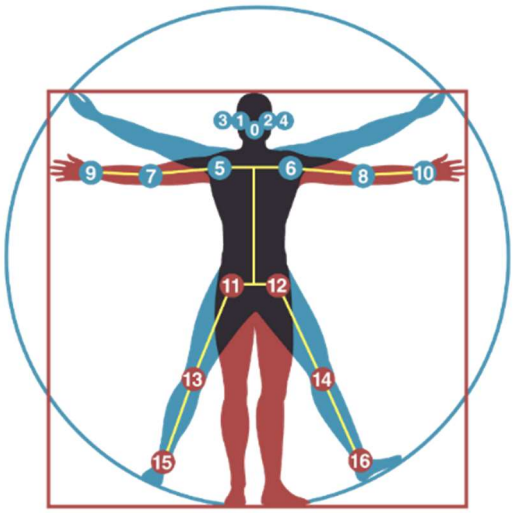


Figure 5.15 Default 2D skeleton of human body parts by RMPE

Table 5.3 Order number of human body parts

Order No.	**Body part**	*Order No.*	**Body part**
0	Nose	*9*	Left wrist
1	Left eye	*10*	Right wrist
2	Right eye	*11*	Left hip
3	Left ear	*12*	Right hip
4	Right ear	*13*	Left knee
5	Left shoulder	*14*	Right knee
6	Right shoulder	*15*	Left ankle
7	Left elbow	*16*	Right ankle
8	Right elbow		

The 2D body skeletons extracted from the images using RMPE can be expressed as in Eqs. (5.5), (5.6), and (5.7):

$$\bar{E} = \{\bar{S}_1, \bar{S}_2, \cdots, \bar{S}_k\} \tag{5.5}$$

$$S = \{j_i | 0 \leq i \leq m\}, \qquad 0 \leq m \leq 16, S \in \bar{E} \tag{5.6}$$

$$j_i = (x_i, y_i), \qquad 0 \leq i \leq m, \tag{5.7}$$

where $\bar{E}$ is a set of human body skeletons $\bar{S_\iota}(i \in 1, 2, \ldots, k\})$ of k people detected by RMPE in an image. Each skeleton S is a set of 2D joint points $j_i(i \in \{1, 2, \ldots, m\})$ that contain 2D coordinates (x_i, y_i), corresponding to the left-view image. m is the total number of body parts listed in Table 2. Each pixel in a 3D map contains both 2D image coordinates and 3D coordinates. The 2D body skeletons are converted to 3D body features by determining the 3D coordinates corresponding to the 2D joint points from the 3D map, using the 2D image coordinates as the index. This transformation follows the concept of triangulation described in Section 5.1. In this manner, a set of 3D body features containing depth information is derived and saved in the queue for further analysis. The 3D body features are combined to generate various kinematic models for evaluating and monitoring human movements, as discussed in the subsequent section.

5.4.2 Derivation of 3D Kinematic Parameters

This study focuses on typical 3D human kinematics, including the velocity of movement (both speed and direction), step length, knee flexion angle, and arm swing angles. Table 5.4 describes the considered human kinematics based on the 17 key joint points (Table 5.3).

Table 5.4 3D human kinematic measurements considered in thread 3

Name of measurement	Description	Body parts used	No. of body parts
Movement velocity	Vector quantity that measures the changes in position in a time interval, including the movement speed and direction.	Left and right hip	11, 12
Step length	Distance covered when a person starts walking and takes one step.	Left and right ankle	15, 16
Knee flexion angle	Measurement of the knee joint motion when a person moves.	Left and right hip Left and right knee Left and right ankle	11, 12 13, 14 15, 16
Arm swing angle	Essential index of the human movement pattern, including the upper-arm and elbow angles.	Left and right shoulder Left and right elbow Left and right wrist	5, 6 7, 8 9, 10

5.4.2.1 Movement Velocity Determination

Seventeen 3D human body keypoints are extracted in Thread 2. Subsequently, it is necessary to identify a suitable keypoint that can serve as the representative point for calculating the movement velocity. According to Zatsiorsky (2002), the centre of mass of an object is the ideal point for calculating velocity. In planar movement analyses, the position of the centre of mass changes with translational displacement and rotational displacement during object motion. Furthermore, in 3D movement computation, the location of the centre of mass changes with tilting displacement as well, as shown in Figure 5.16. Therefore, we categorise general locomotion in the 3D world by considering these three movement states and project them into a 2D plane to clarify the changes in the centre of mass position caused by the abovementioned displacements.

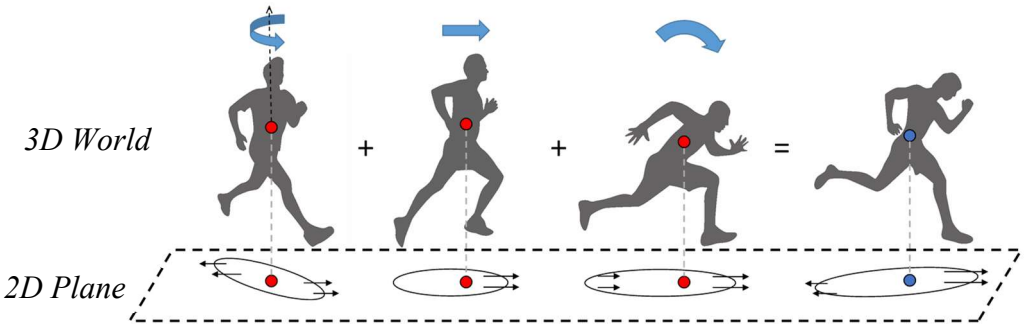


Figure 5.16 Exploded view of human locomotion velocity and centre of mass

Figure 5.16 depicts the realm of human locomotion in a 3D space, where the velocity vector is represented by a blue arrow, and the positions of the centre of mass in different stages of movement are denoted by red dots. The definitive position of the centre of mass is represented by a blue dot. During motion, the rotational velocity marginally counteracts the translational velocity, leading to a shift in the location of the centre of mass. If a tilting angular velocity is introduced, which imparts a slight forward velocity, the final position of the centre of mass will shift towards the location of the blue dot. Consequently, the blue dot is an appropriate reference point for determining the velocity of an object in both its initial and final positions. Previous research suggests that the centre of mass of the human body generally remains within or deviates slightly from the region between the left and right hips (Vlutters et al., 2016). Thus, this study considers the midpoint between the two hips as the centre of mass for velocity calculations.

The movement of an individual involves a combination of multiple variable-speed linear motions. The velocity of these motions can be calculated by considering the initial and final positions in a specific interval. Specifically, the velocity can be determined by measuring the difference in 3D hip positions in consecutive frames, based on the timestamps. The movement speed is calculated following the general principles of velocity calculation:

$$\int_{T_1}^{T_2} v(t)\,\mathrm{d}t = f(T_1) - f(T_2) \tag{5.8}$$

where $v(t)$ is the speed of a human moving from the initial position at time T_1 to the final position at time T_2, and $s(T_1) - s(T_2)$ is the displacement. In the proposed system, the speed is computed every five frames. Thus, Eq. 5.8 can be rewritten as

$$v = \left|\overrightarrow{M_s M_e}\right| \cdot \Delta t_{frame}{}^{-1} \tag{5.9}$$

where $\overrightarrow{M_s M_e}$ is the vector of the midpoint of two hip positions when an individual moves from the position in the first frame (initial position) to that in the fifth frame (final position). Δt_{frame} is the capture time interval between the previous frame and

current frame, determined considering the timestamp of each frame. In this study, the frame interval is set as five units. The human movement speed is calculated according to the abovementioned equations, and the results are stored in a queue for further visualisation.

The human centre of mass tends to remain near the midpoint of the left and right hips, with only slight deviations (Vlutters et al., 2016). Therefore, the midpoint of the left and right hips is examined to calculate the movement speed and direction. The movement speed is calculated based on the 3D coordinates of the midpoint at the initial and final positions during a time interval. Considering that movement can occur in any direction in a 360° arc, the movement direction is assessed based on trigonometric principles, starting from the direction in which the person faces the camera. The arc is partitioned to represent different directions. Specifically, the movement direction is classified as forward, backward, left, and right. In Figure 5.17, P_i (i = 0) indicates the potential initial position, and P_i (i = 1, 2, 3, 4) depicts the possible final positions in each direction within the next frame. The direction is determined by calculating the angle θ_i between the vector from the initial position to the final position and the XY-plane of the camera system, based on the 3D coordinates of the two hip positions. The step length, which is defined as the vector length from one ankle to the other, is calculated using the 3D coordinates of both ankles.

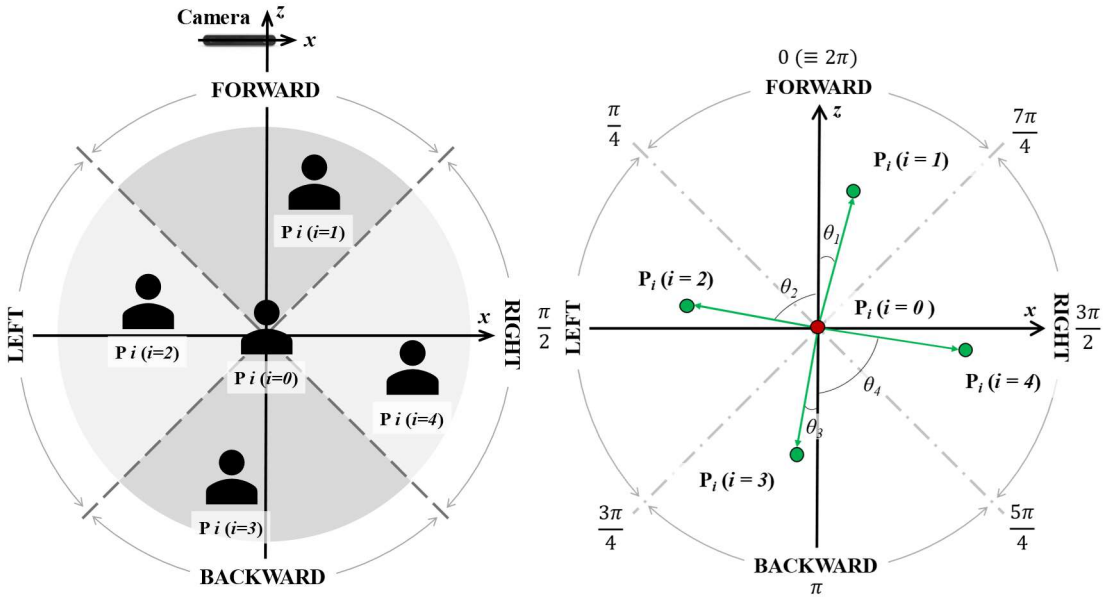


Figure 5.17 Geometric model of human movement direction. (a) Possible initial and final positions of a locomotory action. (b) Geometry between an initial position and each possible final position.

As shown in Figure 5.17, the coordinates of the human centre of mass at potential initial and final positions under the camera coordinate system are denoted as $P_i\ (x_i, z_i)$. To determine the direction of human movement, the angle between the vector extending from the initial position to the final position and the z-axis can be calculated using the (x, z) coordinates of the human centre of mass point. The computation is based on the following equations:

$$\theta_i = \arccos\frac{d(z_i, z_{5i})}{\left|\overrightarrow{P_i P_{5i}}\right|} \quad i \in \{1, 2, \cdots\} \tag{5.10}$$

$$\varphi_i = \begin{cases} \theta_i, & (x_i > x_{5i}, z_i > z_{5i}) \\ \pi - \theta_i, & (x_i > x_{5i,}\ z_i < z_{5i}) \\ \pi + \theta_i, & (x_i < x_{5i,}\ z_i > z_{5i}) \\ 2\pi - \theta_i, & (x_i < x_{5i,}\ z_i < z_{5i}) \end{cases} \tag{5.11}$$

where the constant k determines the frame interval for calculating the movement direction, and $d(z_i, z_{5i})$ is the difference in the z-values of two positions. For example, in Figure 5.17(b), if $P_i(x_i, z_i)$ is the possible initial position of the human centre of mass point in the first frame i ($i = 0$) and $P_{5i}\ (x_{5i}, z_{5i})$ is the final position of the human centre

of mass point in the fifth frame, the angle between the vector $\overrightarrow{P_l P_{5l}}$ and z-axis is θ_i (in radians). As the movement direction is defined in four directions, θ_i must be converted according to the rules in Eq. 5.11. The final angle φ_i is used to determine the movement direction. Therefore, the human is moving forward if $\varphi_i \in [0,\ \pi/4) \cup (7\pi/4, 2\pi]$, leftward if $\varphi_i \in (\pi/4, 3\pi/4)$,backward if $\varphi_i \in (3\pi/4, 5\pi/4)$, and rightward if $\varphi_i \in (5\pi/4, 7\pi/4)$.

5.4.2.2 Step Length Measurement

Step size is a precise measurement used in gait analysis to evaluate the movement and posture of individuals. This parameter varies with an individual's height, as taller individuals with longer legs tend to walk faster than those with shorter legs. Specifically, the step length refers to the distance covered when a person takes a single step, beginning from a standing position with both feet together. This distance can be expressed as the length of the vector from one ankle to the other.

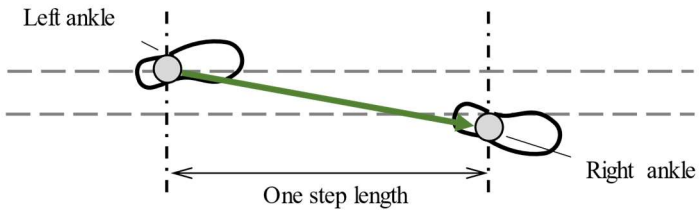


Figure 5.18 Geometric model for step length computation

$$s = \underset{s}{\operatorname{argmax}}\, f\left(\left\|\overrightarrow{A_l A_r}\right\|\right) \tag{5.12}$$

Figure 5.18 illustrates the general geometry for computing the step length. Equation 5.12 is introduced to calculate the step length, where $\overrightarrow{A_l A_r}$ denotes the vector from the left ankle to the right ankle. Therefore, the step length can be directly measured, and the resulting value is saved in the queue.

5.4.2.3 Human Joint Motion Measurement

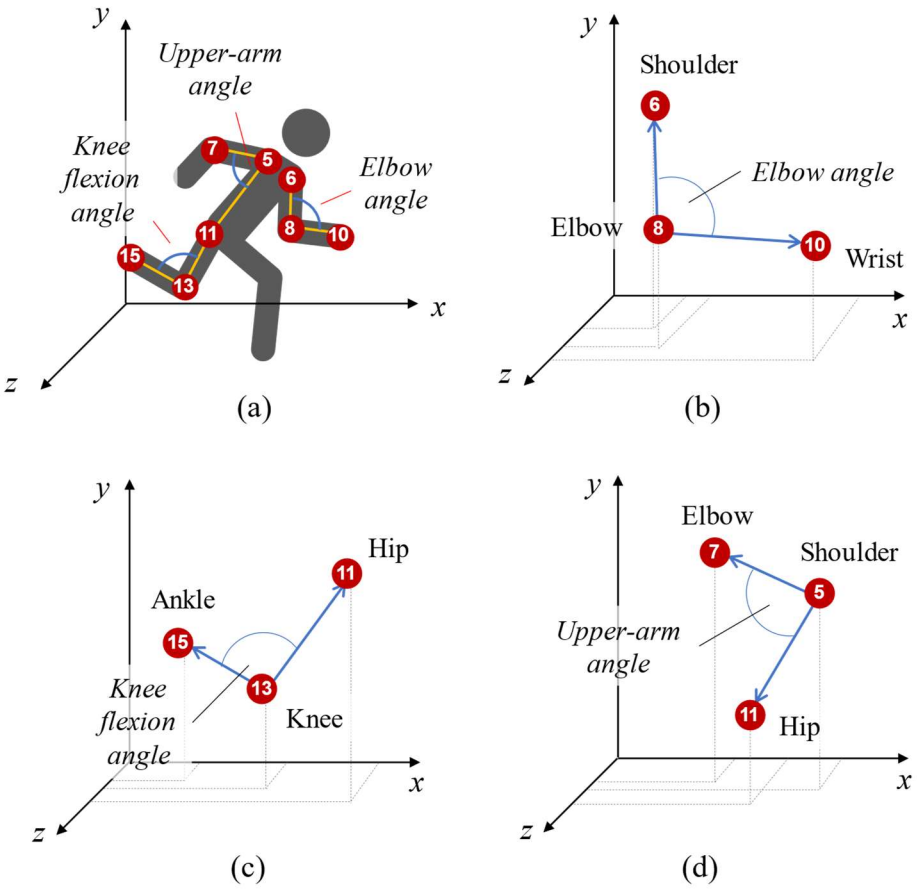


Figure 5.19 Geometric model of joint motion monitoring. (a) Body parts used in joint motion monitoring. For the corresponding order and name, refer to Table 5.3. Geometric model for calculating the (b) elbow angle, (c) knee flexion angle, and (d) upper-arm angle.

Human joint motion measurements include the knee pressure angle and arm swing angle. The arm swing angle is divided into two indices, i.e., the upper-arm angle and elbow angle. The calculation of these angles based on the geometry is shown in Figure 5.19. The knee flexion angle is calculated using the angle between the knee angle and knee–hip vectors in 3D coordinates. The upper-arm angle is the angle between the shoulder–elbow and shoulder–hip vectors. The elbow angle is calculated as the angle between the elbow–shoulder and elbow–wrist vectors.

Walking and running injuries significantly impact the measurement of knee joint motion, with several of these injuries attributable to anomalous knee motion (Lysholm and Wiklander, 1987). Consequently, 3D joint information must be used for accurately assessing the kinematics of the knee during human movement, as it can facilitate the identification of potential injuries that are affected by knee angles. In this work, six key joints within the 3D body structure are considered for calculating the knee angle. Figure 5.19(c) depicts the geometric model used to calculate the knee angle. The knee angle is calculated using Eq. 5.13.

$$cos\,\alpha_k = cos\left(\widehat{\overrightarrow{KH}, \overrightarrow{KA}}\right) = \frac{\langle \overrightarrow{KH}, \overrightarrow{KA} \rangle}{\left|\overrightarrow{KH}\right| \cdot \left|\overrightarrow{KA}\right|} \tag{5.13}$$

where $\langle a, b \rangle$ denotes the scalar product of vectors a and b. α_k is the knee angle, which represents the shortest angle between the knee–hip and knee–ankle vectors. $\overrightarrow{KH}$ is the vector from the knee to the hip, $\overrightarrow{KA}$ is the vector from the knee to the ankle, and $\overrightarrow{HA}$ is the vector from the hip to the ankle. Because the vectors change with the hip (x_h, y_h, z_h), knee (x_k, y_k, z_k), and ankle (x_a, y_a, z_a) positions during movement, the knee angle α_k varies based on the 3D coordinates of these joints. The knee joint angles of the left and right legs are calculated separately in each frame.

Arm swing is an essential component of human walking that can reduce the metabolic cost of walking and enhance gait stability (Bruijn et al., 2010). Thus, it is essential to evaluate the arm swing when analysing walking patterns. Figure 5.19(b) illustrates the geometrical model used for computing the elbow joint angle, and Figure 5.19(d) shows the angle between the upper arm and upper torso. The two indices are computed as follows.

$$\cos \alpha_e = \frac{\langle \overrightarrow{EW}, \overrightarrow{ES} \rangle}{\left|\overrightarrow{EW}\right| \cdot \left|\overrightarrow{ES}\right|} \tag{5.14}$$

$$\cos \alpha_t = \frac{\langle \overrightarrow{ES}, \overrightarrow{SH} \rangle}{\left|\overrightarrow{ES}\right| \cdot \left|\overrightarrow{SH}\right|} \tag{5.15}$$

α_e and α_t indicate the elbow angle and upper arm angle, respectively; $\overrightarrow{EW}$ is the vector from the elbow to the wrist; $\overrightarrow{ES}$ is the vector from the elbow to the shoulder; and $\overrightarrow{SH}$ is the vector from the shoulder to the hip. The left- and right-arm swing are simultaneously calculated, and the results are saved in the queue for visualisation in the final thread.

5.4.3 Implementation and Experimental Evaluation

5.4.3.1 Hardware Configuration and Multi-threading Design

The system was implemented on a computer equipped with two NVIDIA RTX 2080Ti graphics cards, 64 GB of RAM, and two 12-core CPUs. The proposed system consisted of a stereo pair of cameras, each with a horizontal field of view (FOV) of 90°, vertical FOV of 60°, and depth FOV of 100°. The image resolution of the camera was 672 × 376 pixels. The baseline distance between the left and right cameras was 120 cm. The intrinsic camera parameters, including the focal length (3.5 mm), offset of the principle point, and lens distortions, were calibrated for each camera prior to use.

The real-time photogrammetric system involved four threads, each functioning as an individual model that managed different tasks, as shown in Figure 5.20. Thread 1 loaded stereo RGB images with timestamps and known orientation parameters from the camera and applied semi-global matching (SGM) (Hirschmuller, 2007) to generate a disparity map. A 3D map was then retrieved by triangulation based on the disparities and orientation parameters of the camera. A GPU-acceleration solution was implemented in Thread 1 to increase the processing rate of 3D scene reconstruction. Thread 2 extracted 2D human body skeletons from the left-view images using RMPE and extended these skeletons to 3D body features based on the 3D map array produced by Thread 1. Thread 3 computed various human kinematic parameters, including the movement velocity, step length, and joint motion angles, based on the 3D body features. The outputs of each thread were stored in the same queue for data exchange, and the results were loaded into Thread 4 from the queue for real-time system visualisation.

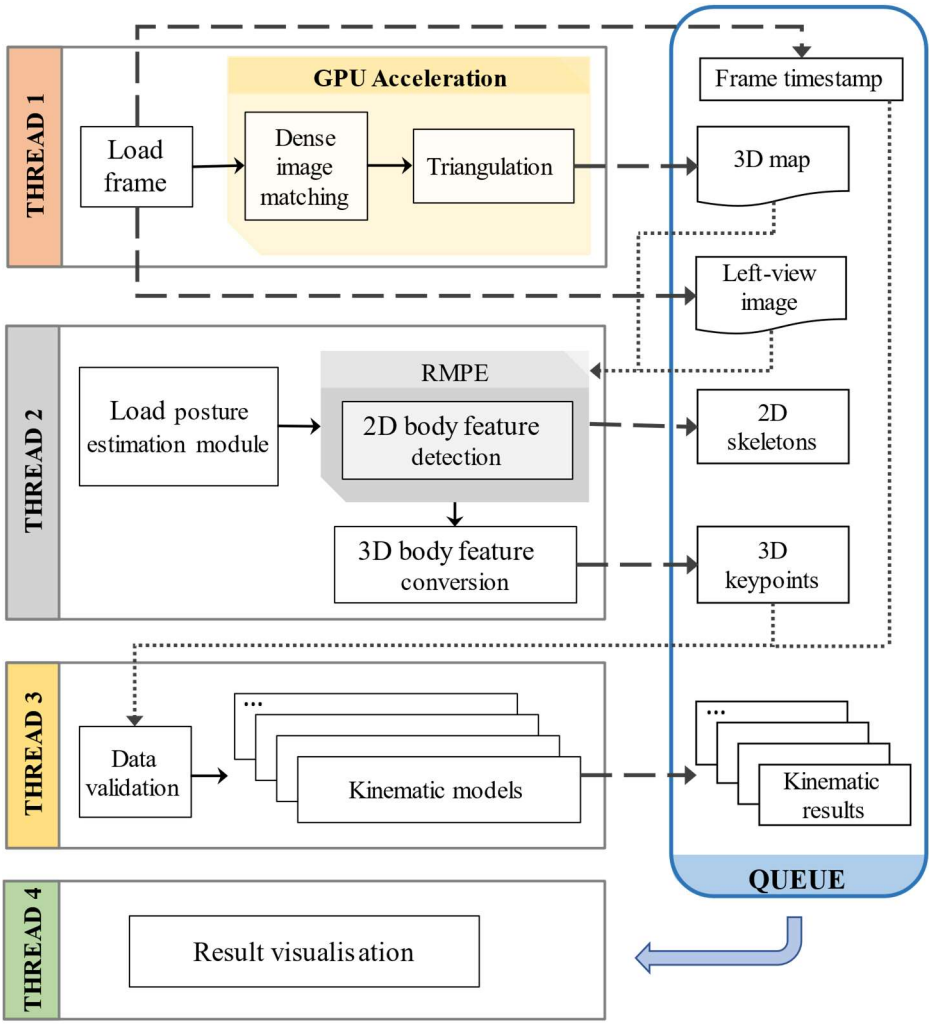


Figure 5.20 Workflow of real-time acquisition and monitoring of 3D human body kinematics

Figure 5.20 presents an overview of the system workflow. Thread 1 performed dense image matching and triangulation with GPU acceleration. GPU-based dense image matching was realised following the approach introduced in Chapter 4, and GPU-accelerated triangulation was executed following the method described in Section 5.1. The algorithm in Thread 1 involved multiple stages. First, stereo RGB images with known interior and exterior orientation parameters were imported from the stereo camera and stored in the host (CPU). Subsequently, the device (GPU) copied the stereo RGB images of the host and separated them into left-view and right-view images. The GPU with acceleration frameworks then executed the dense image matching algorithm SGM and triangulation process for reconstructing the 3D information in real-time. The left-view image and disparity map acquired from the SGM were combined to create the background image for visualisation preparation in Thread 4. Threads 1 and 2

continuously processed frames from the stereo camera. The 3D body features derived from a series of stereo camera frames were used by Thread 3 to analyse the kinematics of a 3D human body over time.

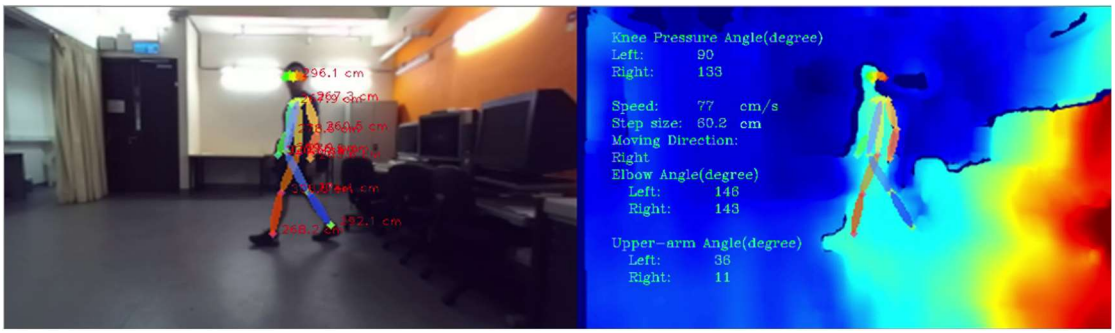


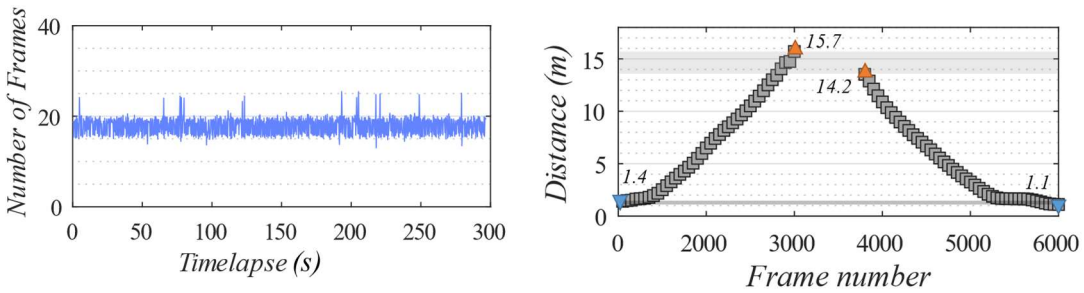
Figure 5.21 Visualisation of the real-time photogrammetric system for human kinematics

Thread 4 was responsible for visualisation. This thread loaded all of the information stored in the queue. When the queue was detected to be full of the stitching and 3D maps generated by Thread 1, the 3D body features extended from the 2D skeleton in Thread 2, and the kinematic results computed from the 3D body features in Thread 3, Thread 4 automatically displayed all of the results in a window. As shown in Figure 5.21, the background consisted of the stitching image with the left-view image of the camera and coloured disparity map. Red colours in the disparity map indicate objects closer to the camera, and darker blue colours represent objects further from the camera. Different-coloured lines connected each joint. The distance of each body joint was loaded from 3D information in the queue and displayed on the left side of the background next to each body joint, using 2D coordinates. All kinematic results were loaded from the queue and displayed on the coloured disparity map for real-time monitoring of human locomotion.

5.4.3.2 Efficiency Evaluation of System Capacity

The capabilities of the proposed system were evaluated by assessing the processing rate and effective detection distance of a person moving in front of the stereo camera. During the assessment, 6,000 frames were captured within 300 s (Figure 5.22). The threads achieved a processing rate of ~18 fps or higher, with a resolution of 1,344 ×

376 pixels. The average processing rate was 17.8 fps. Figure 5.22(a) illustrates the processing time of each frame from Thread 1 to Thread 4. The processing rate exceeded 20 fps in certain instances in which an individual's motion was exceptionally fast, resulting in a ghost effect in the frames, or when the lighting conditions were insufficient, and the person appeared faintly on the screen. In such cases, the RMPE failed to extract the 2D human skeletons, owing to which Thread 2 skipped the current frame and processed the next frame directly, resulting in a delay of several seconds. In the evaluation involving 6,000 frames (Figure 5.22(a)), the proposed system could accomplish real-time processing in nearly all instances.



(a) (b)

Figure 5.22 Results of the efficiency assessment of the real-time photogrammetric system. (a) Frame rate records. (b) Effective measurement distance assessment.

The system demonstrated an effective measurement distance of ~15 m, based on the assessment of a person moving back and forth along the optical axis of the left camera. When the person left the camera FOV and returned along the same path, the system recorded all distance values from the person's waist, defined as the midpoint between the left and right hips. As shown in Figure 5.22(b), when the person moved ~1.4 m, the system extracted the 3D body features of the left and right hips and began computing the corresponding 3D coordinates. When the person moved beyond a distance of 15.7 m from the camera, the system could not measure the distance because the person appeared too small on the screen for detection by the RMPE. As the person started moving toward the camera within a range of 14.2 m, the system could again extract the 3D human body features and simultaneously calculate the distance until the person moved to a distance of less than 1.1 m from the camera. Notably, the measurements were unstable in the distance range of 14–15 m, and the dead zone for close-range

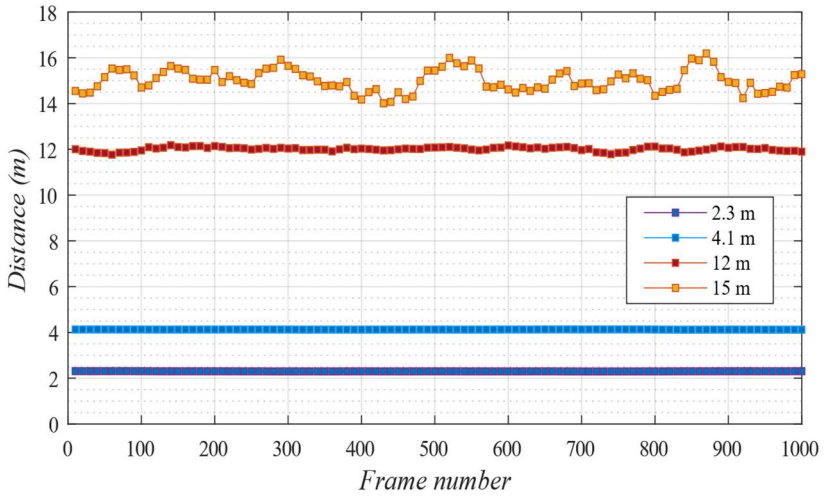
measurements was ~1 m to ~1.4 m. Thus, the effective measurement range was ~1.5 m to ~15 m, which could cover a wide range of scenes.

5.4.3.3 Accuracy Evaluation of the 3D Human Body Kinematics

To evaluate the accuracy of the distances measured by the system at a specific resolution, we conducted an experiment in which a person remained stationary in front of the camera at various distances (Figure 5.23(a)). The measurement accuracy was assessed by comparing the measured distances between the person and camera with the ground truth. As shown in Figure 5.23(b), the system captured 1,000 frames of a person standing still in front of the camera at distances of 2.3 m, 4.1 m, 12 m, and 15 m.

Table 5.5 lists the average measured values. The measured distances were close to the ground truth. When the person was 2.3 and 4.1 m from the camera, the root mean square errors (RMSEs) were 0.4 cm and 2.6 cm, respectively, corresponding to errors of 0.2% and 0.6%. As the person moved to a distance of 12 m, the measurements became unstable. The RMSE increased to 8.7 cm, and the error became 0.7%. The measurements were even more erratic when the person stood 15 m from the camera. The RMSE and error increased to 47.9 cm and 3.2%, respectively. Because the effective measurement range was ~1.5 m to 15 m, the system could not detect the person at a distance of 15 m. Nevertheless, the system could obtain 3D human body measurements with an average geometric accuracy exceeding 1% of the distance.

(a)



(b)

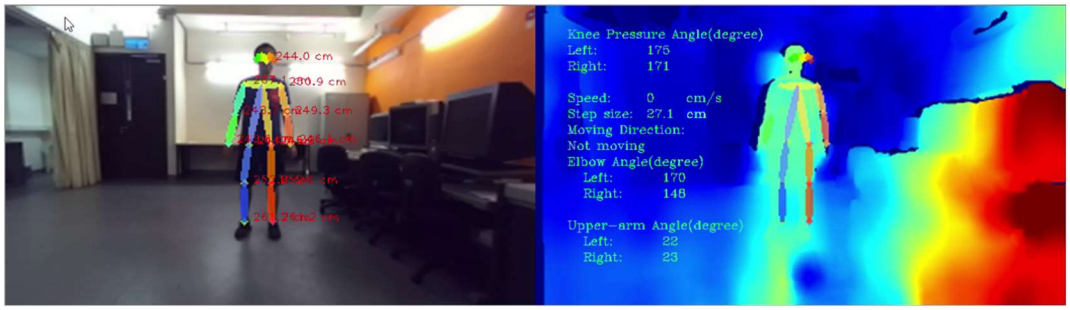
Figure 5.23 Evaluation of distance measurement accuracy. (a) A person stood stationary in front of the camera in an evaluation of the measurement accuracy. (b) Measurements of individuals standing in front of the camera at different distances.

Table 5.5 Assessment of system measurement accuracy

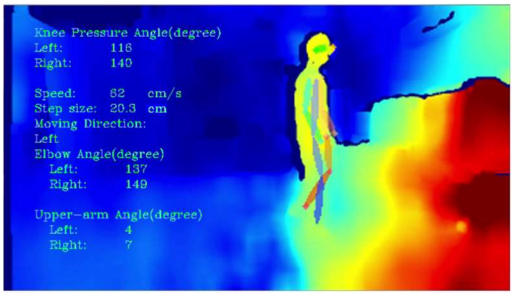
Ground truth (m)	*Mean of measurements (m)*	*RMSE (cm)*	*Error*
2.3	2.3	0.4	0.2%
4.1	4.1	2.6	0.6%
12.0	12.1	8.7	0.7%
15.0	15.1	47.9	3.2%

The human movement direction was assessed by recording a person moving in four directions relative to the camera: leftward, rightward, forward, and backward, as illustrated in Figure 5.24. Figure 5.24 (a) shows the initial position of the person, and Figures 5.24 (b), (c), (d), and (e) display the monitoring results of the person moving in four directions, with the movement speed computed in real-time. The results are

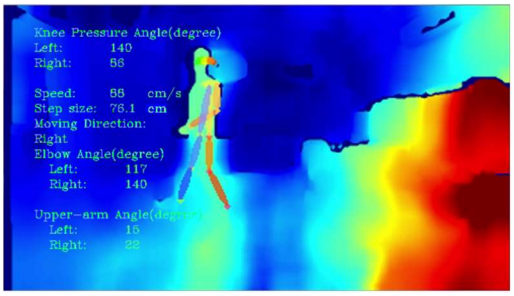
summarised in Table 5.6. The identified movement direction was consistent with the expected behaviour in each direction. In particular, the results presented in Figure 5.24 show that the system accurately identified the direction of movement and calculated the movement speed in real-time.



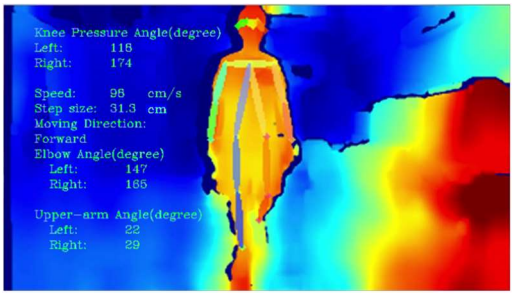
(a) Initial position of the human



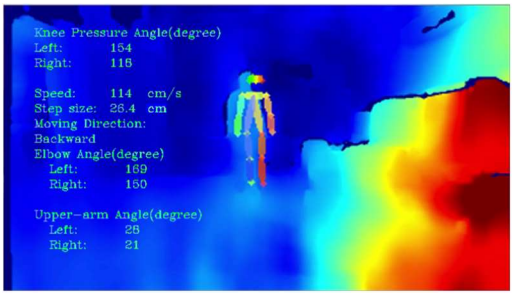
(b) Results for leftward movement



(c) Results for rightward movement



(d) Results for forward movement



(e) Results for backward movement

Figure 5.24 Results of monitoring human movement direction. The direction of movement is determined relative to the position of the camera.

Table 5.6 Movement direction identification results

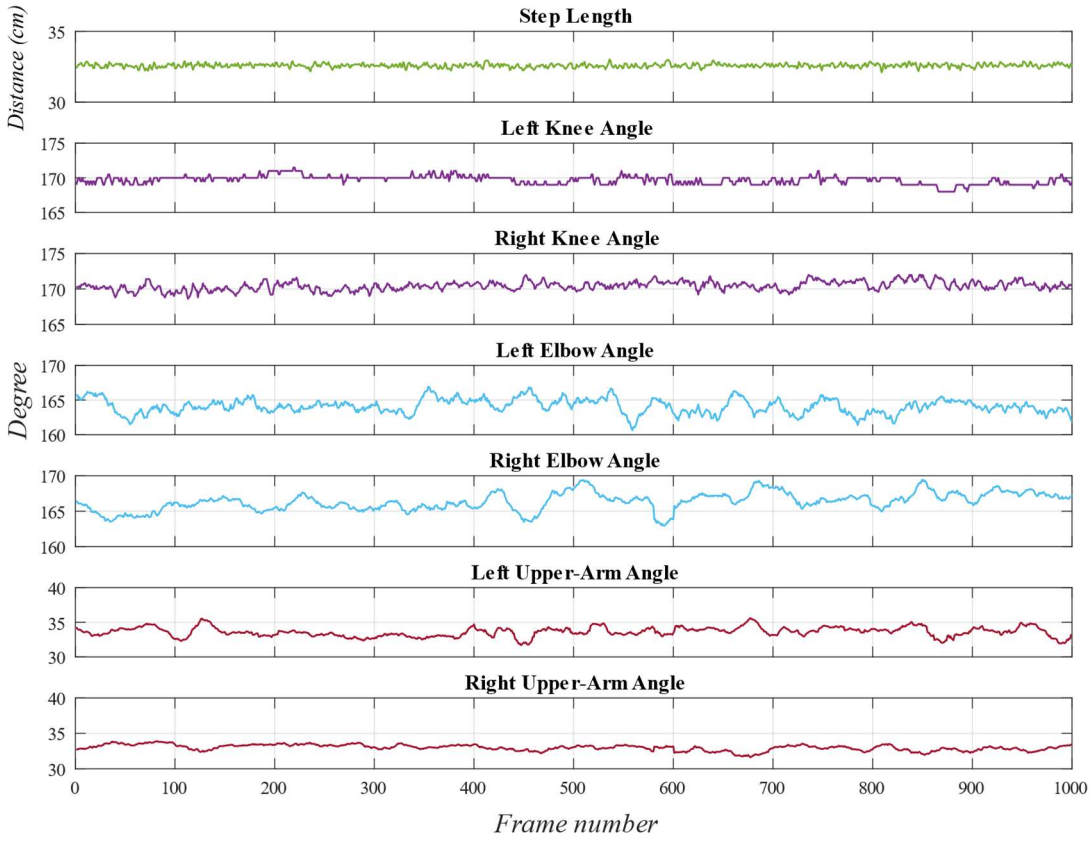
	Expected behaviour	θ (°)	Movement speed (cm/s)	Identified movement direction
(a)	The person is standing still	0.3	0	Not Moving
(b)	The person moves to their left	94.6	52	Leftward
(c)	The person moves to their right	-89.7	55	Rightward
(d)	The person moves forward	-9.2	95	Forward
(e)	The person moves backwards	-1.2	114	Backward

Table 5.7 Analysis results of kinematic applications

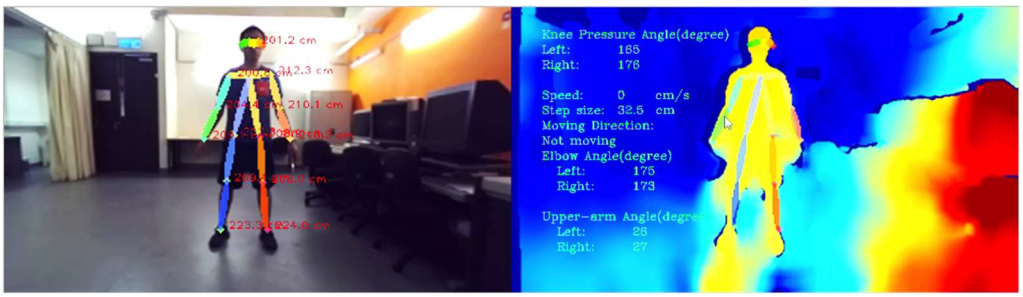
Kinematic application		Mean of measurements	Ground truth	RMSE	Error
Step length (cm)		32.6	33.1	0.3	0.8%
Knee angle (degrees)	Left	169.7	176.0	6.3	3.6%
	Right	170.4	176.0	5.7	3.2%
Elbow angle (degrees)	Left	164.1	161.0	5.4	3.4%
	Right	166.4	160.0	7.1	4.4%
Upper-arm angle (degrees)	Left	33.6	35.0	2.6	7.3%
	Right	32.9	31.0	2.3	7.5%

Table 5.7 and Figure 5.25 present the kinematic analysis results, including the step length, knee angle, elbow angle, and upper-arm swing angle, as measured and recorded from 1,000 frames involving a person standing stationary in front of the camera. For the step length, the RMSE was 0.3 cm and the error was 0.8%. The values of the left and right knee angles fluctuated slightly at approximately 170°. The RMSEs for the left and right knee angles were 6.3° and 5.7°, respectively, and the error was approximately 3%. The measurements of the elbow and upper-arm angle were unstable due to the bright light at their positions in the image. This instability resulted in the RMPE being unable to accurately extract the 2D features and convert the 3D features. The mean values of the measured left and right elbow angles were 164.1° and 166.4°, respectively, with RMSEs of 5.4° and 7.1° and an error within 5%. Overall, the measured values could be considered stable. The RMSEs of the left and right upper-arm angles were 2.6°

and 2.3°, respectively, with errors of 7.3% and 7.5%. These results indicate that the measured angles deviated slightly from the ground truth values.



(a)



(b)

Figure 5.25 Analysis of kinematic measurements by the system. (a) 1,000-frame measurements of the step length, knee flexion angles, and arm swing angles. (b) System-measured kinematics of a person standing still in front of the camera.

Chapter 6 Conclusions and Discussions

This thesis provides a complete framework for achieving real-time photogrammetry by presenting novel approaches and implementations of cross-view feature matching for camera pose determination, dense image matching, and GPU-accelerated triangulation for real-time 3D data generation. This chapter summarises the achievements, draws conclusions from this research and then makes recommendations for future research.

6.1 Summary of the Research Work and Conclusions

Existing techniques for implementing real-time photogrammetry algorithms are limited by high computational requirements and by processing large datasets in various scenes to generate 3D data. These limitations have been discussed in the previous chapters. This study addresses these limitations using a parallel architecture to accelerate computational efficiency as a promising solution for implementing real-time photogrammetry. The evaluation results demonstrate the effectiveness, efficiency, and applicability of the proposed algorithms and techniques in various situations. This study's results can advance real-time photogrammetry applications such as accurate aerial visual navigation, real-time aerial 3D modelling, and human motion tracking and analysis. The following summarises the effectiveness and experimental results in achieving real-time processing at each stage:

First, a novel visual-based approach for real-time camera pose determination was achieved (Chapter 3). The approach utilises a coarse-to-fine strategy involving feature-based cross-view image matching, retrieval, and camera pose determination using VO and space resection techniques. It provided a robust and accurate camera pose estimation by combining the strengths of both techniques. Space resection offered global accuracy based on ground control points, while VO provided incremental updates for continuous tracking. The experiment result indicates that the proposed approach successfully localises the aerial robot by narrowing the region for visual positioning and then precisely identifying its location and orientation through visual odometry and space resection. The proposed method not only achieves the implementation of photogrammetric algorithms into real-time localisation algorithms,

but also offers a cost-effective alternative to traditional navigation methods and provides a flexible solution for accurate visual navigation in a GPS-denied environment.

Secondly, a parallel architecture SGM algorithm was developed and implemented for real-time dense image matching in photogrammetric applications (Chapter 4). The proposed method aims to improve the efficiency of dense image matching, enabling real-time processing in various scenarios, such as the matching cost (MC) and similarity measures selection. The overall processing efficiency is significantly improved by utilising a parallel architecture to increase the computing speed of MC and similarity measures so that it can be adapted to real-time dense image matching. Two applications are considered to evaluate the real-time processing efficiency of the parallel-architecture SGM: real-time generation of disparity maps using stereo images and aerial images. The evaluation results demonstrate that the accuracy of the proposed method outperforms traditional SGM methods and achieves real-time processing efficiency. It also validates that the proposed method can be suitable for disparity map generation for various large-scale scenarios. In conclusion, this research contributes to the achievement of real-time processing of dense matching algorithms in photogrammetry, offering potential benefits for photogrammetry and related fields.

Third, the photogrammetry approaches were extended and implemented to various relative fields of real-time 3D data generation and applications (Chapter 5). One application is the implementation of GPU-accelerated triangulation algorithm incorporating the ROS system to generate 3D point clouds from aerial images in real-time. This application also proposes a new method of point cloud storage and visualisation. The method uses a revised library to incrementally process incoming frames and pass them to the next stage as soon as possible to achieve real-time visualisation of the point cloud. Experimental results indicate that the 3D data is incrementally generated and visualised using the ROS system with multi-threading capabilities. Ultimately, this application achieves the real-time acquisition of detailed 3D point clouds that accurately represent the environment from aerial images. Another application that benefits from photogrammetry techniques and GPU acceleration is the real-time acquisition and monitoring of 3D human body kinematics. This application uses a deep learning approach to extract human features, which are then converted into a corresponding 3D representation by the GPU-accelerated disparity mapping method

presented in Chapter 3. The experimental results quantitatively evaluate the efficiency and accuracy of each measurement for human kinematic analysis and demonstrate that this approach enables the real-time measurement and visualisation of various kinematic parameters related to human motion. The contribution of this work is not only achieving real-time triangulation algorithm for 3D point cloud generation, but also integrating the proposed real-time photogrammetric methods in the previous chapters to achieve real-time 3D application in different scenarios. This work provides a reference and pavement for the real-time 3D photogrammetry applications in the future.

Overall, this research accomplished the stated objectives of achieving real-time photogrammetry by presenting novel techniques and their implementations. The experimental evaluations demonstrate the feasibility and effectiveness of the proposed approaches in real-world scenarios.

6.2 Discussions and Future Works

While the presented approach offers advantages over traditional methods, it may have certain limitations. The limitations could include:

a) Real-time cross-view feature matching and camera pose determination

In the real-time camera pose determination (Chapter 3), the method depends on a pre-built database consisting of cropped orthoimage and DSM tiles with features extracted using a deep learning algorithm. The accuracy and effectiveness of the approach heavily rely on the quality and completeness of this database. Currently, the orthoimages and DSMs in the dataset are cropped using a fixed scale, which must align with the scale of the aerial images captured by the camera mounted on the UAV. This alignment is crucial to ensure the robustness of cross-view image matching, which impacts the accuracy of camera pose determinations. However, the fixed-scale cropping approach poses challenges when variations in the camera's scale vary across different situations. It is imperative to enhance the cropping scheme by implementing alternative methods to address this limitation. One potential improvement is adopting an adaptive scale cropping technique, which dynamically adjusts the cropping of orthoimages and DSMs

to match the specific camera scale during the resumption of the pre-built database. Another possible approach is utilising a content-aware cropping scheme that intelligently analyses the image content to determine the optimal cropping parameters.

Another limitation of the current camera pose refinement method is that it solely relies on space resection. Additionally, this method requires prior knowledge of the world space coordinates of GCPs to accurately determine the camera pose by solving co-linear equations. While this approach allows for obtaining the real-world coordinates of the camera, its feasibility is constrained to environments with a DSM. Consequently, it is essential to explore alternative methods in future work that can achieve accurate camera pose estimation in GPS-denied environments.

A further constraint arises when the algorithm reaches a specific number of input images, at which point it becomes unable to process additional images due to the memory constraints of the on-board computer. To address this challenge, the current approach involves only keeping the feature points extracted by the algorithm without saving the images to reduce the load on memory and storage. Our proposed cross-view image matching and retrieval method also uses only the feature points to determine the position of the aerial image. However, it is crucial to acknowledge a potential constraint in this approach when the same image is needed and reused by the following algorithms for subsequent image processing. Therefore, based on the above analysis, the method for effective image processing needs to be optimised in future work.

(b) Real-time dense image matching based on GPU-acceleration

In Section 4.2 of Chapter 4, the SGM framework for dense image matching is introduced, discussing issues related to processing efficiencies, such as the MC and the selection of similarity measures. However, some limitations need to be addressed. First, implementing a parallel-architecture SGM algorithm may require specialised hardware, potentially limiting its adoption on devices with limited computational capabilities. Secondly, scalability could be challenging when dealing with extremely large-scale aerial images or real-time simultaneous processing of multiple image streams. Further research is needed to optimise the algorithm for such scenarios. Nevertheless, the algorithm's performance might vary in environments such as underwater, in urban

canyons, or heavily occluded scenes. Future work could focus on adapting the algorithm for optimal performance in these scenarios. Section 4.4 evaluates the real-time processing efficiency of the parallel-architecture SGM algorithm. Firstly, images taken by a stereo camera are used to assess the real-time depth map generation. Then, the algorithm is tested using large-scale aerial images collected from a UAV platform. The real-time processing and evaluation results are presented, analysed, and discussed. This evaluation helps identify the algorithm's strengths and weaknesses in generating depth maps in real-time scenarios.

Future research could further concentrate on optimising the algorithm, exploring advanced data structures, parallelisation techniques, and integrating machine learning approaches. Cross-platform compatibility should also be considered to ensure broader accessibility. Improving hardware capabilities, such as faster GPUs and specialised parallel processing units, could enhance real-time performance. Additionally, adapting the algorithm to various environments and automating parameter tuning would increase its robustness and usability. Furthermore, the parallel-architecture SGM algorithm can be integrated with other technologies, such as SLAM or AI-based vision systems, to expand its capabilities and enable more sophisticated applications. Overall, developing and implementing a parallel-architecture SGM algorithm for real-time dense image matching show promising results and present an exciting area for future research in photogrammetric applications.

(c) Real-time 3D data generation and applications

In Section 5.3, the real-time 3D point cloud generation has several limitations and potential areas for future work. Firstly, the processing efficiency of the system might vary depending on the complexity of the scene and the available computational resources. Further optimisation of the camera pose estimation, triangulation, and interpolation processes is necessary to improve overall processing speed, ensuring real-time performance in various scenarios. Another limitation lies in the conversion from sparse to dense point clouds. The effectiveness of the interpolation process in accurately representing the 3D scene relies on the density and distribution of the sparse points. In scenarios with limited coverage or challenging geometries, achieving a high-quality, dense point cloud representation may prove challenging. Future work could focus on

developing advanced interpolation techniques or incorporating additional information and sensor modalities to enhance the accuracy and density of the resulting dense point cloud. The method we propose is only simulated on the computer, and in the future, we hope to put it on the UAV to process it in real time while flying.

Several limitations could be noticed in the experiments for real-time acquisition and monitoring of 3D human body kinematics in Section 5.4. The RMPE failed to detect the 2D human features when the person moved very fast (a ghosting effect appeared on the screen) or when the illumination was dark (the person almost disappeared). Similarly, the light intensity in the environment was not constant, and the SGM did not accurately obtain the disparity value in a very highlighting environment, such as an area near a lamp, or low-illumination environments, such as shadows. The 3D information was not extracted in these cases. Moreover, 3D body features were not extracted over a certain distance, where the person was so small in the image that the 2D human detection algorithm was unable to extract human skeletons. These problems can be improved by optimising the algorithms to support higher image resolutions. The clearer outline of a person in a higher-resolution image allows the deep learning method to recognise the body features at farther distances. With proper optimisation in our future works, real-time processing of image sequences of higher resolutions can be expected. It should also be noted that the moving directions that can be identified in the current system only allow four main directions. The algorithms will be further improved in our future works to allow the identification of more sophisticated moving directions.

This thesis provides a complete framework for achieving real-time photogrammetry with various applications. The findings of this research have the potential to advance real-time photogrammetry applications in areas such as aerial imagery analysis, 3D modelling, and human motion tracking. Further research and development in these areas can build upon the foundations laid out in this thesis to continue advancing real-time photogrammetry algorithms and their practical applications.

Reference

Abdi, G., Samadzadegan, F., Kurz, F., 2016. Pose estimation of unmanned aerial vehicles based on a vision-aided multi-sensor fusion, XXII ISPRS Congress, Technical Commission I, pp. 193-199.

Afriansyah, F.L., Muna, N., Widiastuti, I., Fanani, N.Z., Purnomo, F.E., 2019. Image mapping detection of green areas using speed up robust features, 2019 International Conference on Computer Science, Information Technology, and Electrical Engineering (ICOMITEE). IEEE, pp. 165-168.

Agisoft, L., 2014. Agisoft Photoscan User Manual: Professional Edition. URL: https://www.agisoft.com/pdf/photoscan-pro_1_4_en.pdf (accessed 2023 Jan. 14).

Alhwarin, F., Wang, C., Ristić-Durrant, D., Gräser, A., 2008. Improved SIFT-features matching for object recognition, Visions of Computer Science-BCS International Academic Conference, pp. 179-190.

Allaire, S., Kim, J.J., Breen, S.L., Jaffray, D.A., Pekar, V., 2008. Full orientation invariance and improved feature selectivity of 3D SIFT with application to medical image analysis, 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops. IEEE, pp. 1-8.

Aqel, M.O., Marhaban, M.H., Saripan, M.I., Ismail, N.B., 2016. Review of visual odometry: Types, approaches, challenges, and applications. SpringerPlus, 5, pp. 1-26.

Arm Limited, 2023. Learn the architecture - Introducing the ARM architecture. URL: https://developer.arm.com/documentation/102404/latest/ (accessed 2023 Jan. 10).

Arras, K.O., Siegwart, R.Y., 1998. Feature extraction and scene interpretation for map-based navigation and map building. Mobile Robots XII. SPIE, pp. 42-53.

Balali, V., Noghabaei, M., Heydarian, A., Han, K., 2018. Improved stakeholder communication and visualizations: Real-time interaction and cost estimation within immersive virtual environments. Construction Research Congress 2018, pp. 522-530.

Balntas, V., Lenc, K., Vedaldi, A., Mikolajczyk, K., 2017. HPatches: A benchmark and evaluation of handcrafted and learned local descriptors. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5173-5182.

Barbasiewicz, A., Widerski, T., Daliga, K., 2018. The analysis of the accuracy of spatial models using photogrammetric software: Agisoft Photoscan and Pix4D, E3S Web of Conferences. EDP Sciences, pp. 12.

Bates, D., Eddelbuettel, D., 2013. Fast and elegant numerical linear algebra using the RcppEigen package. Journal of Statistical Software, 52, pp. 1-24.

Bentley, J.L., 1975. Multidimensional binary search trees used for associative searching. Communications of the ACM, 18(9), pp. 509-517.

Besl, P.J., McKay, N.D., 1992. Method for registration of 3D shapes. Sensor fusion IV: Control Paradigms and Data Structures. SPIE, pp. 586-606.

Bradski, G., 2000. The OpenCV library. Dr. Dobb's Journal: Software Tools for the Professional Programmer, 25(11), pp. 120-123.

Bradski, G., Kaehler, A., 2008. Learning OpenCV: Computer vision with the OpenCV library. O'Reilly Media, Inc., Sebastopol, US, ISBN: 9788184045970.

Briechle, K., Hanebeck, U.D., 2001. Template matching using fast normalized cross correlation. Optical Pattern Recognition XII. SPIE, pp. 95-102.

Bruijn, S.M., Meijer, O.G., Beek, P.J., Van Dieen, J.H., 2010. The effects of arm swing on human gait stability. Journal of Experimental Biology, 213(23), pp. 3945-3952.

Bu, S., Zhao, Y., Wan, G., Liu, Z., 2016. Map2Dfusion: Real-time incremental UAV image mosaicing based on monocular SLAM. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp. 4564-4571.

Buttinger-Kreuzhuber, A., Konev, A., Horváth, Z., Cornel, D., Schwerdorf, I., Blöschl, G., Waser, J., 2022. An integrated GPU-accelerated modeling framework for high-resolution simulations of rural and urban flash floods. Environmental Modelling & Software, 156, 105480.

Calonder, M., Lepetit, V., Strecha, C., Fua, P., 2010. Brief: Binary robust independent elementary features. Computer Vision - ECCV 2010: 11th European Conference on Computer Vision, Part IV 11, pp. 778-792.

Campbell, J., Sukthankar, R., Nourbakhsh, I., Pahwa, A., 2005. A robust visual odometry and precipice detection system using consumer-grade monocular

vision. Proceedings of the 2005 IEEE International Conference on Robotics and Automation. IEEE, pp. 3421-3427.

Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E., Sheikh, Y., 2021. OpenPose: Realtime multi-person 2D pose estimation using part affinity fields. IEEE Transactions on Pattern Analysis and Machine Intelligence, 43(1), pp. 172-186.

Chaivivatrakul, S., Moonrinta, J., Dailey, M.N., 2010. Towards automated crop yield estimation-detection and 3D reconstruction of pineapples in video sequences. VISAPP, 1, pp. 180-183.

Chang, J.-R., Chen, Y.-S., 2018. Pyramid stereo matching network. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5410-5418.

Chauhan, M.S., Singh, A., Khemka, M., Prateek, A., Sen, R., 2019. Embedded CNN based vehicle classification and counting in non-laned road traffic. Proceedings of the 10th International Conference on Information and Communication Technologies and Development, pp. 1-11.

Chen, L., Wu, B., Zhao, Y., Li, Y., 2021. A real-time photogrammetric system for acquisition and monitoring of three-dimensional human body kinematics. Photogrammetric Engineering & Remote Sensing, 87(5), pp. 363-373.

Chen, W., Liao, X., Sun, Y., Wang, Q., 2020. Improved ORB-SLAM based 3D dense reconstruction for monocular endoscopic image. 2020 International Conference on Virtual Reality and Visualization (ICVRV). IEEE, pp. 101-106.

Chen, Y., Chen, Y., Wang, G., 2019. Bundle adjustment revisited. arXiv preprint arXiv:1912.03858.

Chen, Y., Jiang, H., Li, C., Jia, X., Ghamisi, P., 2016. Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. IEEE Transactions on Geoscience and Remote Sensing, 54(10), pp. 6232-6251.

Chhabra, P., Garg, N.K., Kumar, M., 2020. Content-based image retrieval system using ORB and SIFT features. Neural Computing and Applications, 32, pp. 2725-2733.

Choudhary, S., Gupta, S., Narayanan, P., 2012. Practical time bundle adjustment for 3D reconstruction on the GPU, Trends and Topics in Computer Vision: ECCV 2010 Workshops, Part II 11, pp. 423-435.

Choy, C.B., Gwak, J., Savarese, S., Chandraker, M., 2016. Universal correspondence network. Advances in Neural Information Processing Systems, 29.

Ciregan, D., Meier, U., Schmidhuber, J., 2012. Multi-column deep neural networks for image classification, 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, pp. 3642-3649.

Cover, T., Hart, P., 1967. Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13(1), pp. 21-27.

Daud, S.M.S.M., Yusof, M.Y.P.M., Heo, C.C., Khoo, L.S., Singh, M.K.C., Mahmood, M.S., Nawawi, H., 2022. Applications of drone in disaster management: A scoping review. Science & Justice, 62(1), pp. 30-42.

Deng, S., Dong, Q., Liu, B., Hu, Z., 2022. SuperPoint-guided semi-supervised semantic segmentation of 3D point clouds. 2022 International Conference on Robotics and Automation (ICRA). IEEE, pp. 9214-9220.

Deng, Z.-A., Xu, Y.-B., Ma, L., 2012. Indoor positioning via nonlinear discriminative feature extraction in wireless local area network. Computer Communications, 35(6), pp. 738-747.

Deshmukh, R., Roros, C.J., Kashyap, A., Kak, A.C., 2023. An aligned multi-temporal multi-resolution satellite image dataset for change detection research. arXiv preprint arXiv:2302.12301.

DeTone, D., Malisiewicz, T., Rabinovich, A., 2018. SuperPoint: Self-supervised interest point detection and description. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 224-236.

Diskin, Y., Asari, V.K., 2013. Dense 3D point-cloud model using optical flow for a monocular reconstruction system. 2013 IEEE Applied Imagery Pattern Recognition Workshop (AIPR). IEEE, pp. 1-6.

Du, P., Zhou, Y., Xing, Q., Hu, X., 2011. Improved SIFT matching algorithm for 3D reconstruction from endoscopic images. Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry, pp. 561-564.

Dusmanu, M., Rocco, I., Pajdla, T., Pollefeys, M., Sivic, J., Torii, A., Sattler, T., 2019. D2-net: A trainable CNN for joint detection and description of local features. arXiv preprint arXiv:1905.03561.

Engel, J., Schöps, T., Cremers, D., 2014. LSD-SLAM: Large-scale direct monocular slam, Computer Vision - ECCV 2014: 13th European Conference, Part II 13, pp. 834-849.

Fan, B., Kong, Q., Wang, X., Wang, Z., Xiang, S., Pan, C., Fua, P., 2019. A performance evaluation of local features for image-based 3D reconstruction. IEEE Transactions on Image Processing, 28(10), pp. 4774-4789.

Fang, H.-S., Xie, S., Tai, Y.-W., Lu, C., 2017. RMPE: Regional multi-person pose estimation. Proceedings of the IEEE International Conference on Computer Vision, pp. 2334-2343.

Fankhauser, P., Hutter, M., 2016. A universal grid map library: Implementation and use case for rough terrain navigation. Robot Operating System (ROS) The Complete Reference, 1, pp. 99-120.

Feng, H., Zhang, G., Hu, B., Zhang, X., Li, S., 2019. Noise-resistant matching algorithm integrating regional information for low-light stereo vision. Journal of Electronic Imaging, 28(1), 013050.

Förstner, W, 2005. Real-Time Photogrammetry. Proceedings of the Photogrammetric Week, 5, pp. 229-238.

Förstner, W., Wrobel, B.P., 2016. Photogrammetric computer vision: Statistics, Geometry, Orientation and Reconstruction. Springer, Switzerland, ISBN: 9783319115504.

Foster, I., Kesselman, C., 2003. The Grid 2: Blueprint for a new computing infrastructure. Elsevier, San Francisco, US, ISBN: 9781558609334.

Fraundorfer, F., Scaramuzza, D., 2011. Visual odometry: Part I: The first 30 years and fundamentals. IEEE Robotics and Automation Magazine, 18(4), pp. 80-92.

Frosi, M., Gobbi, V., Matteucci, M., 2023. OSM-SLAM: Aiding SLAM with OpenStreetMap priors. Frontiers in Robot and AI, 10, 1064934.

Fu, Q., Tong, X., Liu, S., Ye, Z., Jin, Y., Wang, H., Hong, Z., 2023. GPU-accelerated PCG method for the block adjustment of large-scale high-resolution optical satellite imagery without GCPs. Photogrammetric Engineering & Remote Sensing, 89(4), pp. 211-220.

Gálvez-López, D., Tardos, J.D., 2012. Bags of binary words for fast place recognition in image sequences. IEEE Transaction on Robot, 28(5), pp. 1188-1197.

Geiger, A., Lenz, P., Stiller, C., Urtasun, R., 2013. Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research, 32(11), pp. 1231-1237.

Geiger, A., Ziegler, J., Stiller, C., 2011. Stereoscan: Dense 3D reconstruction in real-time. 2011 IEEE Intelligent Vehicles Symposium (IV). IEEE, pp. 963-968.

Gholami, M., Rezaei, A., Cuthbert, T.J., Napier, C., Menon, C., 2019. Lower body kinematics monitoring in running using fabric-based wearable sensors and deep convolutional neural networks. Sensors, 19(23), pp. 5325.

Gonzalez, R., Rodriguez, F., Guzman, J.L., Pradalier, C., Siegwart, R., 2012. Combined visual odometry and visual compass for off-road mobile robots localization. Robotica, 30(6), pp. 865-878.

Grazioso, S., Caporaso, T., Selvaggio, M., Panariello, D., Ruggiero, R., Di Gironimo, G., 2019. Using photogrammetric 3D body reconstruction for the design of patient - tailored assistive devices. 2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4. 0&IoT). IEEE, pp. 240-242.

Gruen, A., 2012. Development and status of image matching in photogrammetry. The Photogrammetric Record, 27(137), pp. 36-57.

Hartley, R., Zisserman, A., 2003. Multiple view geometry in computer vision. Cambridge University Press, Cambridge, UK, ISBN: 9781139449144.

He, K., Gkioxari, G., Dollár, P., Girshick, R., 2017. Mask r-CNN. Proceedings of the IEEE International Conference on Computer Vision, pp. 2961-2969.

Hermann, S., Morales, S., Klette, R., 2011. Half-resolution semi-global stereo matching, 2011 IEEE Intelligent Vehicles Symposium (IV). IEEE, pp. 201-206.

Hernandez-Juarez, D., Chacón, A., Espinosa, A., Vázquez, D., Moure, J.C., López, A.M., 2016. Embedded real-time stereo estimation via semi-global matching on the GPU. Procedia Computer Science, 80, pp.143-153.

Hinzmann, T., Schönberger, J.L., Pollefeys, M., Siegwart, R., 2018. Mapping on the fly: Real-time 3D dense reconstruction, digital surface map and incremental orthomosaic generation for unmanned aerial vehicles. Field and Service Robotics: Results of the 11th International Conference, pp. 383-396.

Hirschmuller, H., 2005. Accurate and efficient stereo processing by semi-global matching and mutual information. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). IEEE, pp. 807-814.

Hirschmuller, H., 2007. Stereo processing by semiglobal matching and mutual information. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(2), pp. 328-341.

Hirschmüller, H., Innocent, P.R., Garibaldi, J., 2002. Real-time correlation-based stereo vision with reduced border errors. International Journal of Computer Vision, 47, pp. 229-246.

Hirschmuller, H., Scharstein, D., 2008. Evaluation of stereo matching costs on images with radiometric differences. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(9), pp. 1582-1599.

Howard, A., 2008. Real-time stereo visual odometry for autonomous ground vehicles, 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp. 3946-3952.

Hu, Y., 2015. Research on three dimensional reconstruction of the ancient building based on images. Computer and Computing Technologies in Agriculture VIII: 8th IFIP WG 5.14 International Conference, CCTA 2014, pp. 73-79.

Hua, S., Chen, G., Wei, H., Jiang, Q., 2012. Similarity measure for image resizing using SIFT feature. EURASIP Journal on Image and Video Processing 2012, 1, pp. 1-11.

Hua, Y., Lin, J., Lin, C., 2010. An improved SIFT feature matching algorithm. 2010 8th World Congress on Intelligent Control and Automation. IEEE, pp. 6109-6113.

Intel Corporation, 2018. Intel® Aero compute board. URL: https://www.intel.com/content/www/us/en/products/sku/97178/intel-aero-compute-board/specifications.html (accessed 2022 Oct. 04).

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., 2011. Kinectfusion: Real-time 3D reconstruction and interaction using a moving depth camera. Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, pp. 559-568.

Jaimes, A., Sebe, N., 2007. Multimodal human–computer interaction: A survey. Computer vision and image understanding, 108(1-2), pp. 116-134.

Jiménez-Jiménez, S.I., Ojeda-Bustamante, W., Marcial-Pablo, M.d.J., Enciso, J., 2021. Digital terrain models generated with low-cost UAV photogrammetry: Methodology and accuracy. ISPRS International Journal of Geo-Information, 10(5), pp. 285.

Johnson, J., Douze, M., Jégou, H., 2019. Billion-scale similarity search with GPUs. IEEE Transactions on Big Data, 7(3), pp. 535-547.

Junior, J.C.S.J., Musse, S.R., Jung, C.R., 2010. Crowd analysis using computer vision techniques. IEEE Signal Processing Magazine, 27(5), pp. 66-77.

Kanade, T., Kano, H., Kimura, S., Yoshida, A., Oda, K., 1995. Development of a video-rate stereo machine. Proceedings 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots. IEEE, pp. 95-100.

Kang, J., Chen, L., Deng, F., Heipke, C., 2019. Context pyramidal network for stereo matching regularized by disparity gradients. ISPRS Journal of Photogrammetry and Remote Sensing, 157, pp. 201-215.

Karunarathne, M.S., Jones, S.A., Ekanayake, S.W., Pathirana, P.N., 2014. Remote monitoring system enabling cloud technology upon smart phones and inertial sensors for human kinematics. 2014 IEEE 4th International Conference on Big Data and Cloud Computing. IEEE, pp. 137-142.

Kendall, A., Cipolla, R., 2016. Modelling uncertainty in deep learning for camera relocalization. 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, pp. 4762-4769.

Kendall, A., Grimes, M., Cipolla, R., 2015. PoseNet: A convolutional network for real-time 6-dof camera relocalization. Proceedings of the IEEE International Conference on Computer Vision, pp. 2938-2946.

Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., Bry, A., 2017. End-to-end learning of geometry and context for deep stereo regression. Proceedings of the IEEE International Conference on Computer Vision, pp. 66-75.

Kern, A., Bobbe, M., Khedar, Y., Bestmann, U., 2020. OpenREALM: Real-time mapping for unmanned aerial vehicles. 2020 International Conference on Unmanned Aircraft Systems (ICUAS). IEEE, pp. 902-911.

Kim, H., Oh, T., Lee, D., Myung, H., 2014. Image-based localization using prior map database and monte carlo localization, 2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). IEEE, pp. 308-310.

Kim, S., Kim, T., Sim, J., 2019. Applicability assessment of UAV mapping for disaster damage investigation in Korea. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences, 42, pp. 209-214.

Kitt, B.M., Rehder, J., Chambers, A.D., Schonbein, M., Lategahn, H., Singh, S., 2011. Monocular visual odometry using a planar road model to solve scale ambiguity. Proceedings of the 5th European Conference on Mobile Robots (ECMR 2011), pp. 43-48.

Knyaz, V.A., Kniaz, V.V., Remondino, F., Zheltov, S.Y., Gruen, A., 2020. 3D reconstruction of a complex grid structure combining UAS images and deep learning. Remote Sensing, 12(19), pp. 3128.

Kotsakis, C., 2007. Least-squares collocation with covariance-matching constraints. Journal of Geodesy, 81(10), pp. 661-677.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2017. ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), pp. 84-90.

La Salandra, M., Miniello, G., Nicotri, S., Italiano, A., Donvito, G., Maggi, G., Dellino, P., Capolongo, D., 2021. Generating UAV high-resolution topographic data within a FOSS photogrammetric workflow using high-performance computing clusters. International Journal of Applied Earth Observation and Geoinformation, 105, 102600.

LaViola Jr, J.J., Kruijff, E., McMahan, R.P., Bowman, D., Poupyrev, I.P., 2017. 3D user interfaces: Theory and practice. Addison-Wesley Professional, Boston, US, ISBN: 9780201758672.

Li, G., Yu, L., Fei, S., 2021. A deep-learning real-time visual slam system based on multi-task feature extraction network and self-supervised feature points. Measurement, 168, 108403.

Longuet-Higgins, H.C., 1981. A computer algorithm for reconstructing a scene from two projections. Nature, 293(5828), pp. 133-135.

Lourakis, M.L., Argyros, A.A., 2005. Is levenberg-marquardt the most efficient optimization algorithm for implementing bundle adjustment?. 10th IEEE International Conference on Computer Vision (ICCV'05) Volume 1. IEEE, pp. 1526-1531.

Lowe, D.G., 2004. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60, pp. 91-110.

Lu, J., Liou, M.L., 1997. A simple and efficient search algorithm for block-matching motion estimation. IEEE Transactions on Circuits and Systems for Video Technology, 7(2), pp. 429-433.

Lysholm, J., Wiklander, J., 1987. Injuries in runners. The American Journal of Sports Medicine, 15(2), pp. 168-171.

Ma, W., Wen, Z., Wu, Y., Jiao, L., Gong, M., Zheng, Y., Liu, L., 2016. Remote sensing image registration with modified SIFT and enhanced feature matching. IEEE Geoscience and Remote Sensing Letters, 14(1), pp. 3-7.

Mahendran, A., Vedaldi, A., 2015. Understanding deep image representations by inverting them. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5188-5196.

Mak, J., Hess-Flores, M., Recker, S., Owens, J.D., Joy, K.I., 2014. GPU-accelerated and efficient multi-view triangulation for scene reconstruction, IEEE Winter Conference on Applications of Computer Vision. IEEE, pp. 61-68.

Maoteng, Z., Shunping, Z., Xiaodong, X., Junfeng, Z., 2017. A new GPU bundle adjustment method for large-scale data. Photogrammetric Engineering & Remote Sensing, 83(9), pp. 633-641.

Min, D., Choi, S., Lu, J., Ham, B., Sohn, K., Do, M.N., 2014. Fast global image smoothing based on weighted least squares. IEEE Transactions on Image Processing, 23(12), pp. 5638-5653.

Mizginov, V., Kniaz, V., 2019. Evaluating the accuracy of 3D object reconstruction from thermal images. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 42, pp. 129-134.

Moustafa, M., Ebeid, H.M., Helmy, A., Nazmy, T.M., Tolba, M.F., 2016. Rapid real-time generation of super-resolution hyperspectral images through compressive sensing and GPU. International Journal of Remote Sensing, 37(18), pp. 4201-4224.

Muja, M., Lowe, D.G., 2014. Scalable nearest neighbor algorithms for high dimensional data. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(11), pp. 2227-2240.

Munguia, R., Grau, A., 2007. Monocular slam for visual odometry. 2007 IEEE International Symposium on Intelligent Signal Processing. IEEE, pp. 1-6.

Mur-Artal, R., Montiel, J.M.M., Tardos, J.D., 2015. ORB-SLAM: A versatile and accurate monocular SLAM system. IEEE Transaction on Robot, 31(5), pp. 1147-1163.

Ni, K., Dellaert, F., 2006. Stereo tracking and three-point/one-point algorithms-a robust approach in visual odometry. 2006 International Conference on Image Processing. IEEE, pp. 2777-2780.

Nickolls, J., Buck, I., Garland, M., Skadron, K., 2008. Scalable parallel programming with CUDA: Is CUDA the parallel programming model that application developers have been waiting for? Queue, 6(2), pp. 40-53.

Nistér, D., 2004. An efficient solution to the five-point relative pose problem. IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(6), pp. 756-770.

Nistér, D., Naroditsky, O., Bergen, J., 2004. Visual odometry. Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2004). IEEE, pp. I-I.

NVIDIA, 2017. Jetson TX2 developer kit user guide. URL: https://developer.nvidia.com/downloads/jetson-tx2-developer-kit-user-guide (accessed 2022 Apr. 11).

NVIDIA, 2018. Jetson Xavier NX developer kit. URL: https://developer.download.nvidia.com/embedded/L4T/r32_Release_v4.2/Jetson_Xavier_NX_Developer_Kit_User_Guide.pdf (accessed 2022 Apr. 11).

NVIDIA, 2021. Nvidia deep learning TensorRT documentation. URL: https://docs.nvidia.com/deeplearning/tensorrt/ (accessed 2022 Apr. 11).

NVIDIA, Vingelmann, P., Fitzek, F.H.P., 2020. CUDA, release: 10.2.89. URL: https://docs.nvidia.com/cuda/archive/10.2/ (accessed 2022 Mar. 19).

Ono, Y., Trulls, E., Fua, P., Yi, K.M., 2018. LF-net: Learning local features from images. Advances in Neural Information Processing Systems, 31.

Owens, J.D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A.E., Purcell, T.J., 2007. A survey of general-purpose computation on graphics hardware, Computer graphics forum. Wiley Online Library, pp. 80-113.

Patias, P., 2002. Medical imaging challenges photogrammetry. ISPRS Journal of Photogrammetry and Remote Sensing, 56(5-6), pp. 295-310.

Peng, K., Chen, X., Zhou, D., Liu, Y., 2009. 3D reconstruction based on SIFT and HARRIS feature points, 2009 IEEE International Conference on Robotics and Biomimetics (ROBIO). IEEE, pp. 960-964.

Pepe, M., Prezioso, G., 2016. Two approaches for dense DSM generation from aerial digital oblique camera system. GISTAM, pp. 63-70.

Perri, S., Corsonello, P., Cocorullo, G., 2013. Adaptive census transform: A novel hardware-oriented stereovision algorithm. Computer Vision and Image Understanding, 117(1), pp. 29-41.

Pesce, V., Colagrossi, A., Silvestrini, S., 2022. Modern spacecraft guidance, navigation, and control: From system modeling to AI and innovative applications. Elsevier, Cambridge, US, ISBN: 9780323909167.

Pfister, A., West, A.M., Bronner, S., Noah, J.A., 2014. Comparative abilities of Microsoft Kinect and Vicon 3D motion capture for gait analysis. Journal of Medical Engineering & Technology, 38(5), pp. 274-280.

Pix4D, S., 2017. Pix4Dmapper 4.1 User Manual. Pix4D SA: Lausanne, Switzerland. URL: https://support.pix4d.com/hc/en-us/articles/204272989-Offline-Getting-Started-and-Manual-pdf (accessed 2023 Feb. 15).

Pulighe, G., Fava, F., 2013. DEM extraction from archive aerial photos: Accuracy assessment in areas of complex topography. European Journal of Remote Sensing, 46(1), pp. 363-378.

Qin, Z., Yu, H., Wang, C., Guo, Y., Peng, Y., Xu, K., 2022. Geometric transformer for fast and robust point cloud registration. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11143-11152.

Ranjan, R., Patel, V.M., Chellappa, R., 2017. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41(1), pp. 121-135.

Rashid, M., Khan, M.A., Sharif, M., Raza, M., Sarfraz, M.M., Afza, F., 2019. Object detection and classification: A joint selection and fusion strategy of deep convolutional neural network and SIFT point features. Multimedia Tools and Applications, 78, pp. 15751-15777.

Recker, S., Hess-Flores, M., Joy, K.I., 2013. Statistical angular error-based triangulation for efficient and accurate multi-view scene reconstruction. 2013 IEEE Workshop on Applications of Computer Vision (WACV). IEEE, pp. 68-75.

Remondino, F., Spera, M.G., Nocerino, E., Menna, F., Nex, F., 2014. State of the art in high density image matching. The Photogrammetric Record, 29(146), pp. 144-166.

Rone, W., Ben-Tzvi, P., 2013. Mapping, localization and motion planning in mobile multi-robotic systems. Robotica, 31(1), pp. 1-23.

Rosten, E., Drummond, T., 2006. Machine learning for high-speed corner detection, Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Part I, 9, pp. 430-443.

Roth, S., Black, M.J., 2007. On the spatial statistics of optical flow. International Journal of Computer Vision, 74, pp. 33-50.

Roy, R., Tu, Y.P., Sheu, L.J., Chieng, W.H., Tang, L.C., Ismail, H., 2023. Path planning and motion control of indoor mobile robot under exploration-based SLAM (e-SLAM). Sensors, 23(7), pp. 3606.

Rublee, E., Rabaud, V., Konolige, K., Bradski, G., 2011. ORB: An efficient alternative to SIFT or surf, 2011 IEEE International Conference on Computer Vision. IEEE, pp. 2564-2571.

Saouli, A., 2019. Effective multi-view stereo 3-dimensional reconstruction for virtual reality (Doctoral thesis). Universite Mohamed Khider-BISKRA.

Sarafianos, N., Boteanu, B., Ionescu, B., Kakadiaris, I.A., 2016. 3D human pose estimation: A review of the literature and analysis of covariates. Computer Vision and Image Understanding, 152, pp. 1-20.

Sarlin, P.-E., DeTone, D., Malisiewicz, T., Rabinovich, A., 2020. Superglue: Learning feature matching with graph neural networks. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4938-4947.

Sawada, R., Hirata, K., 2023. Mapping and localization for autonomous ship using LiDAR SLAM on the sea. Journal of Marine Science and Technology, 28, pp.410-421.

Scaramuzza, D., Fraundorfer, F., 2011. Visual odometry [tutorial]. IEEE Robotics & Automation Magazine, 18(4), pp. 80-92.

Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nešić, N., Wang, X., Westling, P., 2014. High-resolution stereo datasets with subpixel-accurate ground truth. Pattern Recognition: 36th German Conference, GCPR 2014, 36, pp. 31-42.

Scharstein, D., Szeliski, R., 2002. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision, 47, pp. 7-42.

Scharstein, D., Szeliski, R., 2003. High-accuracy stereo depth maps using structured light. 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE, pp. I-I.

Schiele, S., Blaar, H., Müller-Hanneman, M., Thürkow, D., Möller, M., 2012. Parallelization strategies to speed-up computations for terrain analysis on multi-core processors. ARCS 2012. IEEE, pp. 1-6.

Schönberger, J.L., Zheng, E., Frahm, J.-M., Pollefeys, M., 2016. Pixelwise view selection for unstructured multi-view stereo. Computer Vision - ECCV 2016, Part III 14, pp. 501-518.

Seki, A., Pollefeys, M., 2017. SGM-nets: Semi-global matching with neural networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 231-240.

senseFly, 2019. University campus. URL: https://ageagle.com/data-set/university-campus/ (accessed 2022 Oct. 04).

Seo, J., Han, S., Lee, S., Kim, H., 2015. Computer vision techniques for construction safety and health monitoring. Advanced Engineering Informatics, 29(2), pp. 239-251.

Shang, Z., Shen, Z., 2018. Real-time 3D reconstruction on construction site using visual SLAM and UAV, Construction Research Congress 2018, pp. 305-315.

Sheta, B., Elhabiby, M., El-Sheimy, N., 2012. Assessments of different speeded up robust features (surf) algorithm resolution for pose estimation of UAV. International Journal of Computer Science and Engineering Survey, 3(5), pp. 15.

Shigeto, Y., Sakai, M., 2011. Parallel computing of discrete element method on multi-core processors. Particuology, 9(4), pp. 398-405.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Sinha, S.N., Scharstein, D., Szeliski, R., 2014. Efficient high-resolution stereo matching using local plane sweeps. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1582-1589.

Song, X., Zhao, X., Hu, H., Fang, L., 2019. EdgeStereo: A context integrated residual pyramid network for stereo matching, Computer Vision - ACCV 2018: 14th Asian Conference on Computer Vision, Part V, 14, pp. 20-35.

Souici, A., Courdesses, M., Ouldali, A., Chatila, R., 2013. Full-observability analysis and implementation of the general SLAM model. International Journal of Systems Science, 44(3), pp. 568-581.

Spangenberg, R., Langner, T., Adfeldt, S., Rojas, R., 2014. Large scale semi-global matching on the CPU, 2014 IEEE Intelligent Vehicles Symposium Proceedings. IEEE, pp. 195-201.

Spangenberg, R., Langner, T., Rojas, R., 2013. Weighted semi-global matching and center-symmetric census transform for robust driver assistance, Computer Analysis of Images and Patterns: 15th International Conference, CAIP 2013, Part II, 15, pp. 34-41.

Stentoumis, C., Karkalou, E., Karras, G., 2015. A review and evaluation of penalty functions for semi-global matching, 2015 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP). IEEE, pp. 167-172.

Studholme, C., Hill, D.L., Hawkes, D.J., 1999. An overlap invariant entropy measure of 3D medical image alignment. Pattern recognition, 32(1), pp. 71-86.

Sun, J., Shen, Z., Wang, Y., Bao, H., Zhou, X., 2021. LoFTR: Detector-free local feature matching with transformers. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8922-8931.

Szeliski, R., 2022. Computer vision: Algorithms and applications. Springer Nature.

Technology, S.f.U., Turner, J., Yule, D.J., Zanre, J., 1991. Real Time Photogrammetry — A Technique for Today or Tomorrow?, SUBTECH'91: Back to the Future, pp.319-331.

Teke, M., Vural, M.F., Temizel, A., Yardımcı, Y., 2011. High-resolution multispectral satellite image matching using scale invariant feature transform and speeded up robust features. Journal of Applied Remote Sensing, 5(1), 053553-053559.

Thomasian, A., 2022. Chapter 8 - database parallelism, big data and analytics, deep learning, in: Thomasian, A. (Ed.), Storage Systems. Morgan Kaufmann, pp. 385-491.

Tonioni, A., Poggi, M., Mattoccia, S., Di Stefano, L., 2017. Unsupervised adaptation for deep stereo. Proceedings of the IEEE International Conference on Computer Vision, pp. 1605-1613.

Treleaven, P., Wells, J., 2007. 3D body scanning and healthcare applications. Computer, 40(7), pp. 28-34.

Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W., 2000. Bundle adjustment - a modern synthesis, Vision Algorithms: Theory and Practice: International Workshop on Vision Algorithms Corfu, pp. 298-372.

Trucco, E., Verri, A., 1998. Introductory techniques for 3-D computer vision. Prentice Hall, Englewood Cliffs, US, ISBN: 9780132611084.

Tsai, R., 1987. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses. IEEE Journal on Robotics and Automation, 3(4), pp. 323-344.

Tsourounis, D., Kastaniotis, D., Theoharatos, C., Kazantzidis, A., Economou, G., 2022. SIFT-CNN: When convolutional neural networks meet dense SIFT descriptors for image and sequence classification. Journal of Imaging, 8(10), pp. 256.

Van Meerbergen, G., Vergauwen, M., Pollefeys, M., Van Gool, L., 2002. A hierarchical symmetric stereo algorithm using dynamic programming. International Journal of Computer Vision, 47, pp. 275-285.

Vasudevan, A., Kumar, D.A., Bhuvaneswari, N., 2016. Precision farming using unmanned aerial and ground vehicles. 2016 IEEE technological innovations in ICT for agriculture and rural development (TIAR). IEEE, pp. 146-150.

Viola, P., Jones, M., 2001. Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. IEEE, pp. I-I.

Vladimir, A.G., 2016. Point clouds registration and generation from stereo images. Inf. Content Process, 3, pp. 193-199.

Vlutters, M., Van Asseldonk, E.H., Van der Kooij, H., 2016. Center of mass velocity-based predictions in balance recovery following pelvis perturbations during human walking. Journal of Experimental Biology, 219(10), pp. 1514-1523.

Wand, M., Berner, A., Bokeloh, M., Fleck, A., Hoffmann, M., Jenke, P., Maier, B., Staneker, D., Schilling, A., 2007. Interactive editing of large point clouds, PBG@ Eurographics, pp. 37-45.

Wang, C., Zhao, C., Yang, J., 2011. Monocular odometry in country roads based on phase-derived optical flow and 4-DoF ego-motion model. Industrial Robot: An International Journal, 38(5), pp. 509-520.

Wang, Q., 2019. Towards real-time 3D reconstruction using consumer UAVs. arXiv preprint arXiv:1902.09733.

Wang, Z., Chen, J., Hu, J., 2022. Multi-view cosine similarity learning with application to face verification. Mathematics, 10(11), pp. 1800.

Wei, X., Zhang, Y., Li, Z., Fu, Y., Xue, X., 2020. DeepSfM: Structure from motion via deep bundle adjustment, Computer Vision - ECCV 2020: 16th European Conference, Part I 16, pp. 230-247.

Wiechert, A., Gruber, M., Karner, K., 2012. Ultramap: The all in one photogrammetric solution. ISPRS International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 39, pp. 183-186.

Wolf, P.R., Dewitt, B.A., Wilkinson, B.E., 2014. Elements of photogrammetry with applications in GIS. McGraw-Hill Education, New York, US, ISBN: 9780071761123.

Wu, B., 2021. Photogrammetry for 3D mapping in urban areas. In: Shi, W., Goodchild, M.F., Batty, M., Kwan, M.-P., Zhang, A. (Eds.), Urban informatics. Springer Singapore, Singapore, pp. 401-413.

Xu, C., Liu, C., Li, H., Ye, Z., Sui, H., Yang, W., 2022. Multiview image matching of optical satellite and UAV based on a joint description neural network. Remote Sensing, 14(4), pp. 838.

Xu, Z., Yu, J., Yu, C., Shen, H., Wang, Y., Yang, H., 2020. CNN-based feature-point extraction for real-time visual slam on embedded FPGA. 2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM). IEEE, pp. 33-37.

Yang, G., Manela, J., Happold, M., Ramanan, D., 2019. Hierarchical deep stereo matching on high-resolution images. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5515-5524.

Yang, J., Zhang, J., 2015. Parallel performance of typical algorithms in remote sensing-based mapping on a multi-core computer. Photogrammetric Engineering & Remote Sensing, 81(5), pp. 373-385.

Yang, Q., Wang, L., Yang, R., Stewénius, H., Nistér, D., 2008. Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(3), pp. 492-504.

Ye, F., Su, Y., Xiao, H., Zhao, X., Min, W., 2018. Remote sensing image registration using convolutional neural network features. IEEE Geoscience and Remote Sensing Letters, 15(2), pp. 232-236.

Yoo, J.-C., Han, T.H., 2009. Fast normalized cross-correlation. Circuits, Systems and Signal Processing, 28, pp. 819-843.

Yu, Y., Pradalier, C., Zong, G., 2011. Appearance-based monocular visual odometry for ground vehicles, 2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM). IEEE, pp. 862-867.

Zabih, R., Woodfill, J., 1994. Non-parametric local transforms for computing visual correspondence, Computer Vision – ECCV'94: Third European Conference on Computer Vision Stockholm, Volume II, 3, pp. 151-158.

Zach, C., Karner, K., Bischof, H., 2004. Hierarchical disparity estimation with programmable 3D hardware. Proceeding of the International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, pp. 275-282.

Zatsiorsky, V.M., 2002. Kinetics of human motion. Human kinetics, Champaign, US.

Zbontar, J., LeCun, Y., 2015. Computing the stereo matching cost with a convolutional neural network. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1592-1599.

Zhang, B., Jiao, Y., Ma, Z., Li, Y., Zhu, J., 2014. An efficient image matching method using speed up robust features. 2014 IEEE International Conference on Mechatronics and Automation. IEEE, pp. 553-558.

Zhang, F., Prisacariu, V., Yang, R., Torr, P.H., 2019. GA-net: Guided aggregation net for end-to-end stereo matching. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 185-194.

Zhang, J., Chen, G., Jia, Z., 2017. An image stitching algorithm based on histogram matching and SIFT algorithm. International Journal of Pattern Recognition and Artificial Intelligence, 31(04), 1754006.

Zhang, J., Singh, S., Kantor, G., 2014. Robust monocular visual odometry for a ground vehicle in undulating terrain. Field and Service Robotics: Results of the 8th International Conference, pp. 311-326.

Zhang, L., 2003. Shape and motion under varying illumination: Unifying structure from motion, photometric stereo, and multiview stereo. Proceedings Ninth IEEE International Conference on Computer Vision. IEEE, pp. 618-625.

Zhang, Z., 2000. A flexible new technique for camera calibration. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(11), pp. 1330-1334.

Zhao, Q., Zhang, B., Lyu, S., Zhang, H., Sun, D., Li, G., Feng, W., 2018. A CNN-SIFT hybrid pedestrian navigation method based on first-person vision. Remote Sensing, 10(8), pp. 1229.