A Unique Approach for Block Partitioning and Merging for Large-Scale Structure-from-Motion using Apache Spark

[1]L.A.H.NAURUNNA, [2]S.C.PREMARATNE, [1]T.N.D.S.GINIGE

[1]School of computing
Universal College Lanka
Colombo
SRI LANKA

[2]Faculty of Information Technology
University of Moratuwa
Colombo
SRI LANKA

Abstract: - In response to the growing demand for large-scale reconstructions, this paper addresses the scalability challenges encountered by traditional Structure from Motion (SfM) methods. Our research aims to leverage Apache Spark's distributed computing capabilities to enhance the efficiency of SfM methodologies. The motivation behind this work lies in the increasing need for robust solutions capable of handling extensive reconstruction tasks. To tackle this challenge, we propose a method that harnesses the advantages of Apache Spark, including scalability, speed, fault-tolerance, flexibility, and ease of use. The abstracted problem centers around the limitations inherent in Apache Spark's traditional operations like maps, reduces, and joins. Our methodology focuses on a block partitioning and merging strategy, strategically distributing the workload using Spark. Our paper also presents experimental results showing the feasibility of our approach through the 3D reconstructions of multiple datasets. The experiments were executed on a standalone Spark instance, demonstrating the potential of Apache Spark in effectively distributing SfM workloads. In summary, this paper elucidates the necessity for addressing scalability challenges in large-scale reconstructions, outlines the research goals, and details a method leveraging Apache Spark to overcome limitations and enhance the efficiency of SfM.

Key-Words: - Structure-from-Motion, Apache Spark, 3D Reconstruction, Block partitioning and Merging, Workload distribution, Large Scale 3D Reconstruction, Photogrammetry, Resilient Distributed Dataset.

Received: March 25, 2023. Revised: December 7, 2023. Accepted: December 24, 2023. Published: December 31, 2023.

1 Introduction

With the increasing number of applications needing large-scale reconstructions, Structure-from-Motion faces the problem of keeping up with these demanding workloads, [1]. Traditional Structure-from-Motion approaches follow a sequential approach of execution managing workloads by executing them sequentially, this is however inefficient at scale, [2]. In more advanced Structure-from-Motion solutions, distribution algorithms are used to distribute workloads across multiple computing resources. This often involves dividing the main workload into smaller subsets, referred to as blocks, that can be processed concurrently on different computers or nodes, [3]. The use of Apache Spark as a means of running workloads in a distributed manner could yield several advantages associated with Apache Spark, [4], such as:

- Scalability: Apache Spark allows for the distributed processing of data across a cluster of computers, enabling SfM solutions to scale up to process larger datasets more efficiently.
- Speed: The distributed nature of Apache Spark allows SfM solutions to process data in parallel, resulting in faster processing times.
- Fault-tolerance: Apache Spark is fault-tolerant, which means that it can recover from failures and continue processing data without interruption.
- Flexibility: Apache Spark supports multiple programming languages, allowing SfM solutions to be written in the language of choice if needed.
- Ease of use: Apache Spark provides a simple and intuitive programming model, making it easy to write and debug SfM solutions.

Although Spark may seem like a optimal solution for distributing SfM workloads it comes with a few limitations, [4], such as only being able to perform operations in terms of maps, reduces, joins and a few other operations. This may seem to limit the reference to neighboring data or external data at first glance but there are trivial work arounds to these issues that can be utilized to use Apache Spark in the distribution of SfM or 3D reconstruction workloads.

In a traditional workflow, the SfM dataset is broken up into several smaller, more manageable datasets and is sent to different machines for processing, producing their own point cloud and camera pose data, this data is then stitched together using a 3D space similarity transformation, [5], Iterative Closest Point or other methods, [6]. In our method a similar approach will be taken but with the use of spark. There are several differences in our method that defer from other block partitioning and merging algorithms using custom distribution techniques, [5].

2 Background and Related Work

2.1 Apache Spark

Apache Spark, developed at the University of California, Berkeley in 2009, has emerged as a prominent distributed computing framework for processing large datasets, [4]. Facilitated by a resilient distributed dataset (RDD) abstraction, Spark ensures efficient parallel processing and fault tolerance in distributed environments.

Several studies highlight the efficacy of Apache Spark in accelerating data processing and executing computationally intensive algorithms. Notable examples include the Wale Optimization Algorithm (WOA) study, [2], and the multiscale feature extraction and semantic classification of LiDAR point data, [7].

In [2] Apache Spark was employed to implement the WOA, utilizing maps and reduces for parallel processing. The study compared Spark-based WOA with traditional and Hadoop-based approaches, demonstrating superior runtime and scalability in Spark.

[7] illustrates the utilization of Apache Spark and Cassandra for multiscale feature extraction and semantic classification on point cloud data. The study showcased promising results, affirming Spark's capability to handle complex scenarios and algorithms efficiently.

2.2 Workload Distribution in Structure-from-Motion (SfM)

Structure-from-Motion (SfM) stands as a widely-used technique for constructing 3D models from 2D images. The computational intensity of SfM, particularly with larger datasets, has prompted research into optimizing workload distribution.

In [5] a block partitioning and merging strategy was proposed for efficient workload distribution in SfM algorithms. This approach involves splitting images into overlapping subsets, processing them in parallel, and merging the results based on shared images and tie points. The adoption of block partitioning and merging significantly improved the time efficiency of large-scale incremental structure-from-motion problems.

3 Proposed Method

In this proposed method, we will be taking several images and finally processing them into a sparse point cloud, this approach currently will not use bundle adjustment to reduce complexity but can always be added in later studies. We implement our approach in python using the pyspark API to utilize Apache Spark in python. Taking advantage of Resilient Distributed Datasets (RDDs) to handle the distribution processing we process the image data to a point cloud in five stages:

- Image parallelization and preparation: Parallelizing the image data into a ParallelCollectionRDD and doing any basic image processing as needed. Processing may include resizing, converting to grayscale or any other processes preparing the images for the rest of the SfM pipeline.
- Feature detection and camera initialization: In this step Features are detected using the SIFT feature detector and cameras are initialized with a camera index, the image associated with the camera, the key points, descriptors and the intrinsics of the camera.
- Camera partitioning: In this step the initialized cameras are partitioned into overlapping blocks of a pre-defined size for processing and overlap to produce subsets.
- 3D point cloud generation: a point cloud in the generated using traditional Structure-from-motion techniques based on two initial seed cameras and by registering the additional cameras in the block via a Perspective-n-Point based camera registration.

- Collection and Merging: the blocks are then collected after processing and merged together using the overlapping camera's adjusting the block's points and subsequent cameras accordingly.

Our approach stops here as the distribution strategy of SfM workloads is the main focus but additionally the point cloud can be further processed in Spark after the collection and merging step to yield better results. Local Bundle Adjustment (BA) can also be done in the 3D point cloud generation stage to further improve accuracy and furthermore global BA after the collection and merging stage.

3.1 3D Reconstruction

The creation of Three-dimensional (3D) reconstructions is the procedure of creating Three-dimensional recreations of objects seen in the scene via the data collected through equipment which may be a camera or lidar sensor/scanner. In the context of this project, 3D Reconstruction via image data will be the main focus. The 3D output for 3D Reconstruction can vary depending on the context in which it is needed, the base output that can be expected at a minimum from a 3d reconstruction is a sparse point cloud of the object, with further processing a dense point cloud, geometric/mesh models can be produced. Due to the cost effective and efficient nature of 3d Reconstruction as a means to get a 3d representation of an object it has been applied in fields such as civil engineering, [8], capturing snapshots of historic buildings, survey engineering, asset creation for media, [9], and other applications.

In this section, our primary focus is to get an idea on 3D Reconstruction Techniques from 2D Images, providing an overview of general approaches, algorithms, and techniques, giving us a good starting point to choose the best approach for this study. [10] serves as an excellent starting point, providing valuable insights to guide our exploration.

In [10] it reviews 3d Reconstruction as a whole, the algorithms as well as the various techniques used in 3d Reconstruction. One of the things discussed is the requirements/problems in 3d Reconstruction from 2d images, which are as follows, the calibration parameters for the cameras used, data associations between two or more images' features, determining spatial information of the features and producing a dense reconstruction from the sparse data obtained. The problem of obtaining a 3d reconstruction from a 2d image is described as restoring the depth information lost from the transformation from the 3d world to the 2d plane when taking a picture.

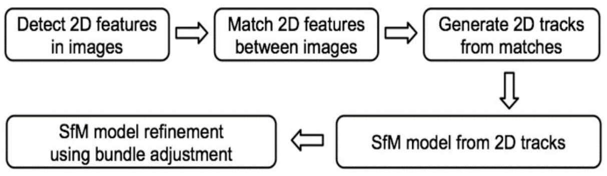
Detect 2D features in images
Match 2D features between images
Generate 2D tracks from matches
SfM model from 2D tracks
SfM model refinement using bundle adjustment

Figure 1: A simple overview of a SFM pipeline

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = M \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (1)$$

The equation above describes this transformation as the camera projection from a point 3d point in the world to 2d (and thus the loss of information) with reference to the pinhole camera model. Z_c is is an arbitrary scaling perimeter, u, v describe the image coordinates, K is the camera intrinsics, $[R\&T]$ describes the extrinsics and M is the camera matrix. The paper also describes Structure from motion as one of the techniques that are used in 3d Reconstruction which will be touched on later. Figure 1 shows a very basic outline of a SfM pipeline

The paper also mentions Multi-View Stereo, [11], which focused on refining the 3D reconstruction obtained from SfM by estimating detailed depth information for each pixel in the images, leading to a dense 3D representation.

In the review of 3d Reconstruction in [8] it goes more into depth about MVS referencing, [12], and their algorithm CMVS and PMVS and how they can be used in conjunction with the output of SFM algorithms to produce a dense reconstruction. Furthermore, it elaborates on how mesh reconstruction can be done with the PSR algorithm. The paper also Illustrates the typical 3d reconstruction pipeline. MVS, although a great addition to a 3d reconstruction pipeline, is not considered with this paper as it deviates from the main focus of exploring the feasibility of using Apache Spark as a tool for distributing 3d reconstruction workloads.

Although many 3d reconstruction methods exist, Structure from Motion was chosen as it can be fully parallelized at certain steps in the pipeline and partially parallelized using the block partitioning and merging approach to provide a final 3d point reconstruction. Figure 2 outlines the pipeline for classical SFM where it shows two flows in which monocular images and stereo images can be used to finally generate point clouds and how the point

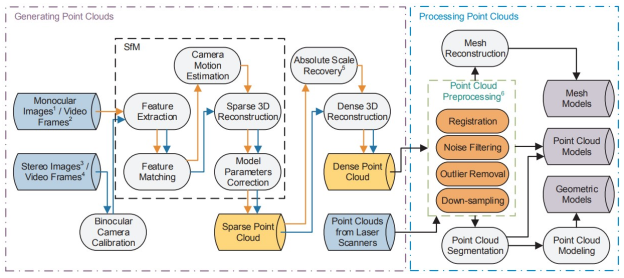

Figure 2: An overview of a 3d reconstruction pipeline that uses SfM

clouds can be processed to generate mesh models, point cloud models and geometric models.

Within Structure from Motion (SfM) itself, there are many approaches and pipelines which can either improve or hinder the ability to parallelize the pipeline. While numerous reviews cover diverse aspects of 3D reconstruction, our approach for SfM is a very generic sequential SfM pipeline, which starts with keypoint detection in each image, using the SIFT descriptor, [13], for cross-image keypoint comparison, and employing random sampling and consensus (RANSAC), [14], to discard outlier matches the essential matrix is then calculated using the 8 point algorithm. Our adaptation diverges from most classical SfM pipelines like in [15] by excluding bundle adjustment from the SfM pipeline. This choice to diverge from this method was to achieve more parallelizability as a global bundle adjustment is not possible with block partitioning and merging. However a local bundle adjustment with each block can be done but was not considered for this study.

3.2 Image parallelization and preparation

The main goal of this stage is to parallelize and prepare the image data for further processing. We first parallelize the data with an appropriate number of partitions for the images, generally following the equation $P_n = 4C_n$, where P_n is the number of partition and C_n is the number of total cores available in the cluster. The data preparation step can be done using a map with any processing needed to be done inside the map function depending on the data-set.

3.3 Feature detection and camera initialization

The main goal of this stage to initialize the cameras with the appropriate data for the rest of the SfM pipeline before partitioning into blocks. Feature detection is an important step in any SfM pipeline, the feature detector used is SIFT which can be computationally expensive and thus makes sense to continue running parallelly. The SIFT features are computed

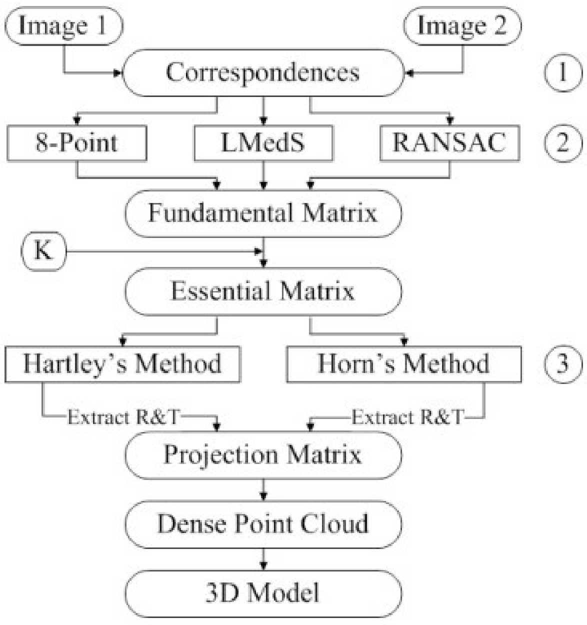

Figure 3: A generic SfM pipeline

and will become a part of the camera initialization in the next step. The camera model used in our approach is the pinhole camera model and as such the camera calibration matrix, K, are defined as follows

$$\begin{bmatrix} f_x & S & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Where $f_x = F/P_x$ and $f_y = F/P_y$, describing the focal length in pixels, where F is the focal length and P_x, P_y describing the size of a pixel in world units. C_x, C_y describes the optical center in pixels and S is the pixel skew. The cameras are initialized with the image associated with the camera and the key points, descriptors from the feature detector, the calibration matrix and the index of the camera. The index of the camera is taken as the frame index.

3.4 Structure from Motion (SfM)

Structure from Motion (Visual SfM) is a computational technique designed to unravel the three-dimensional (3D) structure of a scene from a collection of two-dimensional (2D) images.

Figure 3 describes a classical SfM pipeline which gives the traditional way of doing sequential SfM with no block partitioning and merging. The pipeline starts off with the correspondences between the two images extracted, then mentions the ways in which the fundamental matrix can be calculated, in our method uses the same flow, we use the 8 point algorithm but with

RANSAC to get rid of outliers in the data before calculation. With the fundamental matrix, the essential matrix can be easily calculated by adding the calibration parameters using $E = (K_2)^T F K_1$ where we know $K\_1 = K\_2$ because it is from the same camera. Then we use Hartley's method to extract R & T to calculate the projection matrix and with linear triangulation we achieve a sparse point cloud. With the incorporation of block partitioning and merging the steps are altered slightly with the addition of shuffles. In more detail our pipeline follows the following steps with block partitioning and merging:

3.4.1 Feature Detection and Matching

In the initial phase, distinctive features within each image are identified using the SIFT feature detector, [13]. These features serve as recognizable points in the scene, such as corners or key landmarks. Subsequently, corresponding features across different images are matched, establishing associations that allow for data association throughout the sequence. RANSAC is also employed to detect and remove outliers and also lowe's ratio test, [13], is also applied to filter out non-discriminative matches. The outcome of this step is a list of points that match between each sequential image. The Feature detection part in our block partitioning and merging strategy can be all done in parallel, for the matching part an initial shuffle is done breaking all the images along with their detected features into chunks. Each chunk has a predefined size and overlap and can be expressed by the following equation, where N is the number of chunks, C is the number of cameras, S is the chunk size and O is the Overlap.

$$N = (C - O)/(S - O) \tag{3}$$

Any remaining items are added as a remainder chunk that might not be the predefined size.

3.4.2 Essential Matrix Calculation

The feature matching data can be used to then calculate the essential matrix using the 8 point algorithm, [16]. The essential matrix describes the relationship between two calibrated cameras viewing the same scene from different viewpoints. It encapsulates the intrinsic parameters of the cameras as well as the relative pose between the two cameras. The following equation describes the essential matrix.

$$x'^T F x = 0 \tag{4}$$

$$E = (K_2')^T F K_1 \tag{5}$$

where $K_1 = K_2$ if it's the same camera

The essential matrix as seen in the equation is an encoding of the fundamental matrix with the intrinsic calibration data. In the equation, $E = (K_2')^T F K_1$, F is the fundamental matrix and K_1 is the calibration of the first camera and K_2 is the calibration of the second camera. $x'^T F x = 0$ represents the epipolar constraint in computer vision, where F is the fundamental matrix, and x and x' are corresponding points in two images taken by different cameras. This process takes place within each chunk between the first and the second camera.

3.4.3 Decomposition for Translation and Rotation

The essential matrix can then be solved to give T and R using the method proposed in [17], A. A brief outline of the method. Using the above-mentioned equation, $E = (K_2')^T F K_1$, were

$$K_1 = \begin{bmatrix} fK_u & 0 & u_0 \\ 0 & fK_v & V_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Then we perform an SVD of E as follows:

$$K_1 = U diag(1, 1, 0) V^T \tag{7}$$

R and T have two solutions:

$$R_1 = UWV^T \; and \; R_2 = UW^T V^T \tag{8}$$

$$T_1 = u_3 \; and \; T_2 = -u_3 \tag{9}$$

u_3 is the third column of U

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10}$$

The correct R and T are the parameters when used can lead to 3d point that are In-front of the camera when triangulated. In the Figure 4, the correct solution is the top left solution.

With the essential matrix for all the frames the baseline and rotation can be obtained by decomposing and solving as described above. With the baselines and rotations, the path can be estimated by multiplying each transformation $\begin{bmatrix} R_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix}$ one after the other to obtain a relative path for the initial 2 cameras are still kept within each chunk.

3.4.4 Relative Path Estimation

The obtained baseline and rotation matrices are used to estimate a relative path, outlining the trajectory of the camera through the scene.Note that this is for just the first two cameras, we consider the first camera to be at the world origin with a transformation matrix of

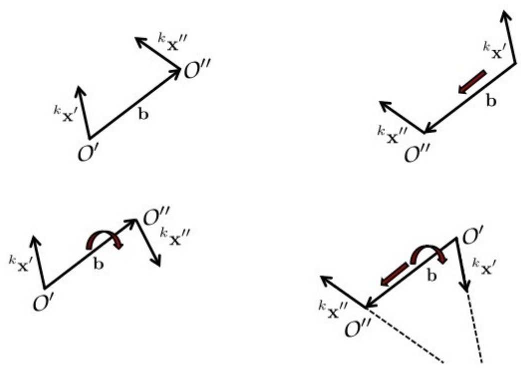

Figure 4: The four solutions from Hartley & Zisserman's method

$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\end{bmatrix}$ while the second camera has the transformation matrix of $\begin{bmatrix}R_{3\times3} & T_{3\times1}\\ 0_{1\times3} & 1\end{bmatrix}$ once finally resolved. This sets the scale for each and every chunk as the translation vector is a unit vector for the second camera and subsequent camera's added would be relatively scaled to the initial baseline.

3.4.5 Triangulation for 3D Point Extraction

Key points identified in the two cameras can then be triangulated to determine their 3D positions in the scene. This step is done by using a linear triangulation on all the corresponding points. This too is to different scales relative to the initial baseline produced by the initial two cameras.

3.4.6 Perspective-n-Point (PnP) for Camera Registration

The Perspective-n-Point algorithm is employed to register new cameras in the scene based on the 2D-3D correspondences with the images and the point cloud. This step adds camera's to the graph whilst also adding new points to the point cloud, the baseline of the camera's added by perspective-n-point will have a base line relative to the initial baseline calculated.

3.4.7 Merging

In the proposed approach, the seed camera positions and rotations of reference blocks are taken from overlapping cameras in target blocks. The reference block is transformed using the 3D space similarity transformation model. For effective transformation, an overlap of more than two cameras is required. If the overlap is less than two cameras, an alternative Perspective-n-Point (PnP) technique is employed to determine the seed camera's position and rotation before transformation.

When the overlap is greater than two cameras, it can be used to scale subsequent point clouds to the first point cloud's scale using an affine transformation. The transformation involves solving for an affine transformation matrix to bring the points and cameras of the subsequent point cloud to the same scale and rotation as the reference point cloud. This process is applied to all subsequent point clouds, resulting in a fully connected graph of cameras and a complete point cloud. All calculations are performed in Homogeneous coordinates and will be described in detail in Section 3.7.

3.4.8 Final Output

The final output is a sparse point cloud representing the 3D structure and detailed camera extrinsics and intrinsics for each frame. It is important to note that while this framework provides a foundation for other SfM pipelines with block partitioning and merging, additional enhancements, such as loop closing algorithms, can be integrated for more advanced applications.

3.5 Camera partitioning

In this stage the main goal is to partition the cameras up into overlapping blocks of a pre-defined sized and overlap to produce subsets of the original dataset for further execution. The overlap needs to exist to facilitate the process of merging as the overlapping images act as a tie point, [5]. Camera partitioning can be done on spark through the following steps:

- Generating a comprehensive associative array where the id of the blocks is the key. And the values are the corresponding images in each block, generated according to a pre-defined block size and overlap
- Each camera is given an index using Spark's zipWithIndex() function
- A Spark map() function is done where each image is duplicated and stored with the index of the blocks it appears in according to the index assigned in the last step
- A Spark flatMap() function is executed to obtain a list of cameras and their corresponding blocks. The output from the previous step is flattened to create this list. If a camera appears in multiple blocks, it is listed more than once in this list.

- A Spark groupByKey() function is finally run to produce each block. Each block will contain the block id and the list of cameras in the block.

3.6 Point cloud generation

In this stage the main goal is to produce sparse point clouds from each associated block using SfM techniques. While there are numerous approaches to SfM, a detailed discussion of the specifics of the methods is beyond the scope of this paper. However, we will briefly outline the SfM approach used in our approach although many other approaches can be taken.

At this point we will find that each block has a list of cameras and their respective camera data. This camera data can be used like in a traditional SfM pipeline to produce a sparse point cloud. In our approach, a conventional approach is taken to, where the detected features are matched between two corresponding cameras and processed to get the essential matrix of the initial pose of the 2 initial cameras. The essential matrix can then be decomposed to get four separate solution which can be disambiguated to one solution. This solution can be used to initially triangulate a sparse point cloud. Other cameras are then registered using a Perspective-n-Point (PnP) based camera registration. And thus, each block can be processed out into a point cloud.

Once the block partitioning is completed, each block will have a list of cameras and their corresponding camera data. This camera data can be used in a conventional SfM pipeline to produce a sparse point cloud. In our approach, two cameras are taken as seed cameras and an initial pose is estimated. To produce this initial pose estimation, the detected features are matched between the two corresponding cameras, and the essential matrix of the initial pose is computed. The essential matrix is then decomposed into four separate solutions, which are disambiguated to obtain one solution. This solution is used to initially triangulate a sparse point cloud. Subsequently, the other cameras are registered using Perspective-n-Point (PnP) based camera registration. This process is run parallelly for each block, resulting in a point cloud for each block. These point clouds will be the output for this stage

3.7 Collection and Merging

Once the block partitioning is complete the sparse point clouds produced by each subset can be merged using several methods such as with a least squares solution where two point clouds with overlapping 3d points are minimized such to minimize the squared distance produced by trying to fit the reference point cloud to the target point cloud. Another approach, as proposed in [5] involves transforming a reference block to match the targeted block using the 3D space similarity transformation model. The transformation parameters are calculated based on the shared points between the two sub-blocks.

In our approach, the reference blocks seed camera's positions and rotations are taken from the target blocks overlapping cameras and the reference block is transformed accordingly using the 3D space similarity transformation model. This method requires the overlap to exceed two cameras in order to be effective. Should the overlap be less than two, an alternative approach utilizing the Perspective-n-Point (PnP) technique must be utilized to determine the position and rotation of the seed camera, prior to transformation.

In the case of the overlap greater than two this overlap can be used to scale each of the subsequent point clouds to the first point clouds scale using an affine transformation. The transformation in detail means, if we have the base point cloud A in it's own chunk with it's camera's A_a, B_a, C_a, D_a and we have point cloud B in the neighbouring chunk with it's camera's C_b, D_b, E_b, F_b, where C_a and D_a is the same camera as C_b and D_b respectively as the chunks overlap cameras. We can figure out the transformation needed to bring point cloud B's points and camera's to be at the same scale as rotation as A by figuring out an affine transformation T that can be solved for by solving $A_T = T * B_T$ where A_T is the baseline from C_a to D_a and B_T is the baseline from C_b to D_a. This T is then used to transform the entire point cloud B into the same space as A and then can be appended onto A by replacing C_a with C_b and D_a with D_b and their newly transformed points. Note that all the calculations are done in Homogeneous coordinates. Doing this process to all the subsequent point clouds should yield a fully connected graph of camera's and full point cloud.

4 Experiments are Results

The experiments conducted in this section will be done to provide evidence that our approach works and is a feasible solution. However, it is important to note that we do not compare the time efficiency of our approach to other methods, as we were not able to access a spark cluster for performance testing at the time of writing. Any performance test done on a standalone Spark instance would not be a fair comparison and thus will not be carried out. The experiments will include 3D reconstructing point clouds for the observatory and statue data-sets using

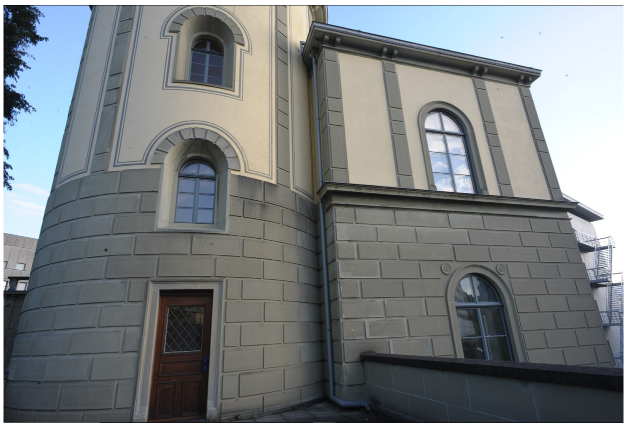

Figure 5: Observatory data-set

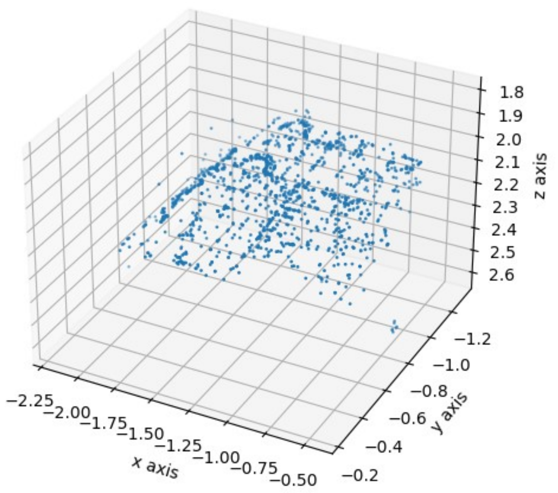

Figure 6: Observatory point cloud

our proposed distribution approach. The experiments will be run on a standalone Spark instance.

4.1 Observatory data-set

The observatory data-set is a set of undistorted images with known calibration parameters of an observatory. The data-set consists of twenty-seven non-aerial images of the observatory taken from a DSLR camera. For this experiment, each image is scaled down to forty percent of its original size and processed using our approach. The results can be seen in Figure 5, Figure 6 and Figure 7

4.2 Statue data-set

The statue dataset, similar to the observatory dataset, is a dataset of undistorted images with known camera parameters. It contains ten images of a statue. These

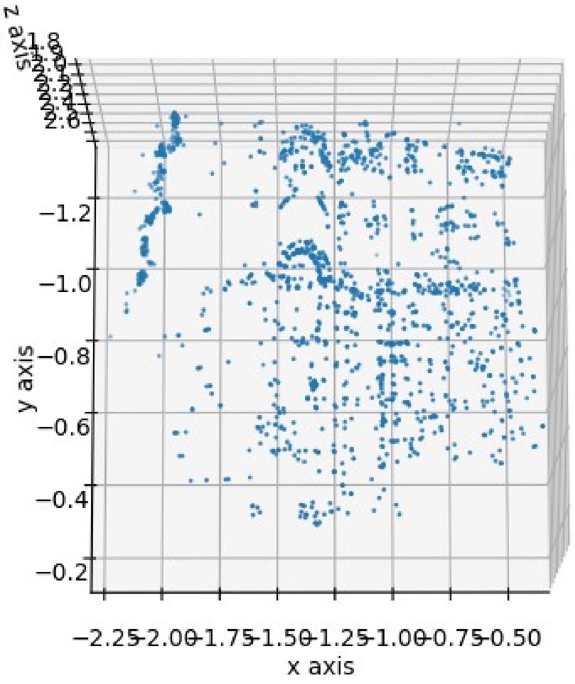

Figure 7: Observatory point cloud

images are scaled to forty percent its original for easier and faster processing. The results can be seen in Figure 8, Figure 9 and Figure 10.

5 Discussion

The results obtained from the experiments conducted on the observatory and statue datasets demonstrate the feasibility of our proposed approach for large-scale 3D reconstruction using Apache Spark. The successful generation of 3D point clouds from these datasets provides evidence that our methodology can effectively distribute Structure from Motion (SfM) workloads across a Spark platform.

The observatory dataset, consisting of twenty-seven non-aerial images, was processed using our approach, resulting in a detailed 3D point cloud of the observatory. Similarly, the statue dataset, comprising ten images, was also successfully processed to generate a 3D point cloud of the statue. These results indicate that our approach can handle varying dataset sizes and complexities, making it a versatile solution for large-scale 3D reconstruction tasks.

However, it is important to note that the performance of our approach was not compared with other methods due to the unavailability of a Spark cluster at the time of writing. While the experiments conducted provide evidence of the feasibility of our

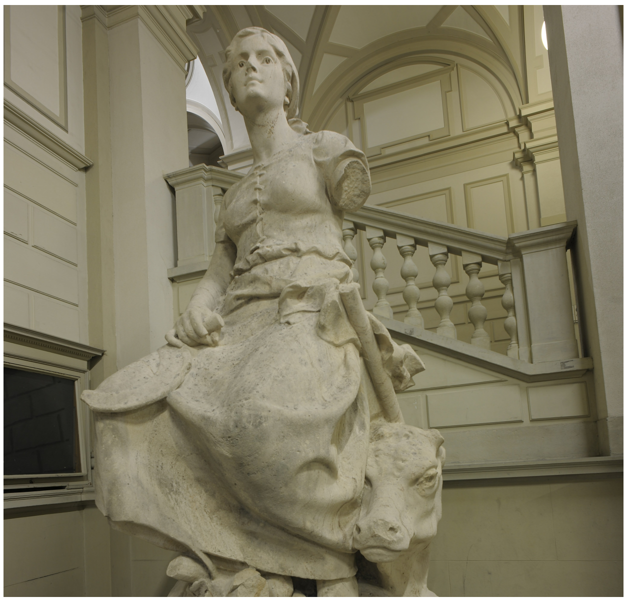

Figure 8: Statue data-set

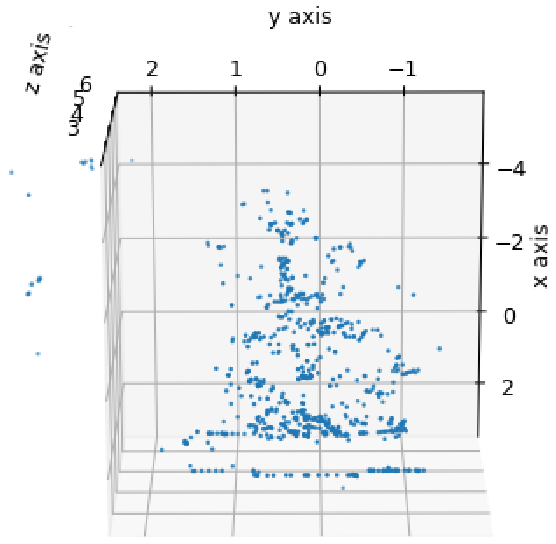

Figure 9: Statue point cloud

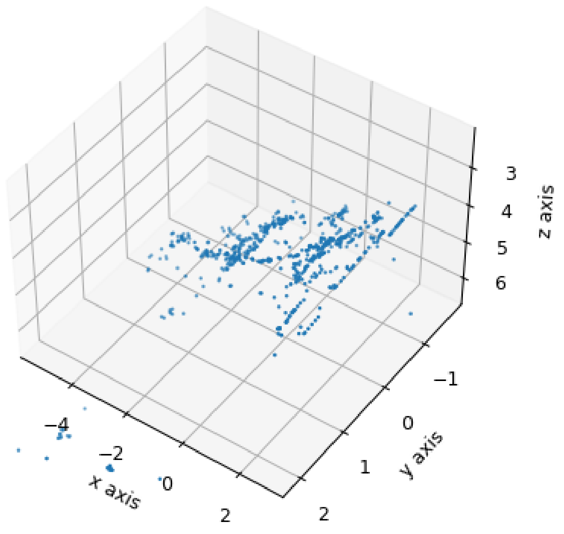

Figure 10: Statue point cloud

approach, a comprehensive performance evaluation would require a comparison with other methods on a Spark cluster. This would allow for a fair assessment of the efficiency and scalability of our approach in a distributed computing environment. Despite this limitation, the results obtained from our experiments provide a promising foundation for future research. The successful application of Apache Spark for distributing SfM workloads opens up new possibilities for enhancing the efficiency and scalability of large-scale 3D reconstruction tasks. Future work could focus on integrating additional enhancements, such as loop closing algorithms, to further improve the accuracy and detail of the 3D reconstructions.

In conclusion, our proposed approach for distributed 3D point cloud reconstruction using Apache Spark demonstrates potential for large-scale reconstructions. The approach leverages the advantages of Apache Spark, including scalability, speed, fault-tolerance, flexibility, and ease of use, to overcome the limitations inherent in traditional SfM methods. While further research is needed to fully assess the performance of our approach, the results obtained from our experiments provide a promising starting point for future developments in this field.

6 Conclusion

Based on the results presented in this report, our proposed approach for distributed 3D point cloud reconstruction using Apache Spark demonstrates feasibility and promise for large-scale reconstructions. The approach enables efficient distribution of data processing and utilization of parallel computing resources, which is crucial for reducing the computational time. Although the approach's performance wasn't able to compared with other methods due to the unavailability of a spark cluster at the time of writing, the experiments provide evidence that Apache Spark is a feasible solution for distributing SfM workloads.

The project has successfully achieved its objectives of designing and implementing a photogrammetry-based system for creating accurate and detailed 3D models from 2D images in a distributed manner. The testing of the system has shown that it can distribute SfM workloads in a robust and reliable way, with functional testing demonstrating this. Additionally, the non-functional testing has shown that the system is capable of running medium to small data sets in a reasonable time-frame making the method suitable for commercial use with more improvement to the underlying SfM used. In summary, this project has demonstrated the effectiveness of Apache spark and its capability in the distribution of Structure from Motion workloads.

7 Further work

While this research has achieved its objectives and made good progress in the field of distributed 3d Reconstruction, there are still limitations and areas for future work to be considered. One limitation of this research is that the project does not utilize bundle adjustment globally or locally as well as lacking a robust merging solution. Additionally, while the block partitioning used in this project was effective, it may not be the most efficient or accurate method for 3D Reconstruction. Other methods that could be explored in future work include improving the underlying SfM algorithms or incorporating something like COLMAP on spark for better results.

This research gives a good starting point for further research into using Apache Spark in the field of 3d Reconstruction. Overall, this project has made good progress in the distribution of Structure from motion, but there is still much to be explored and improved upon. With further research and development, the findings of this project could have important applications in various industries and fields.

Acknowledgment:
We would also like to acknowledge Universal College Lanka and the University of Central Lancashire for providing the opportunity to pursue our academic goals and engage in research.

References:

[1] Snavely, N., Seitz, S.M., & Szeliski, R. (2008). Modeling the world from internet photo collections. *International journal of computer vision*, 80, 189-210.

[2] AlJame, M., Ahmad, I. and Alfailakawi, M., 2020. Apache spark implementation of whale optimization algorithm. Cluster Computing, 23, pp.2021-2034

[3] Jiang, S., Jiang, C. and Jiang, W., 2020. Efficient structure from motion for large-scale UAV images: A review and a comparison of SfM tools. ISPRS Journal of Photogrammetry and Remote Sensing, 167, pp.230-251

[4] Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S. and Stoica, I., 2010. Spark: Cluster computing with working sets. In: Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing. HotCloud'10.

[5] Lu, L., Zhang, Y. and Liu, K., 2019. Block partitioning and merging for processing large-scale structure from motion problems in distributed manner. IEEE Access, 7, pp.114400-114413

[6] Pomerleau, F., Colas, F., Siegwart, R. and Magnenat, S., 2013. Comparing ICP variants on real-world data sets: Open-source library and experimental protocol. Autonomous Robots, 34, pp.133-148

[7] Singh, S. and Sreevalsan-Nair, J., 2020, December. A distributed system for multiscale feature extraction and semantic classification of large-scale LiDAR point clouds. In 2020 IEEE India Geoscience and Remote Sensing Symposium (InGARSS) (pp. 74-77). IEEE

[8] Ma, Z., & Liu, S. (2018). A review of 3D reconstruction techniques in civil engineering and their applications. *Advanced Engineering Informatics*, *37*, 163-174.

[9] Quixel, (2023). 3D world-building made easy. Quixel. 2023. Retrieved February 6, 2023, from https://quixel.com/

[10] Aharchi, M., & Ait Kbir, M. (2020). A review on 3D reconstruction techniques from 2D images. In *Innovations in Smart Cities Applications Edition 3: The Proceedings of the 4th International Conference on Smart City Applications 4* (pp. 510-522). Springer International Publishing.

[11] Furukawa, Y., & Hernández, C. (2015). Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, *9*(1-2), 1-148.

[12] Furukawa, Y., Curless, B., Seitz, S. M., & Szeliski, R. (2010, June). Towards internet-scale multi-view stereo. In *2010 IEEE computer society conference on computer vision and pattern recognition* (pp. 1434-1441). IEEE.

[13] Lowe, D. G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 20(2):91– 110, November 2004

[14] Bolles, R. C., & Fischler, M. A. (1981, August). A RANSAC-based approach to model fitting and its application to finding cylinders in range data. In *IJCAI* (Vol. 1981, pp. 637-643).

[15] Snavely, Seitz and Szeliski (2006): Snavely, N., Seitz, S. M., & Szeliski, R. (2006). Photo tourism: exploring photo collections in 3D. In *ACM siggraph 2006 papers* (pp. 835-846).

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

[17] Hartley, R., & Zisserman, A. (2003). *Multiple view geometry in computer vision*. Cambridge university press.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)
The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself
No funding was received for conducting this study.

Conflicts of Interest
The authors have no conflicts of interest to declare that are relevant to the content of this article.