3D Reconstruction application in Quarrying industry

Ágata Filipa Lopes de Barros

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisor: Prof. Jorge Manuel Mateus Martins

Examination Committee

Chairperson: Prof. Duarte Pedro Mata de Oliveira Valério

Supervisor: Prof. Jorge Manuel Mateus Martins

Member of the Committee: Prof. João Carlos Prata dos Reis

July 2021

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor, Professor Jorge Martins for his guidance and support during the development of this project.

I would also like to acknowledge the company Fravízel-Equipamentos Metalomecânicos, SA in the E-Techstone 4.0 project for being accessible and helpful during this process. I extend my gratitude to my great friend and colleague Filipe Monteiro for being present throughout this assignment which was crucial effort and cooperation in order to be completed.

Finally, I would like to thank my family and friends for being the pillar of encouragement and support in each aspect of this journey.

Abstract

For the past 20 decades, the desire to produce 3D models of the world from 2D images has been researched in order to recover 3D point clouds based on Structure from motion and Multiview stereo methods. Meanwhile, the exploitation of minerals through quarrying, has been a lucrative market and a major contribution to the infrastructures of the cities since before the 18th century. The aim of this study is to apply 3D reconstruction approaches to limestone quarries. Building 3D model of quarry limestones from 2D images to be implemented in the quarry industry in favor of improving and modernizing the cleaving process through computer vision. The 2D images were acquired by a stereo camera pair, in a similar uncontrolled environment of an open-mine pit. In order to create fitting 2D images for structure from motion applications, it was necessary to explore different preprocessing approaches. Preprocessing methods were analyzed, and a gamma correction with a histogram equalization was deemed to be the most successful approach. Once the images were adjusted in terms of brightness and contrast, different SFM and MVS pipelines were tested to conclude which is the best available software application and its correspondent algorithm sequence. Although, the open-source software, Meshroom, was determined to be the best 3D reconstruction system, Agisoft Metashape can be more appropriate for industry applications due to its community support and convenient renewability. Nevertheless, Meshroom allows to study different matching algorithms, with known and unknown intrinsic and extrinsic parameters to rule the most effective approach.

Keywords: 3D reconstruction, Computer vision, Quarry industry, Structure from motion (SFM), Multi-view Stereo (MVS)

Resumo

Nas últimas 20 décadas, diversos estudos científicos, na área de visão computacional, têm incidido na produção de modelos 3D a partir de imagens 2D. Em contrapartida, a indústria pedreira tem desenvolvido um mercado essencial para a construção de infraestruturas correntes, desde o século 18. Esta dissertação tem por objetivo reconstruir um modelo 3D de pedras calcárias, a partir de imagens 2D, de modo a aperfeiçoar e modernizar o processo de corte por meio de reconstruções 3D. As imagens 2D foram adquiridas por um par de camaras stereo, numa situação semelhante ao de uma mina aberta, sem controlo do meio ambiente. De modo a criar imagens 2D adequadas para aplicações de Structure from motion, foi necessário explorar diferentes técnicas de pré-processamento de imagens. Após métodos de pré-processamento serem analisados, o procedimento de correção de gamma seguido pela equalização do histograma foi considerada a técnica mais bem sucedida. A luminosidade e o contraste das imagens foram ajustados e os diferentes software de SFM e MVS foram testados para concluir o melhor software disponível e a sua respetiva sequência de algoritmos. Concluiu-se que, apesar do software Meshroom ter sido determinado o melhor sistema de reconstrução 3D, o Agisoft Metashape pode ser considerado mais apropriado para aplicações de indústria por oferecer mais apoio da comunidade e pelas convenientes atualizações. No entanto, Meshroom permite efetuar analises de diferentes algoritmos de correspondência de pontos de interesse, com ausência e presença de parâmetros internos e externos, para definir o método mais eficaz.

Palavras-chave: Visão computacional, Reconstrução 3D, Indústria Pedreira, Structure from motion (SFM), Multi-view stereo (MVS)

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AC-RANSAC	A Contrario RANSAC
AMD	Advanced Micro Devices
ANN	Approximate Nearest Neighbor
API	Application Programming Interface
ARRSAC	Preemptive RANSAC
BA	Bundle Adjustment
BF	Brute-Force
BRIEF	Binary Robust Independent Elementary Features
BRISK	Binary Robust Invariant Scalable Keypoints
CDF	Cumulative Distribution Function
CG	Conjugate Gradients
CIELAB	Commission International d'Eclairage (L-perceptual lightness, A/B-red, green, blue, yellow)
CLAHE	Contrast limited adaptive histogram equalization
CMOS	Complementary Metal Oxide Semiconductor
CMVS	Clustering Multiview Stereo
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DLT	Direct Linear Transform
DoG	Difference of Gaussian
EPnP	Efficient Perspective-n-Point
FAST	Features from Accelerated Segment Test
FREAK	Fast Retina Keypoint
GB	Gigabyte
GPU	Graphics Processing Unit
GroupSAC	Efficient Consensus in the presence of Groupings
GTX	Giga Texel Shader Extreme
GUI	Graphical User Interface
LM	Levenberg Marquardt
LoG	Laplacian of Gaussian
LO-RANSAC	Locally Optimized RANSAC
LPSIFT	Layer Parallel SIFT
LSH	Locality Sensitive Hashing
LUT	Lookup Tables
MLESAC	Maximum Likelihood Estimation Sample And Consensus
MP	Megapixels
MVE	Multi-View Environment
MVS	Multi-View Stereo
NFA	Number of False Alarms
NNDR	Nearest Neighbor Distance Ratio
NURBS	Non-uniform Rational Basis Spline

ORB	Oriented fast and Rotated BRIEF
PCA-SIFT	Principal Components Analysis - Scale Invariant Feature Transform
PDE	Partial differential Equation
PMVS	Patch-based Multi-View Stereo
PnP	Perspective-n-Point
PROSAC	Progressive Sample Consensus
RAM	Random-Access Memory
RANSAC	Random Sample Consensus
RGB	Red Green Blue color model
SFM	Structure From Motion
SGM	Semi Global Matching
SIFER	Scale Invariant Feature detector with Error Resilience
SIFT	Scale Invariant Feature Transform
SIMD	Single Input Multiple Data
SURF	Speeded Up Robust Feature
SVD	Singular Value Decomposition
TNN	Threshold Nearest Neighbor
UAV	Unmanned Aerial Vehicle
UI	User Interface
UPnP	Unified PnP

1 Introduction

This chapter presents the overview of Computer vision and its crucial applications on the quarrying industry, as the first section. The following section contextualizes the real quarry application that this assignment will approach. The project goals and structure are described on the third section of this study to introduce the objectives and to structurally divide it into main topics that will be discussed in the rest of the document. Lastly, in the fourth section, the contributions display the various software applied.

1.1 Overview

Computer Vision is a scientific discipline in constant improvement, described as the acquisition, processing and analyzing digital images or videos to filter crucial information through mathematical models [1]. University studies have been focused in simulating a human visual system, since late 1960 [2], and the following decades proved to be decisive in the development and applications of this field by deriving a three-dimensional structure from images and therefore, creating the foundation for several computer vision algorithms, such as, edge detection, optical flow and motion estimation. From there on, researchers were able to establish new and more complex mathematical concepts, including the scale-space theory, in order to forge sparse 3D reconstructions of scenes. This led to a better understanding of camera calibration and optimization methods already established, like bundle adjustment. During the 1990s, techniques such as multi-view stereo solved the correspondence problem to advance from sparse to dense 3D reconstruction from multiple images [1].

From an engineering perspective, Computer vision is another study to create and implement automated technologies to improve efficiency and reliability of well-established processes. Computer vision aims to build autonomous systems to substitute human tasks which operate from human vision. In consequence, various fields took advantage and profited from the range of applications it can offer. Such as optical sorting [3], reaching to diagnoses of patients from image data information, such as, detecting tumors [4], measurements of organs or even aiding in medical treatments [5], shown in Figure 1, to autonomous driving cars [6] and military projects, like enemy detection [7] and missile guidance [8].

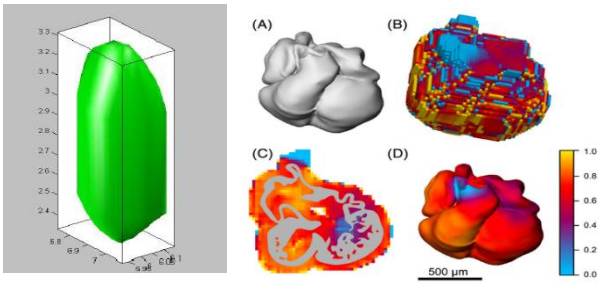


Figure 1: Visualization of a 3D reconstruction application [4][5]

.

Quarrying industry is the business of extraction of natural stone, gravel and sand. The importance of aggregates extraction is not recognized enough in present times, even though quarries are a fundamental factor for our current way of life. Stone is the main requirement for infrastructural construction, for instance, schools, hospitals, factories, even transport networks or sewerage systems. The overall advantages and disadvantages are organized in Table 1.

Although, this business destroys habitats and causes noise and visual pollution, it also creates jobs and improves economies. Concluding that quarries are a vital part of everyday life and to a nation's economy implies that this field should not be overlooked but focused on in order to be included in the automation of traditional industrial practice, through technology.

Table 1: Advantages and disadvantages of quarrying industry

Advantages	**Disadvantages**
Provide employment opportunities in rural areas with low diversity of jobs.	Noise pollution from the extraction process.
Increased income and, consequently, the local economy.	Visual pollution due to the quarry pit.
Industry progress can lead to infrastructure and road improvement.	Destruction of wildlife habitats.
Upgrading the access in remote rural areas for the transport of quarrying products.	Dust pollution from the rock blasting.

1.2 Problem Statement

The process of extraction aggregates starts by cutting and removing big blocks of stone from open-pit mines, afterwards, the stone is transported to another machine to be cut in order to end up with a marketable stone block. The second approach can be described as a cutting machine operated by a worker, whose purpose is to initialize the engine for the cutting structure to travel over the row of stone blocks, for approximately 30 minutes, but also to identify the exact location where the cut should be and placing the cutting apparatus on this location to be able to perform the groove. The cutting location is analyzed by the worker, through experience knowledge, to remove flaws and to obtain a geometrically pleasing stone block for commercial applications.

The idea would be to create an autonomous system that would capture a video of the limestone blocks in line on the machine, shown in Figure 2, to create a 3D reconstruction model of each limestone block and to determine the cut's position more accurately. Resulting in a more reliable and efficient process of cutting block stones. This project and all the input data were established and provided by the company Fravízel.

Figure 2: Limestone quarry cutting machine

The physical set up consists of two cameras Genie Nano C4020 Color with 12.4 MP, as a CMOS sensor, attached to the moving cutting structure and connected to its servo motor, as it can be seen in Figure 3 [9]. The cameras also have a high resolution of 12 MP with industrial lenses [10] mounted inside of a housing due to the constant dust and dirt present in the cutting process, but also to maintain its temperature between the operating restriction ($-20℃$ to $65℃$) [9]. These cameras are also connected to a computer, Nuvo-5000E/P Series [11], where the maximum exposure time and brightness are controlled and the trigger is actuated through the program Sapera Cam Expert of Teledyne Dalsa. The two Genie Nano C4020 Color mounted lenses have an angle of view of $83º5' \times 66º7'$ so that the same features can be seen from multiple views to reconstruct the most accurate 3D model possible [9].

Once the cutting structure of the machine starts to travel above the Limestone Blocks the trigger is activated and the cameras record a video of the aligned stone blocks [12]. Afterwards, this video is divided into different frames and the image acquisition is concluded, as the first step of a 3D reconstruction process [12].

Figure 3: Camera set up in a Limestone quarry

1.3 The Scope

Despite of the number of applications for 3D reconstruction on various scientific fields and the development of distinct algorithms to solve the same computational vision problems, a reconstruction of 3D models of Limestone blocks in quarry environments followed by an automation of the cutting process, has not yet been applied. This thesis statement is to implement and compare methods for full 3D reconstruction on a Limestone block from images, captured in an uncontrolled environment, addressing the requirements, effectiveness and reliability, for each step of the 3D reconstruction. This assignment aims to answer a main question:

How viable is to reconstruct a 3D model of a limestone taken in an uncontrolled environment for quarrying industry applications?

In order to answer this question, it is imperative to answer parallel questions, such as: What is the best software to perform 3D reconstruction of limestones? What is the best method to preprocess the images for better 3D model quality but also to allow the pipelines to reconstruct the scene? What is the best combination of algorithms within the chosen software? What is the duration of a 3D reconstruction of a limestone line? Is it possible to improve the duration and the quality of the 3D model by applying known intrinsic and extrinsic parameters?

Therefore, this assignment focuses on the following main objectives:

1. Investigate open source and commercial pipelines of 3D reconstruction applications;
2. Apply preprocessing methods to enhance brightness and contrast of the images;
3. Study each 3D reconstruction algorithm of each software;
4. Analyze the results of each approach in the respective steps of a 3D reconstruction;
5. Compare the implementation of 3D reconstruction methods, with known and unknown intrinsic and extrinsic parameters;
6. Conclude the best full 3D reconstruction in order to be implemented in a quarrying industry.

1.4 The challenges

The installed hardware consists of two cameras, and so, the limestone blocks aren't completely visible through just 2 cameras. Both the bottom side and the back side of each stone block do not have imagery information in order to build a 3D model of those sides. Therefore, the 3D models of the block are expected to be hollow and only the front, left and right side are reconstructed.

The constant variation of illuminations and shadows, due to direct sun exposition during the working hours, contribute to one of the biggest challenges in detecting and matching the same feature in different views. Also, the object that is being reconstructed is a limestone stone block which means low variation of texture and color in each block that can contribute to another difficulty in detecting and matching features.

Another challenge is the high resolution of the images taken by both cameras that contribute to a lengthier reconstruction process. For commercial software, high resolution can be solved by down scaling the images in the case of taking too much time to process, however for most open source pipelines, high resolution tends to be extremely time consuming for each 3D reconstruction step and sometimes it is even recommended to take more photos with lower resolution than fewer with high resolution.

1.5 Thesis Structure

After this introduction, the existing methods for 3D reconstruction of close range scenes in uncontrolled environment are reviewed in Chapter 2. This Chapter focuses on the most recent and promising theories in 3D reconstruction from images, but also in the most established methods. Conveys a background review organized with each main stage of a 3D reconstruction along with the most recent applications of 3D reconstruction methods.

Chapter 3 formulates the problem at hand by first describing the physical set up and the importance of appropriate image acquisition. The camera representation and model are presented with the respective parameters. This Chapter also describes the different algorithms that will be operated and the corresponding hardware requirements.

The general workflow of a 3D reconstruction establishes Chapter 4 composition. Chapter 4 describes the main steps involved in the stereo reconstruction, as the following stages: Feature extraction and description, feature matching, camera location estimation, internal and external calibration optimization and robust estimation for a sparse cloud generation. The Multi-view stereo approach to generate a dense cloud. The Delaunay triangulation and Poisson Surface reconstruction are the methods applied to create the mesh of the scene. Lastly, the textured reconstruction is addressed. Each implementation stage displays the results for the individual methods.

The reconstruction methods applied in the previous chapter are assessed in Chapter 5. In this Chapter, the results are presented and discussed for each step of the software reconstruction with the intention to analyze and compare the respective methods.

Finally, Chapter 6 arrives to the main conclusions and suggests improvement ideas for future works.

1.6 Contribution

This assignment examines the different recent methods applied in open-source and commercial software. Approaches using software tools, such as, *VisualSFM*, *COLMAP*, *MeshLab*, *Meshroom* and *Agisoft Metashape*, were studied in order to reach the best performance to apply in a Quarrying industry.

2 Background

This chapter presents the current state of the art by displaying preprocessing methods and the approaches for the various stages of 3D reconstruction, from well-established reserches to more recent ones. More specifically, Structure form motion, dense, surface and textured reconstruction background approaches are explored to organize some algorithms that will be explained later. A couzple of 3D reconstruction pipelines are also described along some practical 3D reconstruction applications of closed range scenes.

2.1 Preprocessing Methods

One of the obstacles of 3D reconstructions is the quality of the input images. If the images are captured in an uncontrolled environment, the quality of the images might need to be improved, through contrast and brightness enhancement, before reconstructing a 3D model of the scene. There are a variety of methods to enhance contrast and brightness that can be implemented in *Python* such as the histogram equalization, CLAHE and gamma correction [13]. According to [14], a good contrast enhancement can also be achieved by combining gamma correction to avoid excess brightness and a histogram equalization method, such as , CLAHE, to provide a good contrast to the images

2.2 3D reconstruction

In this chapter, the current state of the art is presented, focusing on a passive method of image-based 3D reconstruction. An image-based 3D reconstruction is a robust, inexpensive and flexibly automated approach to acquire data in the form of 2D images to reach a virtual 3D model [15][16]. A reconstruction made using multiple cameras can achieve better geometric data, a surface texture of visually realistic models, increase robustness and decrease image noise compared to a two-view camera reconstruction approach [17]. Therefore, the 3D model of the quarry stone in this assignment has been reconstructed from multiple views to achieve visual realism. Since the acquisition of data was enforced through passive sensors, the 3D coordinates necessary to generate a surface are derived based on mathematical models.

The approach of reconstructing a 3D model of an object or scene using correspondences between sequences of images taken from multiple viewpoints is the Multi-view geometry method. Multi-view geometry establishes correspondences between points in different images from the 3D position in a scene of the image plane [1]. Examples of the 3D reconstruction applications in quarrying industry can be found in [18][19][20][21].

Multi-view geometry approaches can sustain some adversities during interest point detection and matching due to smooth surfaces or low texture objects [1]. Uniform neighborhood of features can induce wrong correspondences between interest points and occlusions in images from ambiguous correspondences.

However, a SFM approach is essential in order to estimate information for a multi-view stereo method to be applied, in case the camera parameters and the camera location and orientation are unknown.

The process of 3D model building has several steps: The camera alignment procedure consists of a feature detection and description algorithm, a feature matching algorithm, pose estimation methods, such as, the Perspective-n-Point problem for calibrated or uncalibrated cameras and a robust estimation method. When the camera poses are known and features are detected and matched, the Direct Linear Transform (DLT) is used for the triangulation process. Once a sparse point cloud is generated, it is possible to create a dense point cloud through MVS approaches. A 3D surface is built from surface reconstruction methods, such as, the Delaunay triangulation and the Poisson surface reconstruction algorithm.

2.2.1 Camera Alignment

Camera alignment consists in 5 stages: Feature detection, feature matching, pose estimation, robust estimation and triangulation. These methods are fundamental to computational vision and have been already solved through different principles.

2.2.1.1 Feature detection

Feature detection is usually the first step in image processing with the purpose of identifying relevant pieces of information in the image, from edges to corners, or even blobs, expressed as interest points or regions of interest points. Features can be described by the appearance of the pixels around the point location or by the orientation and the local appearance of the pixels surrounding the interest point. However, an interest point is commonly defined as the intersection point between two or more edges in favor of identifying a feature with a well localized position in the image but also stable enough to be reliable for computation repeatability if subjected to brightness variations. Once the interest point is detected, a local descriptor for this keypoint is built to best describe the feature in order to differentiate one from another, acting as a numerical “fingerprint” and typically presented as a descriptor vector. A local descriptor represents the local shape and appearance in the neighborhood around the point but also the orientation of the neighboring pixels. Ideally, this descriptor should be invariant under illumination, translation, scale and in-plane rotation changes. The most common algorithms to identify features are the following ones: Harris Corner detector, Speeded Up Robust Feature, Features from Accelerated Segment Test, Oriented FAST and Rotated BRIEF (ORB), BRISK, FREAK and Scale Invariant Feature Transform (SIFT). Although ORB is the fastest algorithm to detect and match features, being 10 times faster than SURF and 40 times faster than SIFT, the algorithm SIFT is the most common method for feature detection in 3D reconstruction applications for its high accuracy [22].

Scale invariant feature transform (SIFT) was proposed by David G. Lowe in 2004. This method detects and describes local features in images [23]. SIFT locates keypoints and attributes quantitative information in

order to match the same keypoint from an image with different perspectives [23]. This information is placed in descriptors which are developed through an algorithm grounded in 4 main steps: The estimation of a scale space extrema, a keypoint localization, the assignment of the orientation to a keypoint and finally, the generation of a local image descriptor for each keypoint [23]. This approach grants scale, rotation, illumination, viewpoint and affine transformation invariance proving to be one of the most ideal approaches with the best performance among the other feature detector algorithms [24]. However, SIFT demands a large computational complexity invalidating this approach for real-time applications. Compared to other similar techniques such as SURF, FAST, BRIEF and ORB that have almost the same matching accuracy and a faster performance, SIFT still provides the most efficient performance in most scenarios [24]. Through the years, extensions and variants of SIFT have been improving its computational complexity for a faster performance, such as extracting features through SIFT in a real time method [25], or improving SIFT overall performance, like a hybrid approach by applying Hessian detector and SIFT to extract features [26], a PCA-SIFT approach [27] or even SIFER [28].

PCA-SIFT or Principal components analysis SIFT computes its local image gradient and calculates a compact feature vector based on a pre-computed eigenspace as the gradient image of local patches. In relation to SIFT, this approach has a smaller feature vector, and so it not only improves its speed but also its matching accuracy [27].

Scale invariant feature detector with error resilience or SIFER is another variation to the Standard SIFT, without sustaining planar rotational invariance [28]. This method applies the Cosine Gaussian filter to detect scale invariant features with the intention of reducing the scale detection error and increasing the feature localization accuracy on features that are resistant to variation of noise, image compression, illumination, motion-blur, zoom, resolution and scale [28]. Depending on the scale-space filter specifications, SIFER can achieve 20% better detection and matching results compared to SIFT [28].

For real time applications, It was developed a layer parallel SIFT with integral image (LPSIFT with integral image and hardware design) [25]. LPSIFT with integral image solves the obstacle of long latency caused by the Gaussian blur iteration which is replaced by a reconstructed box kernel parallel layer that was simplified with the reuse of the sub-kernel sum when applied in integral images. The keypoint localization process is also simplified with a low brightness analysis taking the place of the low contrast test due to the Taylor expansion's high complexity. In [25] was acknowledged that the scale space extrema detection and the keypoint localization required more storage memory and were the most computationally complex steps overall, therefore, both Gaussian pyramid, the DoG pyramid and the keypoint localization were computed at the same time to reduce storage and latency in order to reduce the computational complexity of the problem. It was also adopted the integral image approach for multiple box kernels in each scale. The hardware design that only stores partial temporal results and the keypoint localization computation was implemented by a low universal operation unit to reduce the gate count. This approach achieves the real time demand by saving 56% gate count and 90,4% memory cost compared to the previous methods [25].

SURF is a rotation and scale invariant feature extractor algorithm [29]. This method is computationally effective and can prevent aliasing by implementing a different scale space representation and convolving box filters of various sizes with integral images [30]. Also, SURF applies an approximation of the Hessian-matrix through integral images and determines the location of the interest point as the maximum determinant value of the Hessian matrix [29]. This approach was inspired by SIFT algorithm [29]. On the other hand, a Hybrid feature extractor is the integration of the SURF detector into the SIFT algorithm in order to increase the True Positive matches [26]. The feature extraction methods of both algorithms are combined. Therefore, both methods are applied: The Lowe's method develops a scale space representation, estimates the Difference of Gaussians of the images and finds its maxima and minima to localize de key point. The SURF method generates a scale space representation by employing Gaussian and convolves the box filter (squared-shaped filters as an approximation of Gaussian smoothing) with integral images to find the Hessian matrix as a blob detector to identify the interest point location as its maximal determinant. Both methods assign the orientations to keypoints. The interest points are ordered according to their values and the numbers with the best features by solving it according to an adaptive threshold. This Hybrid approach is scale, rotation, view change, blurring and noise effect invariant. Besides, it has higher precision of extracting features and produces more True Positive matches than each feature extraction method involved [26]. However, this algorithm has a demanding computational cost, preventing it to be a good alternative for real time applications [26].

2.2.1.2 Feature Matching

Feature Matching consists in a searching technique that enforces a similarity measure to compare features from each image based on a matching strategy. Therefore, a process of feature matching is divided into three parts: The similarity measure, the matching strategy and the searching technique. The similarity between two features is usually evaluated by a distance measure. If the feature is represented by a histogram, such as the case of SIFT, then there are bin-by-bin and cross-bin measures [31]. The bin-by-bin technique doesn't regard the information on the neighboring bins, comparing only the corresponding histogram bins. The most common method is the Euclidean distance. Contrary to the bin-to-bin approach, the cross-bin method considers the nearby bins information.

There are three main matching strategies: the absolute threshold, the thresholded nearest neighbor (TNN) and the nearest neighbor distance ratio (NNDR). The absolute threshold method assumes a correct match when the absolute distance between the two features is less than a certain threshold. The thresholded nearest neighbor matches a feature from one set with the nearest neighbor feature from the other set and establish a correct match if the absolute distance between them is less than a threshold. The nearest neighbor distance ratio takes into consideration, not only the nearest feature of the second set but also the second nearest and creates a correct match if the ratio between these distances are less than a pre-set threshold.

There are two main approximate nearest neighbor (ANN) searching techniques: The hierarchical space partition-based [32] and the hash-based methods [33].

The hierarchical space partition-based methods for ANN searching is the k-d tree [32] as the most commonly used, the R-trees [34] and the B-trees [35]. The k-d tree was proposed by Bentley in 1975 as a binary search tree. Each key value from a node is associated with k-dimensions, and its space is divided into two parts a left subspace with k^{th} components features that are less than the key value and the right subspace corresponding to elements greater than the key value. The query feature is compared during the tree search, with the node key value until a leaf node is reached. The closest feature to this query feature is reached after comparing it with all the features in the leaf node. The closest feature is the true nearest neighbor if it is centered on a sphere where the radius is the distance between the query feature and the closest feature.

The hash-based methods have a better efficiency than the previous method but less accuracy for using features represented as binary code. An example of a hash-base method is locality sensitive hashing [33]. LSH applies a set of hash functions to a group with similar features under the same hash bucket. The query feature is mapped on the hash tables with similar features to the query feature and then each distance between the query feature and the rest of the similar features is computed to determine the feature with a distance less than a certain threshold.

Cascade hashing [36] also transforms all features in binary code through LSH and the method is divided in 3 layers: the hashing lookup, hashing remapping and hashing ranking. Hashing lookup matches point by constructing lookup tables with buckets using binary codes and all the points that are in the same bucket of the feature from the first image that wants to be matched in the second image is kept. Hashing remapping calculates the Euclidean distance between every candidate provided by the hashing lookup stage through a hashing function to remap them into a Hamming space (set of binary strings) and sort them by their Hamming distance. The hashing ranking is organized by computing the Hamming distance to construct buckets with all the points with the same Hamming distance to access the points with 0 Hamming distance, if the number of points is less than k then the next bucket is accessed until the number of points summed is k. These k top candidates allow to calculate their Euclidean distance and find the two nearest neighbors.

2.2.1.3 Pose estimation

A pose estimation problem's objective is to estimate the intrinsic and extrinsic parameters of the camera: the camera calibration parameters (the focal length, the principal point, the skew and other distortion coefficients) and the six degrees of freedom of the camera pose. A well-established approach to estimate camera position is the Perspective-n-Point method. An example of an iterative method is the Gao's approach of the P3P problem [37] and a non-iterative method is the EPnP method [38]. However, before solving a problem of camera position estimation it is crucial to first compute the fundamental or essential matrix depending on prior knowledge. If the intrinsic parameters of the camera are known, only the essential

matrix needs to be calculated through a 5 point algorithm [39], If any parameter of the camera is known, then the fundamental matrix needs to be computed through a 8 point algorithm [40]. There are more robust and recent methods with similar approach as the eight-point algorithms, such as, the robust fundamental matrix estimation [41] that improves the estimation of the fundamental matrix by upgrading and correctly selecting the 8 point correspondences to compute the Fundamental matrix.

During the 1950s, Bundle adjustment was conceived and through the years developed as image-based positioning method in photogrammetry [42]. Bundle adjustment is the problem of refining camera parameters and 3D points as the last step of feature-based 3D reconstruction algorithms [43]. The objective of this optimization problem is to minimize the reprojection error between the observed image location and the predicted image point through a nonlinear least-squares algorithm [43]. The most popular approach into solving nonlinear least-squares problem is the Levenberg-Marquardt method due to being an effective damping strategy that converges from a wide range of guesses and for its ease of implementation [44]. However, in 2019, Bundle adjustment methods were reanalyzed and the conventional method of solving the bundle adjustment problem based on Levenberg-Marquardt algorithm was compared to a distributed bundle adjustment [45]. In [45] it was concluded that the distributed bundle adjustment surpasses the conventional approach both in accuracy and efficiency since the conventional method is limited to single machine leading to memory limitation.

2.2.2 Triangulation

The triangulation recovers the 3D position from 2D projections to build the tie points represented in sparse point clouds. These positions can be obtained after the keypoints were detected and matched and the camera positions and parameters are known. The linear triangulation method (DLT) is an approach to solve triangulation [1].

2.2.3 Robust estimation

An important extra step to take is a robust estimation approach to improve the solution into a more stable and reliable process. A robust estimation algorithm removes outliers or some mismatches in order to obtain the most accurate matching and camera pose estimation. The most popular method is the RANSAC algorithm [46]. Due to the high computational complexity of this step, the method only relies on random sampling a subset for the estimation. Therefore, the RANSAC method has a couple of variation in order to efficiently sample data, such as: PROSAC, LO-RANSAC, Guided-MLESAC, GroupSAC, ARRSAC. PROSAC and Guided-MLESAC algorithms sample correspondences through an appearance based score [47][48]. GroupSAC method samples the data through image segmentation for a better efficiency and ARRSAC or Preemptive RANSAC is used in certain events to have a more accurate estimation by avoiding

more outlier contaminated hypotheses [49][50]. Some main RANSAC variations used in 3D reconstruction software are [51]: PROSAC and LO-RANSAC.

The progressive sample consensus (PROSAC) [47] method is a robust variation of RANSAC which applies a linear ordering through a similarity function to establish correspondences. Unlike RANSAC, PROSAC treats the correspondences differently by straining them from progressively larger sets of top-ranked correspondences. PROSAC [47] algorithm begins by generating hypothesis through random sampling. The hypothesis size is continuously increasing, and the set of uncontaminated samples are examined first. The algorithm converges to a solution when the probability of the existence is a better solution than the present best. The crucial steps of this process are the size of the hypothesis generation set and the stopping criterion of the sampling process.

LO-RANSAC or locally optimized RANSAC [52] is identical to RANSAC approach, the main difference is that the estimated solution from random sample is locally optimized. Similarly to RANSAC, model parameters are computed to fit a sample which was randomly selected from the input data. The size of the sample is crucial to develop an accurate model because a model can be less accurate if its parameters are estimated by a small data set. Therefore, it is created minimal samples that if points are added the probability of selecting an outlier free sample decreases exponentially. This random sample has the minimum number of data but enough to compute the model parameters. Afterwards, a cost function is used to evaluate the quality of the model parameters by calculating the number of inliers and comparing it with the model assumption. This process is repeated until the probability of finding a better model becomes low. Each time a new maximum of inliers occurs, the local optimization is applied, and the best model stored.

AC-RANSAC is another variation of RANSAC that finds a model that best fits the data without relying in a fixed threshold but automatically adapting a confidence threshold to noise [53]. This method is a parameter free approach that focuses in minimizing the Number of False Alarms (NFA), equation 2.1, instead of minimizing the median of errors and the only requirement is for the model to return at least $2N_{samples}$ inliers [53].

$$NFA(M,k) = N_{out}\left(n - N_{sample}\right)\binom{n}{k}\binom{k}{N_{sample}}\left(e_k(M)^d\alpha_0\right)^{k-N_{sample}} \qquad 2.1$$

Where k is the number of hypothesized inlier correspondences, n is the total number of correspondences, N_{sample} is the number of a RANSAC sample, N_{out}t is the number of models that can be estimated form a RANSAC sample of N_{sample} correspondences, $e_k(M)$ is the k-th lowest error to the model M among all n correspondences α_0 is the probability of a random correspondence having error 1 pixel and d is the error dimension, which is attributed 1 for point-to-line distance and 2 for point-to-point.

2.2.4 Dense reconstruction

This step recovers the details of the scene by using both image and camera parameters for each image to generate a dense point cloud. There are two main methods to build a dense reconstruction of the scene: The CMVS and the PMVS. The CMVS generates clustered images from clustering the non-redundant images [54] and PMVS generates a dense point cloud through matching, expansion and filtering the previous clustered images [55][56]. Another approach is through SGM [57] which builds a dense point cloud based on global energy minimization process [58]. Although SGM is faster than PMVS, PMVS is accurately more stable than SGM [59].

2.2.5 Surface reconstruction

A surface reconstruction can be obtained through interpolation and approximation [60][61]. These techniques are also categorized into two stages: A parameterization stage for topology, shape and boundary assignment to the surface, and a surface fitting stage [62].

Further, surface representation can be divided into two types: Explicit and implicit [63]. An explicit representation defines a particular location of a surface. The most practiced explicit representation methods are the parametric surfaces and the triangulated surfaces. The B-Spline and NURBS are an example of parametric surfaces techniques that fit local surface patches. NURBS or Non-uniform Rational B-Spline is the generalization of a B-Spline method to be able to process more complicated shapes and non-uniform data set. NURBS surface reconstruction problem can be solved through different approaches, such as, the [64] approach by using linear least squares fitting, or a shape reconstruction of 3D conducting curved plates present in [65]. The triangulation surface reconstruction technique applies the Voronoi diagram and the Delaunay triangulation on a data set dense enough. This approach depends on the quality of the input data set because it can't handle noise [63]. There are other algorithms based on the Delaunay triangulation, like the Crust algorithm, the Alpha shape and the Cocone algorithm. The Crust algorithm based on the three-dimensional Voronoi diagram and ensures that given a "good sample", the output is guaranteed to be topologically correct [66], however, if the sample is too large, then the running time is too slow, and so this algorithm wouldn't be suitable for any problem, thus Power Crust algorithm is an improvement to the Crust algorithm, according to [67]. The alpha shape method is a heuristic approach that derives the surface shape from the Delaunay triangulation in which the minimum area of the triangles is found and the alpha value is adjust based on the point density [68]. The Cocone algorithm determines the triangles to build the surfaces through Delaunay triangulation and it has a faster performance compared to the Crust algorithm [61].

An implicit representation is an isocontour of a scalar function, also known as volumetric representations or function fitting approaches. According to [69] the techniques used in surface reconstruction for implicit representation are the following: The least square method, the Poisson surface reconstruction algorithm,

the PDE method and the Level set method. The main advantage of implicit surfaces is the ability to fill holes automatically due to a better topology [70].

The least square method minimizes the error from the difference between the data value and the model solution to fit the problem behavior. Usually, this approximation technique is combined with an optimization method [71]. The most common variation is the moving least square method that reconstructs the surface from a set of unorganized points by calculating the weighted least square measure around the point that is being reconstructed [72].

Poisson surface reconstruction algorithm processes all of the data to obtain smooth surfaces through data fitting [73][74]. According to [73], this method was improved by applying a linear interpolation to produce a surface similar to a curve and the search method reduced the time of reconstruction. This method has variations such as the Screened Poisson surface reconstruction [75].

Partial differential equation method proved to be useful in image interpolation and compression field by improving topology [63], decreasing the reconstruction time [76] which made possible its application in biomedical image analysis [77].

Oshe and Sethian [63] introduced the Level set method as a technique to process and reconstruct the surface of complex topology and noisy data.

2.2.6 Textured Reconstruction

The last step of 3D reconstruction is the texturing of the mesh previously generated. Generally, texturing a surface is assigning color to each face of the mesh. Before texturing, the model goes under a preprocessing step to decimate the model in order to have larger triangles while maintaining the overall geometry and produce a more effective texturing process [78]. Texturing methods are divided into two main approaches: The Single-view approaches select the best view for each face and the Multi-view approaches that blend together a subset of the images to get a more uniform and consistent image [78].

2.3 3D Reconstruction Software

Nowadays, there are a diverse amount of software tools which can be used to obtain a 3D model from a set of images. These pipelines can be divided into two different concepts: A Structure from motion and a Multi-view stereo approach. A SFM is used for image alignment, implying it estimates the image location and orientation but also the camera parameters. A MVS continues the process of 3D reconstruction by taking the information given by the SFM approach and building a point cloud of the scene. The overall process starts by building a sparse point cloud that consists in matched features of the input images, followed by the construction of the dense point cloud, once the extrinsic and intrinsic information is known from the previous

step. The third stage is the surface reconstruction in which the points are connected to form faces. Lastly, the fourth step consists in reanalyzing the images and applying the estimated texture in the 3D model.

As the first step of 3D reconstruction, SFM is represented by pipelines that execute a three-dimensional reconstruction of an object or a scenery based on images as its input. There are different types of SFM software, and each has distinct steps.

In order to generate an optimal reconstruction, it is fundamental to follow certain features: The object must have a varied texture and asymmetrical geometry so the SFM pipelines are able to correctly estimate the pose of the cameras. Also, the data set must include a sufficient number of images to cover the entire surface of the object. Another important factor is the quality of the images, if the resolution of the sensor and the use of optic angles are too wide and demanding, the pipeline needs to take into account the presence of radial distortions. In case the intrinsic parameters of the camera are known, the pipeline can estimate the distance of points based on the focal length, sensor and image size and therefore, the software can reach better and more accurate results. On the other hand, inaccuracies present in the intrinsic parameters of the camera can lead to imprecise camera pose estimation and point triangulation. Likewise, if the images are cropped, the intrinsic parameters of the camera need to be recalculated. The images need to depict the same area of the object from different viewpoints, therefore the images should be paired to efficiently estimate the 3D points of the reconstruction. Some pipelines improve the reconstruction with more images that cover the same portion of the object [51].

In 2018, an evaluation between the most popular Structure from Motion software was accomplished and discussed in [51]. Among the open source pipelines the mentioned ones were *COLMAP*, *Theia*, *OpenMVG*, *VisualSFM*, *Bundler* and *MVE*. These pipelines were evaluated according to its source code public availability, its customization and effectiveness. According to [51], SFM implementations were tested in which it showed that *COLMAP* had the best average results followed by *VisualSFM* while *OpenMVG* software couldn't perform the reconstruction and had the worst results, overall. *COLMAP* is a 3D reconstruction software released to the research community in 2016 as an open-source implementation. This new SFM method aimed to achieve better results in terms of completeness and robustness [79].

Structure from Motion can be divided into two categories: A global SFM and an incremental SFM. Global SFM is more scalable but susceptible to outliers. Incremental SFM is more time consuming but due to the extensive use of RANSAC and bundle adjustment, it can achieve more robust and accurate results by filtering outliers.

COLMAP introduces a new incremental SFM that focus in 5 main changes: The creation of a scene graph augmentation to improve robustness for the triangulation [79]. The minimization of the reconstruction error by selecting the best views, the image with most triangulated points and with a more uniform distribution, since PnP algorithm depends on the number of observations and their distribution in the image [79]. The

method proposed for triangulation was a sampling-based triangulation method to increase completeness at reduced computational cost [79]. Once the registration and triangulation are performed, the Bundle adjustment is applied locally through *Ceres Solver,* and a global BA is executed after the model has grown by a certain percentage. After performing the Bundle adjustment, the degenerate cameras are checked. Cameras with a large field of view or distortion coefficient are filtered after global adjustment. Observations with large reprojection errors are also filtered. Contrary to other pipelines, such as *VisualSFM*, *COLMAP* proposes a pre and a post-bundle adjustment on the re-triangulation step. Post-bundle adjustment contributes to increase redundancy for the next phases and completeness of reconstruction by continuing to triangulate points that have failed before. The re-triangulation and filtering work are performed in an iterative optimization until the number of filter observations decrease, on contrast with other pipelines that execute bundle adjustment in only one instance [79]. A redundant view mining is applied by clustering cameras with higher scene overlap to efficiently parameterize the BA stage for dense photo collection [79]. Overall, *COLMAP* has the best results comparing to other open-source pipelines in terms of completeness and pose accuracy [51][79].

2.4 3D reconstruction applications

The approach of 3D reconstruction has applications in various fields, from archeology [80], civil engineering [81], medicine [4] to a more recent augmented reality field [82]. Nevertheless, 3D reconstruction in quarrying industry has been applied and demonstrated to have significant advantages in performance. As an example, in 2017 a research was performed in open-pit mines and quarries to reconstruct the quarry topography through UAV imagery [19]. A three dimensional reconstruction of the geomorphic features of a quarry centimetric spatial resolution is important for mine managers to report the amount of extracted material, according to regulations, but also a way to be able to rehabilitate mine sites after intensive quarrying activities [19]. Traditional aerial technologies to survey small areas can be expensive and avoidable through SFM aerial photogrammetry based on images acquired by unmanned aerial vehicles [19]. This technique proved to be reliable, affordable and a validated tool for high resolution reconstruction surveys and quarry monitorization, if taken in to account that the quality of the 3D models depend on the quality of the images [19].

In 2019, an innovative approach to the traditional methods for assessing riprap geometric properties through subjective visual inspection and hand measurements, was presented to provide a convenient, reliable and sustainable solution to accurately evaluate the volumetric properties of large sized aggregates and riprap rocks [20]. This research applies 3D volumetric reconstruction algorithms to images taken under an uncontrolled field lighting and concludes that the study is a promising future potential in replacing subjective visual inspection or time consuming hand measurement process to this convenient, affordable and quality controlled technology [20]. On the same year, unmanned aerial vehicles-based 3D photogrammetric analysis was proven to be effective to inform rockfall behavior at slopes and provide risk assessment design of suitable protection measures [21].

3 Problem formulation

This Chapter introduces the approach and overall measures of obtaining a 3D reconstruction of a quarry Limestone. The adopted camera representation and the input data are presented. The software that is going to be used to obtain experimental results is also presented in Chapter 3, as well as the hardware and software requirements.

3.1 Introduction

As it was referred previously, the focus of this assignment is to build a 3D reconstruction of a quarry limestone. Three-dimensional reconstruction from images, in computer vision, is the process of capturing the shape and geometry of an object or a scenery. There are a variety of techniques to approach this type of problems, however, it can be divided within two main categories: The contact and the non-contact methods. The contact approach adopts a 3D scanner to reconstruct the shape of the object through physical interaction with the object. This form of application has two dominant disadvantages: It requires contact with the object, meaning it can modify or damage it, and it is a process with a significantly slower approach than non-contact methods [83][84]. Furthermore, a contact approach isn't feasible because techniques, such as 3D scanning are mainly applicable to small size objects. Concluding that the non-contact approach is the best method to reconstruct a scene of limestones, this approach can be branched in two methods: The passive methods and the active methods.

The active methods can involve contact or a projection with the object by using a mechanical or a radiometric approach, such as, ultrasound, laser scanners, structured lights, microwaves, and others. On the other hand, the passive methods do not tamper with the reconstruction technique neither interacts with the object, since it uses sensors to measure the radiance reflected or emitted by the object surface, only based on optical imaging techniques [85][86].

In this case a passive method was chosen due to the lack of effectiveness of the active methods performance in sunlight and also the inadequately controlled environment, which both aspects make the application of an active technique in an environment such as a quarry unyielding [87].

The active method approach demands expensive devices which only a few experts can have access to. In opposition, the passive method only requires cheaper devices, such as a simple digital camera or images of the object/scenery connected to a three-dimensional reconstruction algorithm. This fact also contributes to support the choice of a passive approach [87].

The 3D reconstruction of the limestone using a passive method targets a Structure from motion and multiple view stereo algorithms for both open source and commercial pipelines. After acquiring the images, the software starts by applying a structure from motion approach. SFM is formed by 3 methods: The feature extraction and description, the feature matching and the pose location estimation. This phase generates a

sparse point cloud. Once the camera locations are known the next step uses the intrinsic and extrinsic parameters of the camera for each image and the corresponding images to improve details of the scene and build a dense point cloud. Then, the mesh model of the scene is generated in order to visualize better the object and detect cracks. The textured model is built in the end. This framework can be seen in Figure 4.

The feature extraction detects local features that describe the points of interest of the images [1]. The algorithm chosen to solve feature detection will influence the robustness of the features and the efficiency of the matching stage [51]. Image and Feature matching matches these keypoints from distinct images that are overlapping [1]. The pose location estimation aims to find the camera parameters of each image, by computing the essential and fundamental matrix [1]. This step also corrects the camera parameters of each image from the intrinsic and extrinsic parameters. Once the position and parameters of the camera are known the 3D location of points is estimated through triangulation to generate the sparse point cloud. The next stage requires the images of the scene and corresponding camera parameters to build a dense point cloud. The mesh reconstruction approximates the surface of the scene based on the dense point cloud. The texture reconstruction is the mapping of the texture from the images onto the reconstructed 3D surface.

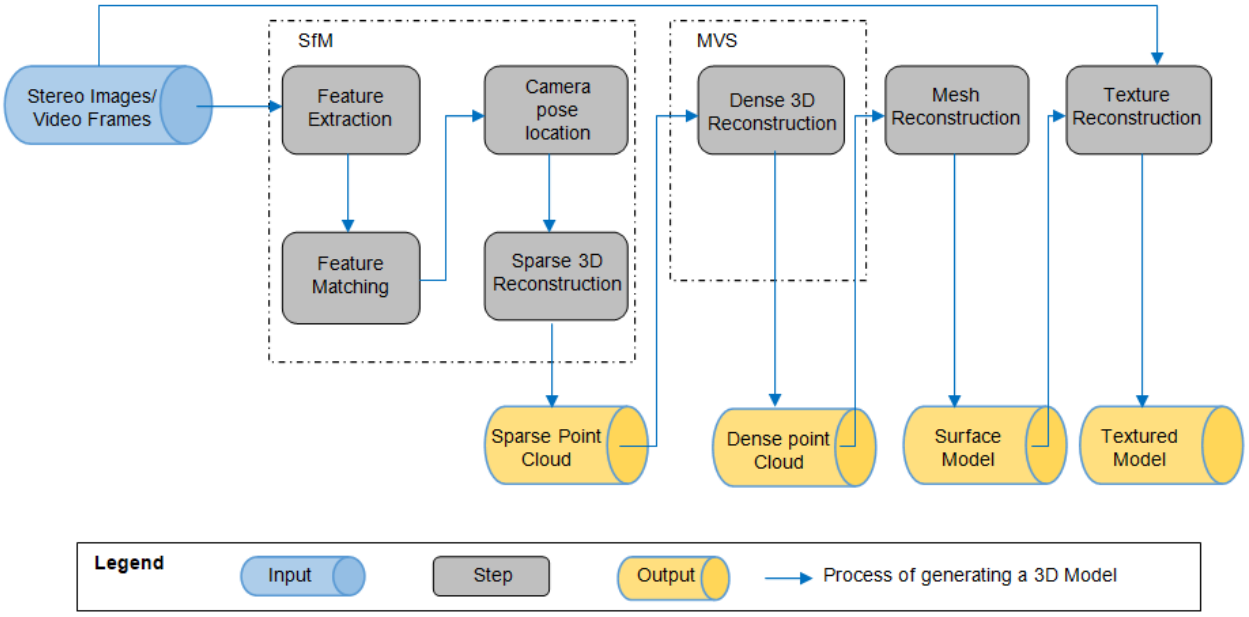


Figure 4: General Framework of a 3D Reconstruction

In order to compare the distinct software and the pipeline effectiveness within the best-chosen software, the result study is divided in 4 stages:

Stage 1: The comparison between pipelines in order to recover the most suitable software to reconstruct 3D models of limestones in an uncontrolled environment.

Stage 2: The comparison between different combination of algorithms within the best software

Stage 3: The comparison between 3D models with known and unknown intrinsic parameters.

Stage 4: The comparison between 3D models with known and unknown intrinsic and extrinsic parameters.

3.2 Camera representation

The physical set up of this assignment was already explained on the first chapter. The adopted cameras were Genie Nano C4020 Color, which is a CMOS sensor of 12,4 MP and their respective lenses of 83,029º of field of view [9][10]. This camera can be represented as a pinhole camera model to portray the fixed internal camera parameters and establish the mathematical relation between the camera coordinate and the pixel coordinate in the image frame. Generally, this model transforms and maps 3D world coordinates to 2D image coordinates. The center of projection is considered to be in the origin of a Euclidean coordinate system and the image plane or focal plane is $Z = f$ [1].

The projection of a 3D world point $(X, Y, Z)^T$ is mapped to the point $(fX/Z\,, fY/Z\,, f)^T$ on the image plane [1]. The final image coordinate can be described as [1]:

$$(X, Y, Z)^T \mapsto (fX/Z\,, fY/Z)^T \qquad 3.1$$

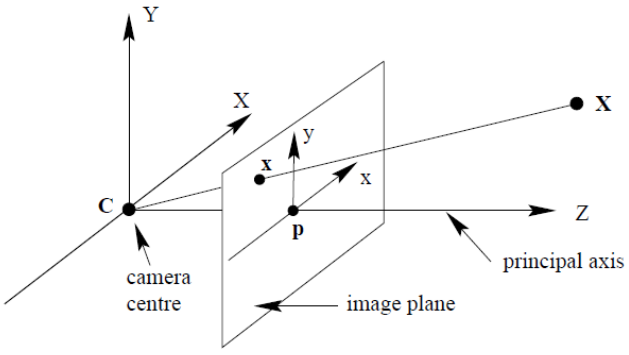


Figure 5: Pinhole camera geometry [1].

The camera center or optical center is the center of projection, the principal axis or principal ray of the camera is the line perpendicular to the image plane and the principal point is the intersection between the principal axis and the image plane, as it is shown in Figure 5

This can be reformulated to be linearly written in homogeneous coordinates as [1]:

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad 3.2$$

The intrinsic matrix [1][88] is represented as:

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad 3.3$$

The matrix K is the camera calibration matrix with the coordinates of the principal point as $(x_0, y_0)^T$, α_x and α_y as focal length in the x-coordinate direction and the y-coordinate direction, respectively, and s is the parameter the describes the skewness of the two image axes.

The distortion parameters, radial distortion (k_1, k_2, k_3) and tangential distortion (p_1, p_2) are represented by the following vector:

$$[k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3]$$

The matrices applied in this study were calculated in [12], in which the intrinsic parameters of the right camera are represented by the following matrices:

$$\begin{bmatrix} 2645.9 & 0 & 1634.1 \\ 0 & 2648.1 & 1246.4 \\ 0 & 0 & 1 \end{bmatrix}, [0.85285 \quad -35.26023 \quad -0.03693 \quad -0.04618 \quad 443.56103]$$

And the left camera intrinsic parameters are the following:

$$\begin{bmatrix} 2647.4 & 0 & 1625.1 \\ 0 & 2649.7 & 1241.5 \\ 0 & 0 & 1 \end{bmatrix}, [0.34260 \quad -8.80952 \quad -0.02080 \quad 0.01760 \quad 10500844]$$

3.3 Data Acquisition

The images taken on the fields of Fravízel company were close range and in an uncontrolled open environment with similar conditions to a Quarry mine. The data acquisition needs to take into account a series of rules, according to software requirements. In terms of object/scene requirements, not textured, shiny, highly reflective or transparent object should be avoided, as well as moving object within the scene, but also flat objects [89]. A good capture of photos should avoid direct light such as in the daylight, shadows and one colored surfaces [90]. In terms of capturing scenarios, it is better to have more photos than not enough, the object of interest should take the maximum area in the scene, flash should be avoided and good lighting is crucial to achieve better quality of results [89]. The images must have a side overlap of, at least, 60% and frontal overlap of 80%, without moving the object or scene [90]. In this regard, the photos were taken with the limestone still and the cameras facing the scene moving, with an angle of 83.029º, in different positions as it is shown in Figure 6.

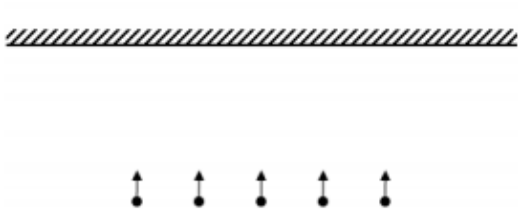


Figure 6: Overall capturing scenario [89]

Some of the data taken in Fravízel company for this thesis can be seen in Figure 7. A video was recorded at the same time from both cameras and there were 68 images recovered. These images were taken in pairs, and so the 1st image and the 35th image were taken at the same time but with the left camera and the right camera respectively.

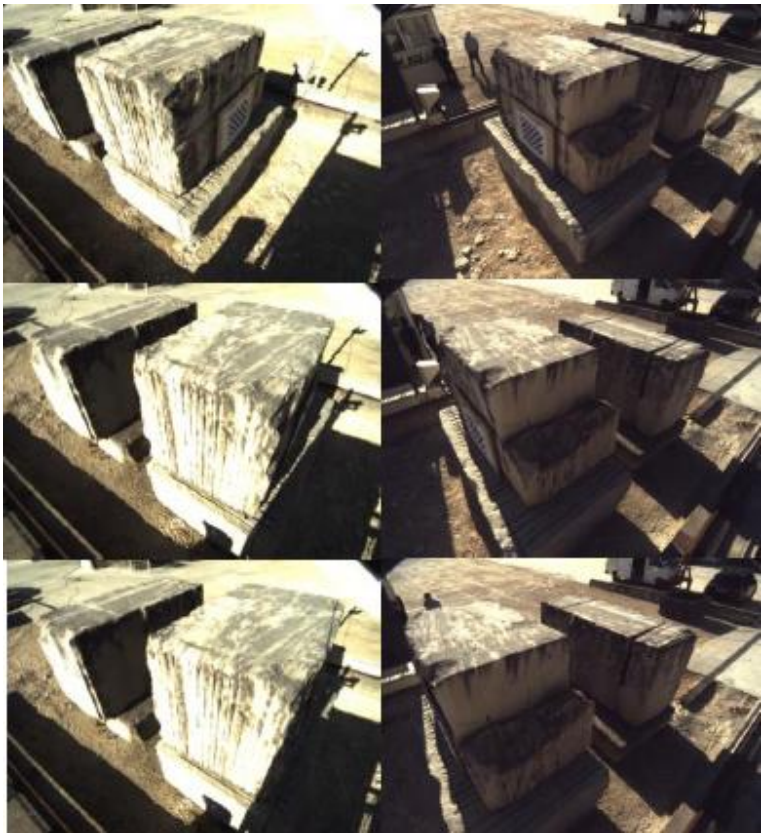


Figure 7: Data collected in a Limestone quarry

3.4 Open source and commercial pipelines

There are various SFM and MVS pipeline implementation proposed through the years. This study focuses in the few most popular open source and commercial software. The open source pipelines used are the following: VisualSFM, COLMAP, Meshroom and MeshLab. The commercial pipeline used is the Agisoft

Metashape. There are different algorithms that can be applied through these pipelines, however it was chosen the most similar path to reach a sparse point cloud for all software.

Along with the most leading software of 3D reconstruction, VisualSFM and COLMAP generated the best results according to [51], therefore these pipelines were chosen to be analyzed. MeshLab is a well known open source system for processing and editing 3D meshes but also commonly used to open the models created by other pipelines such as COLMAP, that can compute a 3D mesh but not illustrate it. Meshroom was selected among other pipelines for being a free, open-source 3D reconstruction software based on the commercial *AliceVision* framework and thus providing quality and robustness to reconstruct, not only sparse and dense point clouds, but also, surface and textured models. Contrarily to other 3D reconstruction pipelines, *Meshroom* developers have recently improved its contents with a major update in 2019 contributing to a better quality and speed in generating 3D models, while competing with some commercial pipelines [90], *Agisoft Metashape* was also chosen for being an acclaimed 3D reconstruction pipeline and for its recent update transitioning from the popular *Agisoft Photoscan* to *Agisoft Metashape* while improving performance.

VisualSFM is a pipeline with a friendly user graphical interface that executes an incremental Structure from Motion and a Multi-view stereo algorithm to reach a sparse and dense cloud reconstruction as a result. A downside of this software resides in the non-availability of algorithms in certain stages of the reconstruction, respectively, the pose camera estimation and triangulation, however it is known that this software detects features through SIFT, matches the images though an exhaustive approach, corrects the camera pose estimation through multicore bundle adjustment and excludes outliers by applying RANSAC [51]. Its output is a sparse cloud reconstruction [91]. Builds a dense point cloud through a CMVS/PMVS method.

COLMAP is the implementation of a C++ open-source algorithm of the incremental SFM and MVS pipeline [92]. The focus of this software is to reconstruct a scene or object in 3D and apply enhancements to conventional SFM pipelines such as robustness, accuracy and scalability. Its creators built an intuitive graphical interface that allows different configurations, regarding the algorithms, so it would be possible to improve and shape to each individual reconstruction [45][93]. The SFM workflow used in this project was similar to VisualSFM, starts by applying SIFT as features extractor, an exhaustive approach for image matching and brute force for feature matching. The pose camera estimation is obtained through PnP and the triangulation step with a DLT method. Ceres Solver is used to correct the camera poses estimated in a pre and post triangulation stage and RANSAC to determine the outliers. The MVS stage applies a CMVS and PMVS methods to obtain the dense point cloud. COLMAP's first step to create a dense point cloud is undistortion of the images. Undistortion process applies a lens correction on the input images and generates undistorted images. Once the images are undistorted, the heaviest step in this pipeline is the stereo reconstruction that relies in the GPU using CUDA. COLMAP uses a Patch based stereo method to generate a dense point cloud by estimating a depth and normal maps in stereo and fuse them to the sparse point

cloud in the fusion step [93]. Although COMAP's GUI can't show the 3D models, this pipeline computes a 3D surface of the scene based on Screened Poisson surface reconstruction or Delaunay triangulation approach.

Meshroom is another open-source pipeline with a user graphical interface that implements a Structure from motion process, a Multiple-view stereo process, a surface and a textured reconstruction. Meshroom uses a node connection GUI approach allowing the user to try a variation of workflows [90]. The procedure of SFM used implements SIFT as the feature detector, matches images through an exhaustive process and the features through a Brute force algorithm. The camera pose is estimated through a PnP in a RANSAC framework and refined though a Bundle Adjustment. In MVS process it is implemented an SGM method to obtain the depth maps. The fusion of depth maps is accomplished in the Mesh phase. The surface reconstruction is accomplished through a Delaunay triangulation and the textured reconstruction by associating each vertex to the texture information.

MeshLab is a mesh processing open source pipeline that can support a variety of 3D formats built in a large stable C++ library. This pipeline can be used as a GUI to paint, smooth and color meshes but it also has cleaning, remeshing, colorization filters, measuring tools and many other applications. However one of the most important applications in this study is the ability of generating meshes based on different algorithms from dense point clouds that were built in other pipelines such as, VisualSFM, COLMAP and others [94][95]. MeshLab builds a 3D mesh of the scene based on a main algorithm: Screened Poisson surface reconstruction from a Dense point cloud

Agisoft Metashape is a commercial advanced image-based 3D modeling pipeline to generate professional quality 3D content from 2D images. The disadvantages of being a commercial software consists in the non-availability of which methods and algorithms are implemented. Nevertheless, the range of applications, stability and constant improvement of the GUI offers a good option for real Quarry application. This pipeline builds sparse and dense point clouds from images, 3D surfaces and textured reconstructions, orthomosaic images and DEM representations [89]. Even though this pipeline is commercial, it is known that the feature detection and matching was based on a similar approach to SIFT to promote higher alignment quality and the camera intrinsic and extrinsic are refined through a similar approach to bundle adjustment.

To summarize, the methods used in each software are organized in Table 2.

Table 2: Algorithms used in each phase of the reconstructions

	SFM							MVS		
	Feature Extraction	**Image Matching**	**Feature Matching**	**View Geometry**	**Camera pose estimation**	**Triangulation**	**Model Correction**	**Dense Cloud Reconstruction**	**Surface Reconstruction**	**Textured Reconstruction**
VisualSFM	SIFT	Exhaustive	Brute-Force (RANSAC)	N/A	N/A	N/A	Multicore BA	-	-	-
COLMAP	SIFT	Exhaustive	Brute-Force (RANSAC)	8point algorithm	PnP in RANSAC framework	DLT	Ceres Solver	CMVS, PMVS	Screened Poisson Reconstruction, Delaunay triangulation	-
MeshLab	-	-	-	-	-	-	-	-	Screened Poisson Reconstruction	-
Meshroom	SIFT	Exhaustive, Vocabulary tree, sequential	Brute-Force, ANN, Cascade Hashing (RANSAC)	8point algorithm	PnP in RANSAC framework	DLT	BA	SGM	Delaunay triangulation	UV mapping
Agisoft Metashape	Similar to SIFT	N/A	N/A	N/A	N/A	N/A	Similar to BA	N/A	N/A	N/A

3.5 Hardware and Software requirements

The hardware used for this thesis is a computer MSI GT72 3QE Dominator Pro. The essential specifications for a reconstruction are the CPU, RAM and GPU. The laptop MSI GT72 3QE Dominator Pro has CPU of the 5th generation Intel® Core™ i7 5950HQ / 5700HQ processor, a 32 GB RAM and a NVIDIA GeForce GTX 980M. The most demanding pipelines are the commercial ones, therefore the minimal hardware requirement configuration was established according Agisoft Metashape specifications as it follows in Table 3.

Table 3: Minimal and recommended configuration of Agisoft Metashape [89].

Minimal configuration	Recommended configuration
Windows XP or later (32 or 64 bit), Mac OS X Mountain Lion or later, Debian/Ubuntu with GLIBC 2.13+ (64 bit)	Windows 7 XP or later (64 bit), Mac OS X Mountain Lion or later, Debian/Ubuntu with GLIBC 2.13+ (64 bit)
Intel Core 2 Duo processor or equivalent	Intel Core i7 or AMD Ryzen 7 processor
4 GB of RAM	32 GB of RAM

The available RAM is an essential requirement for the software to process a certain number of photos. The minimal requirements, 4 GB of RAM, can build a model based on 30 to 50 photos of a resolution of 10 MP. However a 16 GB RAM can process up to 300-400 photographs [89]. The better the hardware configuration the more efficiently the 3D reconstruction process can achieve a 3D model, therefore there are advanced configuration in which the CPU is a Octa-core or hexa-core Intel Core i7 with 64 GB of RAM and a Nvidia GeForce RTX 2080 Ti, TITAN X or TITAN RTX as GPU, and even, extreme configurations with 64GB RAM to process large data sets.

A necessary characteristic is that the GPU NVIDIA should be supported by CUDA in order to accomplish reconstruction in certain pipelines such as COLMAP, Meshroom and Agisoft Metashape [79][89][90]. For this project CUDA was installed for the pipelines to utilize the processing power of CUDA.

In terms of software, most of the popular software work on Windows XP or later (32 or 64 bit), Mac OS X Mountain Lion or alter, Debian/Ubuntu. This assignment was executed in Windows 10.

4 Methodologies of 3D reconstructions

This chapter aims to describe the general workflow and each method that will be implemented through the different pipelines in order to reconstruct a 3D model of a limestone. Various preprocessing methods are analyzed in Chapter 4, in order to enhance the brightness and contrast of the input images.

4.1 Image Preprocessing

The uncontrolled environment of the open pit mine retrieves images with inconsistent lighting and shadows between the two cameras. The brightness and contrast of an image can change how the images are processed in a 3D reconstruction pipeline, by varying the number of detected features and matched points, but also, compromise the estimation of the camera pose location. Between both cameras the lighting should be similar for the software to be able to detect features and consider those as the same scene but also the enhancement of brightness and contrast allows the software to detect more features and therefore tie points. Since the output information from the SFM process is crucial to apply MVS and build a surface reconstruction, the higher quality is the camera parameters, their pose location and the overall points of the sparse point cloud, the more correct and authentic is the dense point cloud and the surface model.

The original images are the 68 images taken in the Fravízel company, 34 images were captured by the camera on the right and 34 images were taken by the camera on the left. As it is shown in Figure 8, the images from the first three rows and a half are considered to be the camera on the left and the rest of the images are the camera on the right. At the time of the day that the images were captured, the right camera retrieves images brighter than the left camera.

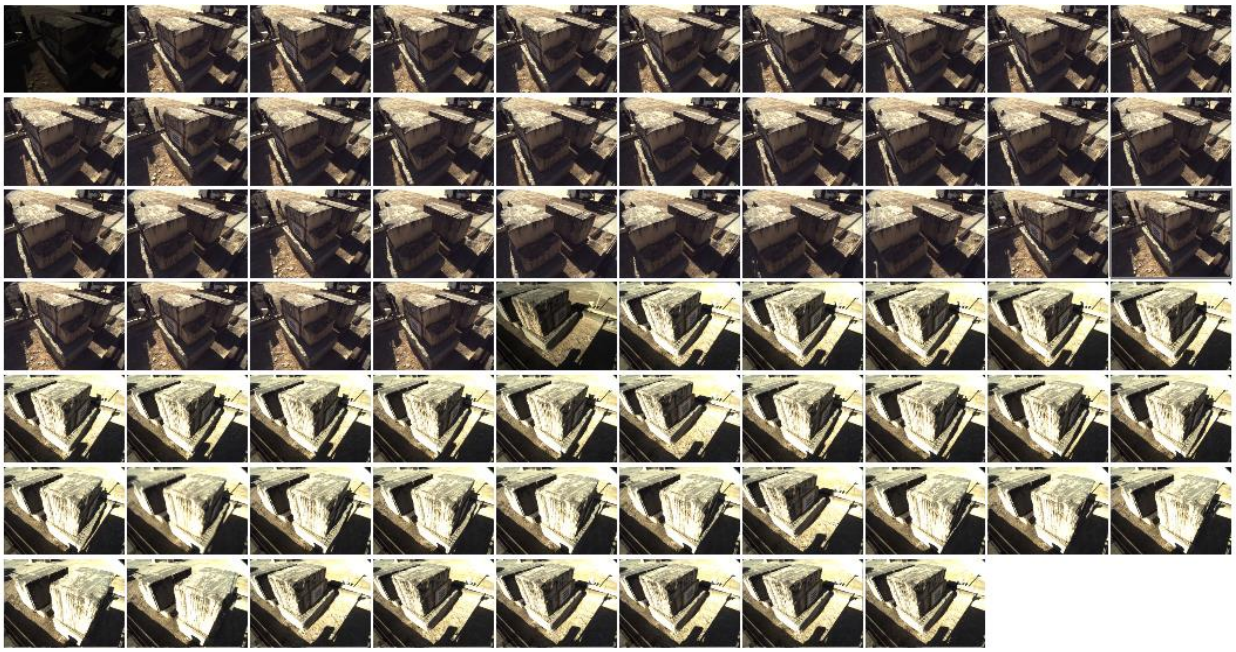


Figure 8: Original images

There were 6 methods applied to these images to improve their quality in order to build a good 3D model. The preprocessing methods were the following: An averaging method, histogram equalization, CLAHE, gamma correction, gamma correction with histogram equalization and gamma correction with CLAHE. These methods were all implemented in *python* through two modules, the *Pillow* and *OpenCV*.

4.1.1 Averaging Method

The averaging method aimed to compute the perceived brightness and contrast to all images and average them. Brightness is the overall lightness and darkness of the image and contrast is the difference in brightness between regions of an image. The perceived brightness of the original images was calculated through the arithmetic mean of each color band in the image (RGB values). Considering RGB color space as a cube where each of the three colors are an axis, a corner of this cube corresponds to black with RGB (0,0,0) and the opposite corner to white with RGB (255,255,255). And so, the perceived brightness is the 3D distance between the first corner to the actual color value for each band, which in this case is the mean of the RGB values of the image. Also, it is necessary to take into account that some colors look brighter than others by giving a different weight to each axis, as it is shown in equation 4.1 [96].

$$Brightness = \sqrt{0.241R^2 + 0.691G^2 + 0.068B^2} \qquad 4.1$$

The perceived contrast can be calculated by converting the image into a CIELAB color space that expresses three values: L for lightness, A from green to red and B from blue to yellow. Once the lightness values for each pixel of the image are computed, two images are created with the maximum and minimum values through dilation and erosion respectively. A 5 by 5 Mask is run through each pixel and replacing its value

with the maximum or minimum value within the mask for dilation and erosion respectively. The contrast of the image is calculated by subtracting the eroded image from the dilated image and dividing its sum. Then, the average of the contrast matrix of each original image is computed, according to equation 4.2 [97].

$$average\ contrast = \frac{\sum_{i,j=0}^{i,j=m,n} \left(\frac{max - min}{max + \min}\right)_{i,j}}{m \times n} \qquad 4.2$$

where max and min are $m \times n$ matrices of the dilated and eroded images respectively.

Once the perceived brightness and contrast are known, it is calculate the average of the brightness and contrast of the image from the camera in the left and right without taking into account the first image in case of the perceived brightness and the first and 18th image in case of the perceived contrast for being outliers as it is shown in Figure 9. Also, the overall brightness and contrast average is calculated without the same considered outliers.

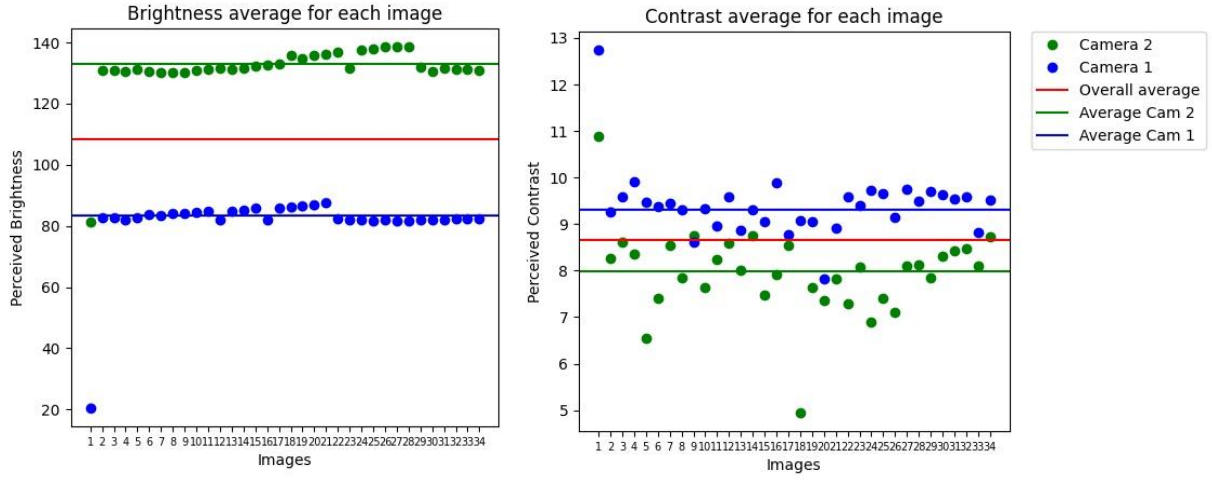


Figure 9: Perceived Brightness and Contrast average of Original Images

In Figure 9 and Figure 10, it is shown the average perceived brightness and contrast, respectively, of the 34 images from the right camera, as Camera 1, in blue and the 34 images from the left camera as, Camera 2, in green. In red, it is represented the overall average of brightness and contrast of the 68 images. In order to have images that can be perceived as the image from the same scene and smooth the light of all images in order to have images with similar brightness and contrast it was applied an interpolation between the original images and an image with only black pixels with same size as the first image. This interpolation is implemented according to a certain factor. This factor varies according to the images on the left and on the right and if it is for brightness or contrast application. For the images on the right and on the left, in the case of brightness, this factor is the percentage of the absolute value of the difference between the overall average brightness and the average brightness of the right (Camera 1) and on the left images (Camera 2), respectively. In the case of contrast, the images on the right and on the left, this factor is the percentage of the absolute value of the difference between the overall contrast and the average contrast of the right

(Camera 1) and on the left images (Camera 2), respectively. This factor certifies that the interpolation is between $(1 - factor\ \%)$ of the black images and $(factor\ \%)$ of the original images. This method is able to ensure that all images tend to the same overall average brightness and the same overall average contrast, as it is shown in the graphs bellow. Nevertheless, the outliers continue to be outliers and would need to be removed. Further, even though the images have the same average perceived brightness and contrast, the images continue to have varying lightness value pixels within the images.

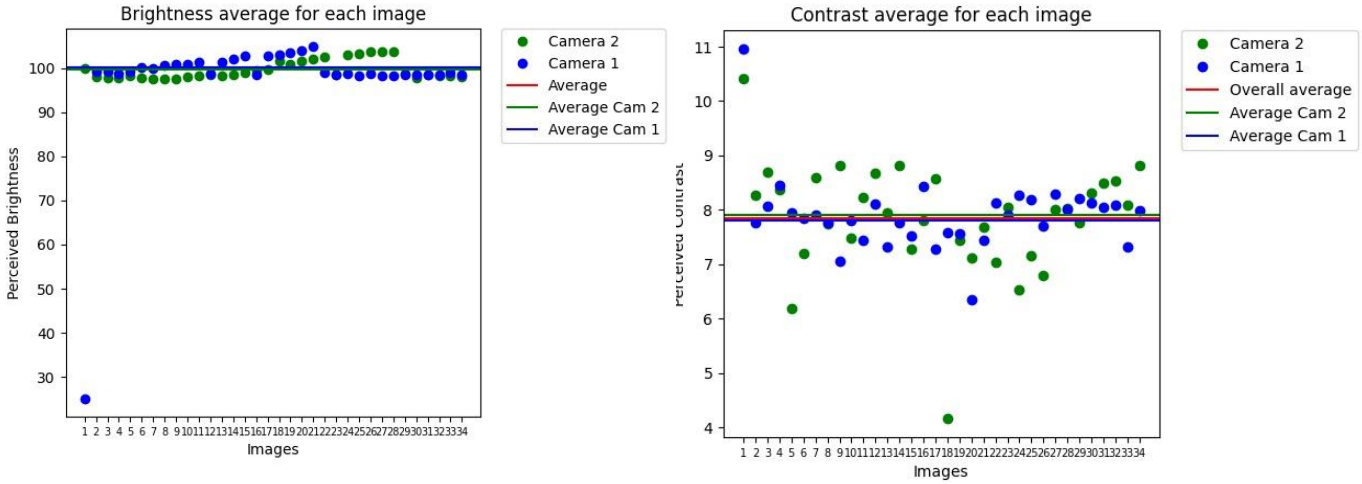


Figure 10: Perceived Brightness and Contrast of averaging method images

4.1.2 Histogram Equalization

Histogram equalization is an image processing technique of contrast enhancement. The histogram represents graphically the intensity distribution of an image with pixel values in X-axis (ranging between 0 to 255) and the corresponding number of pixels in the image on the Y-axis. Therefore, histogram equalization stretches the intensity range of the images to produce a good image with pixels of different intensity values instead of having peaks, concentration of pixels with small range of intensity value, as shown in Figure 11.

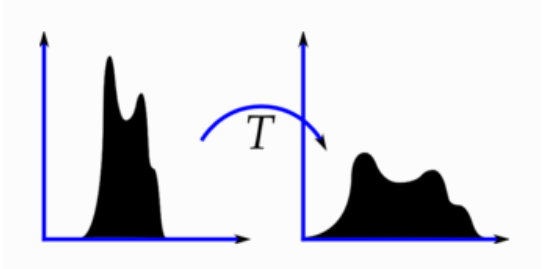


Figure 11: Histogram Equalization [13]

The Histograms of the original images were computed in order to obtain the intensity values of each pixel from their grayscale image and to organize this information in a graph. An example of the histogram of the

pair image 20 for both cameras is shown below, where the image from the left camera is the left graph and the image from the right camera is the right graph. According to the histograms of the original images from the left camera, in Figure 12, the images have around 900,000 white pixels, which shows in the histogram as the biggest peak of intensity values in the images, and a smaller peak that represents around 200,000, almost, black pixels. On the other hand, the histograms of the original images on right camera only have one main peak that shows around 300,000 dark pixels of, around, 50 intensity value. From this histogram analysis and based on the original images, the images taken in Fravízel company environment returned bright images from the camera on the left and darker images from the camera on the right.

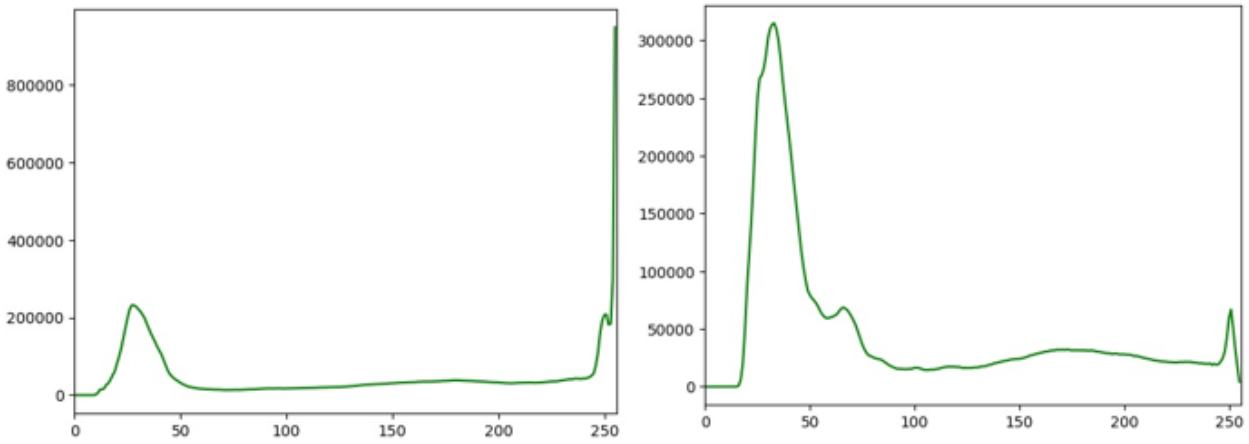


Figure 12: Histogram of the original images (images 20)

Histogram Equalization aims to stretch the intensity range of these images by calculating the cumulative distribution function (CDF), which is the cumulative sum of all of the probabilities of a pixel value (p_x) to occur in an image seen in equation 4.3 [13] .

$$cdf_x(i) = \sum_{j=0}^{i} p_x(x = j), \qquad where\ p_x = \frac{n_i}{n} \tag{4.3}$$

where i is the pixel value that can vary between 0 and 255, n_i is the number of pixels with a certain intensity n is the total number of pixels in the image.

Once the cumulative distribution function is known, it is applied a transform onto the original image to produce an image with a flat histogram through a linearized distribution function along with the intensity range of these images by multiplying the cumulative probability of each pixel intensity by the range of the intensity that wants to be stretch which in this case is 256 intensity values. Finally, the decimal values obtained through these calculations are rounded to their lower integer value.

The results of histogram equalization on the original images are shown as an example in Figure 13, where the image 20 from the left camera is displayed in the left and the image 20 from the right camera is presented on the right. The Figure 13 demonstrates that the histograms of the images from the left camera were flattened, in which, the darker peak of pixels disappeared, meaning that there aren't as many darker pixels

as in the original image and, although there is still a peak of light pixels, the whiter pixels still decreased from 900,000 to 600,000 pixels. The histograms of the images from the right camera show that the histogram equalization distributed the pixels through different intensity values, with an almost even variation of peaks around an average of 50,000 pixels, eliminating the main peak of pixels of the 50 pixel value. The images from both cameras with histogram equalization implementation have less contrast and even tones in relation to the original images.

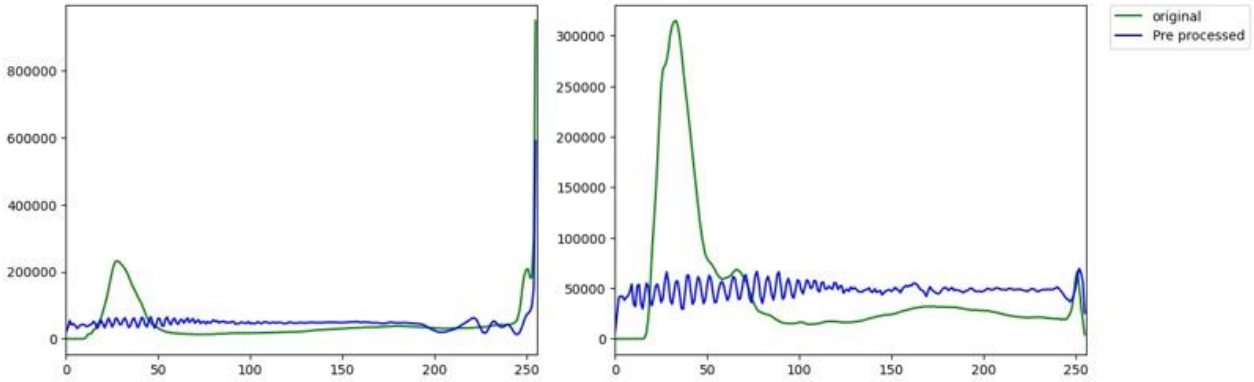


Figure 13: Histogram Equalization (images 20)

4.1.3 CLAHE

CLAHE or contrast limited adaptive histogram equalization is another image processing technique to enhance contrast. This approach is more robust than normal histogram equalization for being an adaptive histogram equalization. The process lies in computing several histograms to different sections of the image in order to redistribute the intensity values of the image. It also avoids the possible noise caused by over amplification by applying a contrast limiting method to each neighborhood from where the transformation was established.

The procedure starts by splitting the image into R, G and B, and for each image channel, the images are divided into small blocks of 8 by 8 elements and applying histogram equalization on them. If this small area has noise, this noise will be amplified, therefore to improve this adaptive histogram equalization, it is applied a contrast limit by establishing that, if any histogram bin is above a certain threshold (in this case 40 pixel intensity value) than those pixels are distributed uniformly to other bins before applying histogram equalization [13]. Afterwards, the histogram equalization is applied in the same way as it was described previously in this project. Lastly, the R,G and B output for each image are merged

The results of CLAHE implementation on the original images displayed in Figure 14, show that the image 20 from the left camera is presented on the left, representing the left images, as well as the image 20 from the right camera is on the right. The left graph shows that the darker pixels dropped around 100,000 pixels and the white pixels also decreased, around 100,000 pixels. These pixels were evenly distributed along the

other intensity values. Despite the images from the right camera still show a big amount of pixels in darker intensity values with a histogram peak around 50 intensity value, in relation with the histogram of the original images, the histogram of CLAHE implementation show that 150,000 pixels from the peak of darker pixels were distributed to different intensity values, between 50 and 150 pixel value. The images from both cameras with CLAHE implementation have less contrast than the original images

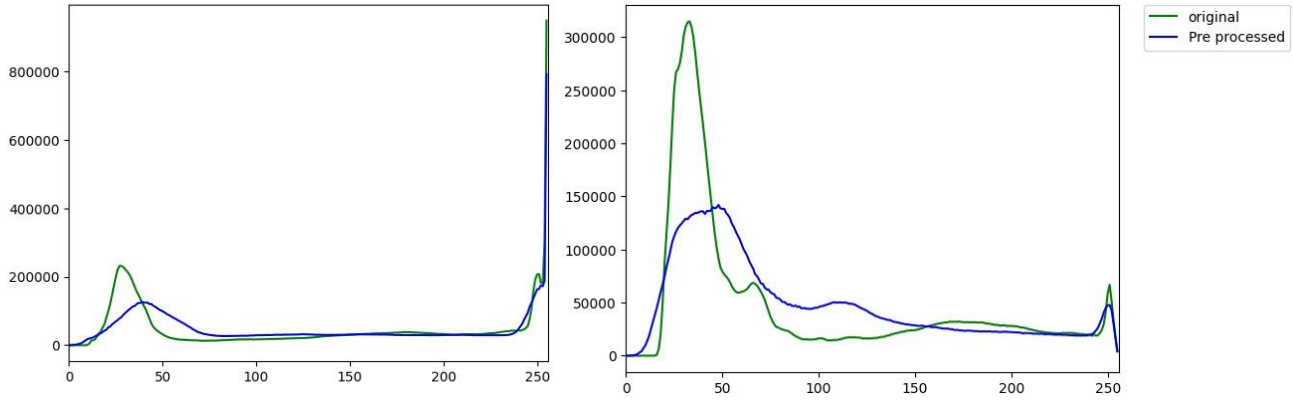


Figure 14: CLAHE (images 20)

4.1.4 Gamma Correction

Gamma correction is another image processing technique that enhances the contrast and brightness of an image through a nonlinear transformation between the input and the output values. This method is also known as the Power Law Transform and it can be written according to equation 4.4 [13]. The value of gamma γ is changed to choose its best value in order to produce the best output image. When $\gamma < 1$ the dark regions of the original image get brighter and the histogram is shifted to the right and when $\gamma > 1$ the original brighter regions turn darker and the histogram is shifted to the left.

$$O = \left(\frac{I}{255}\right)^{\gamma} \times 255 \qquad 4.4$$

where γ is the gamma value, O is the output image values, I is the input image values for an image with values between 0 and 255.

Gamma adjustment can be implemented by creating a table with 256 elements computing the equation 4.4 where I is a number that ranges between 0 and 255 and gamma is a given variable. The adjusted image is obtained by applying a gamma and a lookup tables (LUTs) function, from *openCV*, where it is performed a transformation that assigns a new pixel value to each pixel of the source image, according to the table previously created. This method is less computationally expensive than computing the equation 4.4 for each 12,000 pixels per image.

Based on the images with gamma correction and analyzing the results through their histograms, the chosen gamma correction was $\gamma = 0.4$ for both left and right images. The main objective was to lower the brightness in the images from the left camera, however, by applying $\gamma > 1$ to decrease the brightness on the left images there is a trade-off between decreasing the white pixel peak and decreasing the overall brightness of the images by increasing the darker pixel values. The histograms from Figure 15 show that the intensity values shift to the left producing images that are too dark for the small decrease of the white pixel peak, making it impossible to produce similar brightness for left and right images.

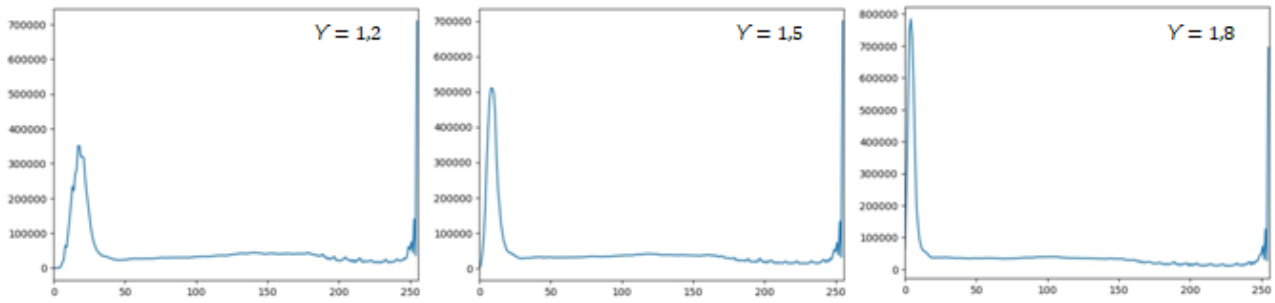


Figure 15: Gamma correction on image 20 of the left camera

Therefore, it was chosen to transform the original images from the right camera to brighter images, as well as the images on the left, to shift the histogram peak of darker pixels to intensity values that are neither too dark nor too bright, around 127 pixel value. Therefore, as a result of different gamma correction trials, it was determined a gamma correction of $\gamma = 0.4$ for both left and right images. As it is displayed in Figure 16, the left histogram corresponds to the images from the left camera with gamma correction of $\gamma = 0.4$ and the right histogram corresponds to the images from the right camera with gamma correction of $\gamma = 0.4$. The left histogram presents a shift of the darker pixel peak to 127 intensity value without increasing the white pixel peaks of the original histogram and the right histogram also shifts the darker pixel peak to 127 pixel value and most pixels are shifted to a brighter region, comparing to the original histogram. The images from both cameras with gamma correction implementation have less contrast and lighter tones than the original images, but also around 200000 pixels of 127 pixel value, approximately.

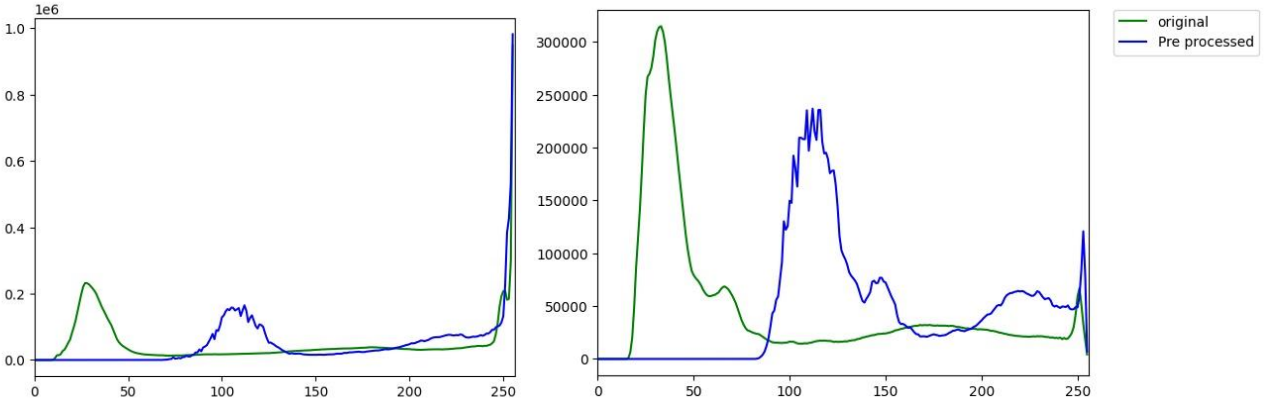


Figure 16: Gamma correction of 0,4 (images 20)

4.1.5 Gamma correction with Histogram equalization

Gamma correction with Histogram equalization was another implemented method with the objective of stretching the intensity range of images that already have been gamma corrected to have similar brightness. This method is the union of the 2 approaches already explained. First, a gamma correction of $\gamma = 0.4$ was applied to all images and then a histogram equalization was implemented to all corrected images.

In Figure 17, the left graph corresponds to the histogram of the gamma correction and histogram equalization implementation of the image 20 from the camera in the left, and the right graph, it's the same histogram implementation, but for the image 20 of the camera in the right. The left histogram shows that the overall pixels were distributed between all pixel values and the white pixel peak decreased, approximately, from 900,000 pixels to 600,000 pixels compared to the original histogram. The right histogram presents an overall even intensity value for all pixels within the images, varying from 45,000 to 55,000 pixels for each pixel value, instead of a 30,000 pixel intensity peak in darker pixels values, present in the original histogram. The images from both cameras with gamma correction and histogram equalization implementation have a more even tone, therefore, less contrast and an overall lighter shade in comparison to the original image.

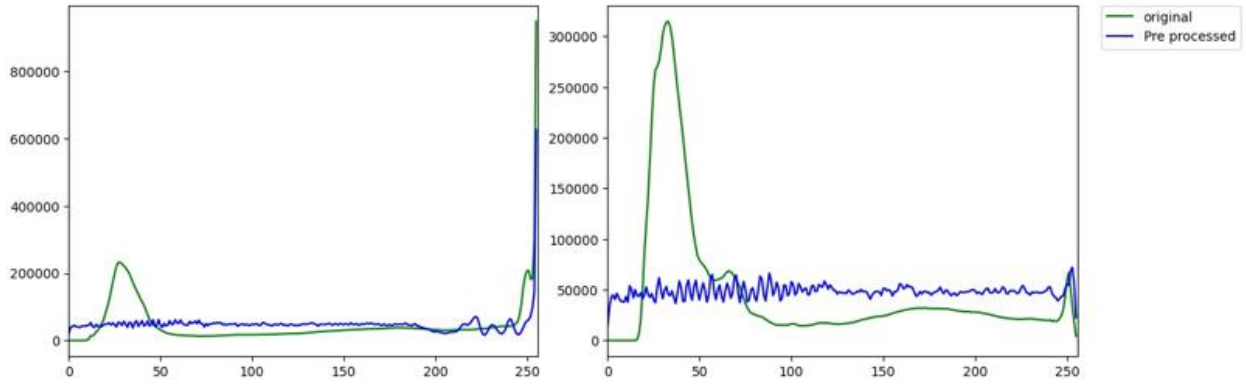


Figure 17: Gamma correction and Histogram equalization (images 20)

4.1.6 Gamma correction with CLAHE

Similarly to the concept of gamma correction with histogram equalization, Gamma correction with CLAHE is the association of 2 methods previously explained. However, in this case, a CLAHE is applied on images that have already been gamma corrected to have similar brightness. This CLAHE implementation is identical to the previous approach, implying that the histogram equalization is enforced on small blocks of 8 by 8 elements and the contrast limit is 40 pixel value.

In Figure 18, the left graph corresponds to the histogram of the gamma correction and CLAHE implementation of the image 20 from the camera in the left, and the right graph, it's the same histogram implementation, but for the image 20 of the camera in the right. The left histogram still displays uneven number of pixels for different intensity values, however with this implementation there's no darker pixel peaks, only a slightly bigger amount off pixels with, approximately, 100 pixel value. The white pixel peak decreased 100,000 pixels compared to the original histogram. In the right histogram, instead of a dark pixel peak, shown in the original histogram, it produced a peak around 127 intensity value, with almost all pixels revolving around this pixel value or whiter values. The images from the right camera with gamma correction and CLAHE implementation are brighter and from the left camera, are greyer compared to the original images.

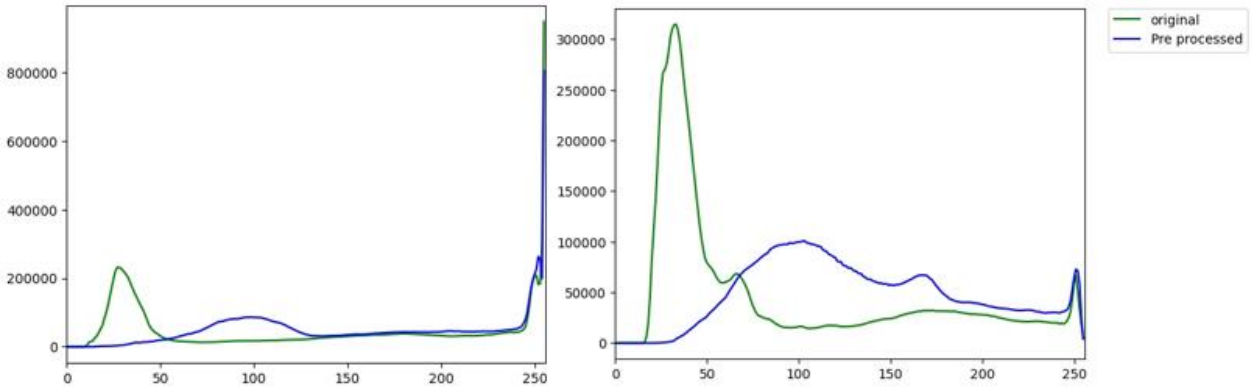


Figure 18: Gamma correction and CLAHE (images 20)

4.1.7 Preprocessing method comparison

The different approaches previously implemented are compared in this section in order to sort out the best input images to build a 3D model. The comparison of the 6 preprocessing methods is done by processing the images from each implementation through a 3D reconstruction software. Since the first part of 3D reconstruction, Structure from motion, is the basis for a good reconstruction and therefore, crucial to have a correct MVS computation, as well as, a surface and texture reconstruction, each step of SFM will be evaluated and compared. The software chosen to reach to the best preprocessing method and its output images was COLMAP for being one of the best open source software for SFM implementation [51][79] .

The preprocessed images for each approach were input in COLMAP to begin the SFM procedure. In the first phase, the features were detected to be matched through SIFT and exhaustive brute force, respectively. On the second phase it was estimated the pose camera through PnP and corrected by bundle adjustment. The overall number of features detected for all images in each preprocessing method, the operation time, the matching point operation time, the process time of pose camera estimation and the number of tie points are displayed on Table 4 and Table 5.

The feature detection elapsed time was very similar for each preprocessing approach, around 0.2765 minutes, but also identical to the feature detection process time of the original images. The feature matching processing time varies between each preprocessing method: The gamma correction with CLAHE approach takes more time than any other, approximately 8 minutes, followed by, the histogram equalization and gamma correction with histogram equalization that takes, around, 7 minutes, CLAHE approach takes 6 minutes and the fastest approaches were the average method and the gamma correction with an elapsed time of, approximately, 5.2 minutes, also compared to the original images that took 5.6 minutes. The camera pose location is the approach that takes more time in Structure from motion, most preprocessed images took 13 to 14 minutes, including the original images, excepting the gamma correction with CLAHE that takes 16.7 minutes, the histogram equalization taking 21.5 minutes of elapsed time and CLAHE with the longest processing time of 32.7 minutes. The sum of the time of these three phases is the total time of sparse point cloud reconstruction, therefore the slowest approach was CLAHE with an elapsed time of 39.3 minutes and the fastest approach was the gamma correction method with 18.9 minutes.

Table 4: Elapsed time for each SFM procedure

	Elapsed time (min)			
	Feature detection	**Feature matching**	**Camera pose location**	**Sparse Point Cloud Reconstruction**
Original	*0.267*	*5.627*	*13.109*	*19.003*
Average Method	*0.26*	*5.412*	*14.084*	*19.756*
Histogram Equalization	*0.289*	*7.579*	*21.522*	*29.39*
CLAHE	*0.291*	*6.333*	*32.675*	***39.299***
Gamma correction	*0.265*	*5.152*	*13.574*	*18.991*
Gamma correction with Histogram Equalization	*0.277*	*7.657*	*14.913*	***22.847***
Gamma correction with CLAHE	*0.277*	*8.117*	*16.647*	***25.041***

The number of detected features and the total number of points of the generated sparse point cloud are related. If less features are detected, then the faster is the reconstruction, the same works in the opposite case. Nevertheless, CLAHE approach is automatically excluded for having the largest time period to reconstruct and still corresponding to the method with the 4th more features detected and reconstructed points. Gamma correction is also refused for having the smaller amount of number of features detected and reconstructed points, explaining the fact that gamma correction was computationally the fastest method. The original images and the average method images retrieved very similar results in terms of processing time, number of features detected and number of points in the sparse point cloud, however, both methods revealed an average of 745,371 detected points, which corresponds to, approximately, 100,000 points less than other approaches, also the total number of reconstructed points were an average of 47,694 point, which is also 10,000 points less than other methods.

The best three methods were gamma correction with CLAHE, histogram equalization with Gamma correction, and histogram equalization. The method that achieved more features detected and sparse points was the gamma correction with CLAHE, as the third slowest approach. Histogram equalization returned the second greatest number of detected features and reconstructed points, yet, this method was the second slowest approach. Gamma correction with histogram equalization produced the third biggest number of detected features and sparse points, between the three best methods, and was the fourth slowest approach. In sum, gamma correction with CLAHE method achieved more sparse points, secondly the histogram equalization approach and thirdly the gamma correction with histogram equalization method. The difference between the histogram equalization and the gamma correction with histogram equalization method was just of 517 sparse points, but the processing time was of 6,543 minutes. This means that the gamma correction with histogram equalization method returned only 0.9% less sparse points but was 22.3% faster than the histogram equalization method, therefore the histogram equalization was excluded from being the best preprocess. The last comparison is between gamma correction with histogram equalization and gamma correction with CLAHE approaches. The difference between these two methods was only of 768 reconstructed sparse points and 2.194 minutes of processing time, meaning that the gamma correction with histogram equalization method obtained 1.3% less sparse points, but took 8.8% less time to generate a sparse point cloud compared to the gamma correction with CLAHE method Therefore, the best preprocessing approach to build a 3D model of a quarry stone in an uncontrolled environment is the gamma correction with histogram equalization approach, according to the software and images used.

Table 5: Feature detection and Sparse cloud reconstruction

	Total Number of features detected	**Total Number of points in Sparse Cloud**
Original	*746,079*	*47,705*
Average Method	*744,663*	*47,683*
Histogram Equalization	*824,746*	*57,742*
CLAHE	*779,481*	*52,554*
Gamma correction	*721,560*	***46,998***
Gamma correction with Histogram Equalization	***818,405***	***57,255***
Gamma correction with CLAHE	***836,234***	***58,023***

And so, the best preprocessed images are the original images that were subjected by a gamma correction followed by a histogram equalization. As shown in Figure 19, the first image was completely recovered from the dark tone original image, seen in Figure 8. Further, the difference in brightness and contrast between the images from the camera on the left and on the right are almost unidentifiable. This set of preprocessed images are going to be adopted for the rest of this study in order to reach to the best results.

Figure 19: Preprocessed images through Gamma correction and Histogram equalization

4.2 Sparse Point Cloud Reconstruction

The Sparse Cloud Reconstruction workflow starts by detecting features, matching features and ends by estimating the pose location to build a Sparse point Cloud. Although these are the main steps, other crucial methods are discussed to remove outliers, correct the model parameters and to determine the position of the 3D point. This phase of 3D reconstruction is the foundation for the next steps, therefore, the quality of results this stage achieves influences the results in the next steps.

4.2.1 Feature detection and extraction

The pipelines adopted on this assignment implement SIFT or its variations to accomplish feature extraction. SIFT is patented by the university of British Columbia and described by one of the most influential research in Computer Vision [23]. SIFT stands for scale invariant feature transform which can be applied to different size images, different depth and the most advantageous characteristic of this feature detection algorithm, it is scale invariant. SIFT transforms image data into scale-invariant coordinates. The goal of this algorithm is to extract invariant features to be able to correctly match against a large database of features from many images. The features are invariant to image scale, rotation and robust to change in 3D viewpoint, noise and affine distortion [23].

The advantages of this approach are the locality, distinctiveness, quantity and efficiency. Locality because similarly to the Harris corner detection, SIFT detects local features therefore it is robust to clutter and occlusion. Individual features can be matched to a large database of objects thus It has distinctiveness. Since many features can be generated and processed even for small objects, quantity is an advantageous factor to return better reconstruction. And it is efficient because it performs as fast as almost real-time performance [23].

As it was mentioned previously, SIFT is divided into 4 main steps: Scale space extrema estimation, keypoint localization, keypoint orientation assignment and the generation of a local image descriptor for each keypoint [23].

The estimation of the Scale space extrema is described as the creation of a Scale space with the intention of calculating the DoG to find the maxima and minima in the DoG images. According to Lowe [23], The most efficient method in order to get rid of the details and ensure that false details are not introduced is to perform a Gaussian blur. The objective is to create a Scale space that progressively generates blur onto the original image, building the first octave, and then resizing the original image to half size and blurring the images again, this process is shown in Figure 20. This process repeats itself until, as SIFT suggests, 4 octaves and 5 blur levels are completed, as the ideal scale space for the algorithm. And so, there are four Octaves, which are images of the same size vertically and each with 5 images. Blurring is mathematically a convolution of the Gaussian operator and the image. Gaussian “blur” applies an expression to each pixel as follows [23]:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \qquad 4.5$$

With L as the blurred image, G as the Gaussian blur operator, I is an image, (x, y) are the coordinates of the pixel location and σ is the scale factor.

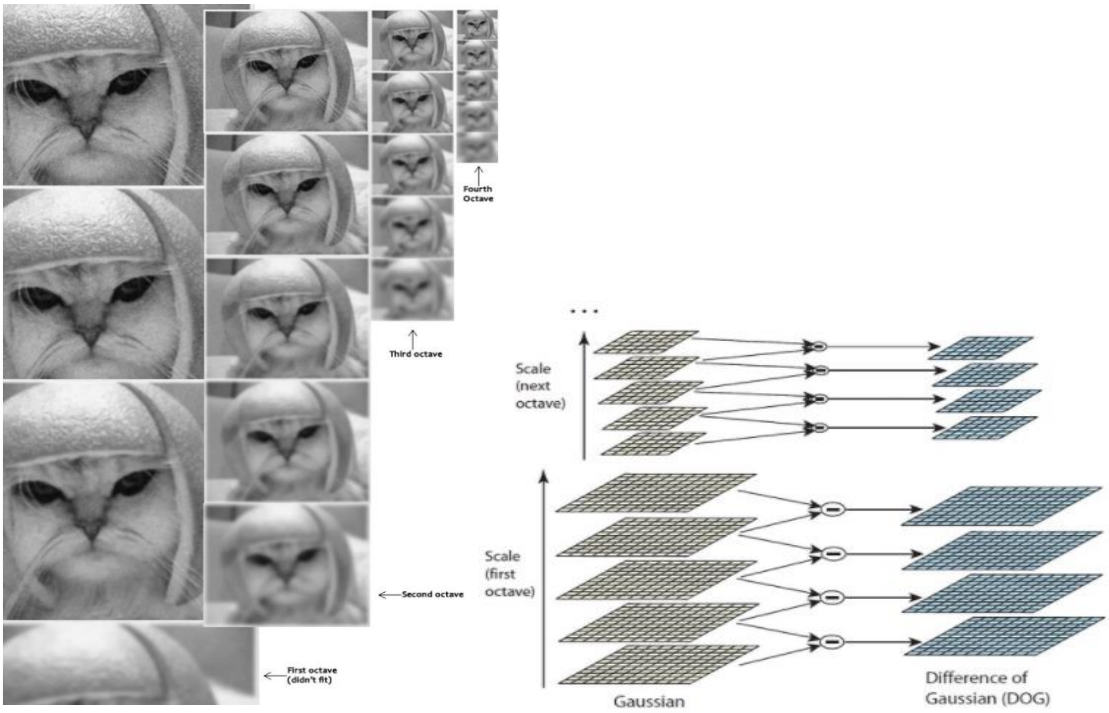
The expression of the Gaussian blur operator can be written as follows:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \qquad 4.6$$

The amount of blurring in each image is also important. Assuming the amount of blur in a particular image is K, the next images will be $K\sigma$. According to Lowe, the optimal values are 4 octaves, 5 scale levels, a σ initial of 1,6 and $K = \sqrt{2}$ [23].

The next step is to execute a LoG approximation. This step computes the difference of Gaussian, shown in Figure 20, as an approximation of the Laplacian of Gaussian, in order to replace the computationally complex step of calculating the second derivative for each blurred image. Further, the LoG approximation isn't scale invariant, the σ^2 of the denominator in equation 4.6 represents the scale, and only by disposing of σ^2 the process turns to be scale invariant. However, after the DoG operation, the resultant images are multiplied by the σ^2, and therefore scale invariant [23].

Figure 20: Scale space and LoG approximation [23]



The following step is to find keypoints. Finding keypoints is branched in two stages: The localization of the maxima and minima in the DoG images and the detection of the subpixel maximum and minima.

In order to locate the maxima and minima in DoG, every pixel is iterated and checked through all neighbors [23]. This verification is performed to the current, above and below image. As it is shown in Figure 21, the X marks the current pixel and the green circles mark the neighbor pixels. The X can be marked as a keypoint if it is the greatest or smallest of all 26 neighbors. Also, the non-maxima and non-minima don't have to go through all the verifications since they are usually discarded on initial steps, as well as the lowermost and top most scales that aren't able to detect any keypoints due to insufficient neighbors for comparison.

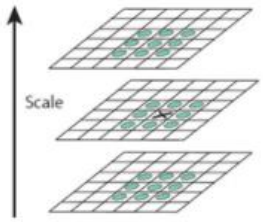


Figure 21: keypoint location [23]

Once this approach is finished, the marked points are the approximate maxima and minima. It corresponds to approximations because the maxima and minima almost never lie exactly on the pixel, but somewhere between the pixels. Therefore, it is necessary to locate the subpixel. The subpixel values are calculated through a Taylor expansion of the image around the approximate keypoint [23]:

$$D(x) = D + \frac{dD^T}{DX}X + \frac{1}{2}x^T\frac{d^2D}{dx^2}x \qquad 4.7$$

By differentiating and equating the expression 4.8 to zero, the solution is the subpixel keypoint location. The calculations of these subpixels are crucial for a better matching and stability of the algorithm. Lowe [23] recommends to generate two extrema images, and so it is needed 4 DoG images, and 5 Gaussian blurred images to produce them.

Some keypoints are found on edges or areas with not enough contrast. These keypoints need to be discarded because they don't return useful features. The edges are removed according to the Harris Corner Detector and the low contrast features are discarded based on their intensities. The method of removing low contrast features begins by identifying if the magnitude of the intensity at a current pixel in DoG image is less than a certain value, if it is true, this feature is rejected. It is applied the Taylor expansion for the intensity value at a subpixel location to be known. Removing edges imply the calculation of two gradients at the keypoint which are perpendicular to each other. The image around the keypoint can be a flat region, meaning that both gradients are small. An edge, where the gradient perpendicular to the edge is big and the gradient along the edge would be small. And a corner in which both gradients are big. The only features that can't be rejected are the corners for being a useful feature. Consequently, if the intensity at the extrema location is less than a certain threshold value, being the recommended value 0,03, then it needs to be

rejected, according to Lowe. DoG has a higher response for edges, since edges can be unstable to small amounts of noise and so it is essential to remove them buy detecting them through a similar approach as the Harris Corner Detector algorithm but instead of calculating the eigenvalues, SIFT method only calculates the ratio between those eigenvalues [23].

The Harris Corner Detector [98] is a mathematical operator that finds features in an image. In SIFT, the Harris Corner detector is implemented with the intention of detecting edges to remove them. It is fast to compute and rotation, scale and illumination variation independent. This detector creates a window when moved in any direction, translated as:

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2 \quad 4.9$$

Where E is the difference between the original and the moved window, u is the window displacement in the x direction, $w(x,y)$ is the window at position (x, y) acting like a mask, ensuring that only the desired window is used, I is the intensity of the image at the position (x, y), $I(x+u, y+v)$ is the intensity of the moved window, $I(x,y)$ is the intensity of the original window.

The aim of this algorithm is to find the window that returns a large E value. Therefore, the term $\sum_{x,y}[I(x+u, y+v) - I(x,y)]^2$ is maximized and expanded using the Taylor serie, by rewriting the term with their respective derivatives:

$$E(u,v) = \sum_{x,y}[I(x,y) + uIx + vIy - I(x,y)]^2 \quad 4.10$$

Since Taylor series is infinite, all the terms after the third are ignored, giving an approximation. Further, the square is expanded into:

$$E(u,v) = [u\ v]M\begin{bmatrix} u \\ v \end{bmatrix} \quad 4.11$$

where $M = \sum w(x,y)\begin{bmatrix} {I_x}^2 & I_X I_Y \\ I_x I_y & {I_y}^2 \end{bmatrix}$.

The eigenvalues of the matrix determine the suitability of the window.

$$R = \det(M) - k(traceM)^2 \quad 4.12$$

$$\det M = \gamma_1 \gamma_2 \quad 4.13$$

$$traceM = \gamma_1 + \gamma_2 \quad 4.14$$

If R is greater than a threshold value then it is a corner. However, as it was referred previously, the method implemented in SIFT algorithm is similar to Harris Corner detection and doesn't calculate the eigenvalues but its ratio. The principal curvatures can be computed from a 2×2 Hessian matrix, H, in which the

derivative is estimated from the neighboring sample points differences. In order to eliminate the principal curvatures greater than a certain threshold the ratio of principal curvature need to be below that threshold, as written in expression 4.18.

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad 4.15$$

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta \quad 4.16$$

$$Det(H) = D_{xx}D_{yy} - \left(D_{xy}\right)^2 = \alpha\beta \quad 4.17$$

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \quad 4.18$$

With r as the ratio between the largest magnitude eigenvalue and the smaller one, so $\alpha = r\beta$. And so, both extrema images go through a contrast test and an edge test. As a result, the numbers of keypoints are reduced but the efficiency and robustness of the algorithm is increased.

At this stage, the objective is to assign an orientation to each keypoint by gathering the gradient directions and magnitude for all of the pixels around each keypoint [23]. Afterward, the most pronounced orientation is designated to that keypoint. It is also critical to ensure rotation invariance. The gradient of magnitude and orientations are calculated through the following mathematical expressions [23]:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1)^2} \quad 4.19$$

$$\theta(x,y) = (\tan^{-1}(\frac{(L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)})) \quad 4.20$$

This information is summarized in a histogram. Where the $360°$ of orientation are broken into 36 bins of $10°$ each. The gradient direction, at a certain pixel, is inserted into their respective degree bin according to its value, being the size that is add to each bin proportional to the magnitude of gradient at that pixel. Once all the pixels are calculated and assigned to their bin, a peak on the histogram can be detected. This peak determines the orientation of the keypoint, according to its bin.

Further, if any other peak is 80% of the highest peak, then it is converted into a new keypoint. This new keypoint has the same location and scale of the original, however its direction corresponds to this new peak, and so, one keypoint can be split into multiple keypoints according to the orientation. This will contribute to the stability in feature matching.

Finally, the last step in SIFT method is the creation of a descriptor for each keypoint. A numerical descriptor performs as a fingerprint of the pixel, it identifies a keypoint and distinguish it from other interest points [23]. A window of 16×16 pixels, around the keypoint, is split into sub-blocks of 4×4 pixel size. Afterwards, it is created a histogram of 8 bin. Each bin corresponds to 45 degrees and the amount added to the bin depends on the magnitude of the gradient and the distance from the keypoint, therefore gradients farther away from

the interest point contribute less in the amount added to the histogram's bin. These calculations are executed through a Gaussian weighting function that generates a gradient by multiplying it with the magnitude of the orientation. In sum, the descriptor is a feature vector of sixteen 4×4 regions assembled into 128 dimensional vectors, shown in Figure 22. Once there are 128 numbers, it is necessary to normalize them. Since the keypoint, typically, doesn't lie in a pixel but between pixels, in order to generate the orientation and magnitude data between the pixels the image needs to be interpolated.

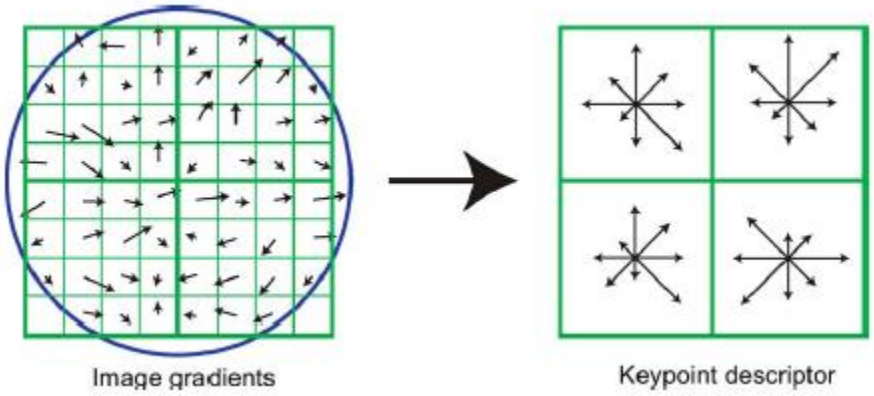


Figure 22: Keypoint Descriptor [23]

There are some downsides of this process since the feature vector is still rotation and illumination dependent. The descriptor uses gradient orientation, therefore if the image is rotated, the gradient orientations changes and the same is verified by varying illumination. Thus, rotation independence is achieved by subtracting the keypoint rotation from each orientation to end up with a gradient orientation relative to the keypoint orientation. Illumination independence can also be obtained by thresholding big numbers from the descriptor. Once these modifications are made, the feature vector requires to be normalized again.

4.2.2 Feature matching

According to Lowe in [23], the features between two images are matched by identifying its nearest neighbor in the database of keypoints. The definition of nearest neighbor is the keypoint with the minimum Euclidean distance. Some features won't have any correct match for not being detected in the previous step or due to noise, therefore, the most effective measure to match these features is to compare the distance of the closest neighbor to the second closest neighbor. The second closest match, sometimes, may be near to the first due to noise, therefore the ratio between the closest distance and the second closest distance is analyzed. If the ratio is greater than 0,8 these matches are rejected, eliminating 90% of false matches and discarding only 5% of correct matches [23].

The process to determine the best match between features from different images starts by defining a distance function that compares two descriptors and test all the features in an image to be able to find the

feature with minimum distances. The concept used in all software for comparison purpose was to apply an exhaustive search for the nearest neighbor to find the keypoint with minimum Euclidean distance, such as, the brute-force matcher [13]. The brute-force matcher can be described as a process where the interest point is matched with all the interest points of another image with a different perspective of the same object. The best match between the interest point of one image and all the keypoints from the other image is chosen according to the closest and smallest distance between the descriptors of the interest points. The most efficient nearest neighbor matching is by analyzing the ratio of distance between the best and the 2nd best match. If this ratio is low it indicates that the 1st nearest neighbor is close and the 2nd nearest neighbor is far, therefore, the 1st neighbor is a good match and it is kept, however if the ratio is very close to 0.8, implying the 2nd nearest neighbor is 80% as good as the 1st nearest neighbor then it is ambiguous and the match might be incorrect, leaving it out. This approach is computationally expensive, especially if the input is a large number of images, in this case this brute-force method can be replaced by a Cascade hashing method or an Approximate Nearest Neighbor (ANN). The approximate nearest neighbor method preprocesses the points into a data structure to assign the nearest points of all the points to a query point q and the distance between two points can be define in Euclidean distance. The cascade hashing [36] method uses hashing to convert all feature into binary code to conduct a fast speed operation.

The image matching strategies [92] compared in this study are the following: The exhaustive matching, the sequential and the vocabulary tree matching. The exhaustive matching compares each image with all images, matching all image pairs exhaustively. The sequential image matching depends on the location relationship between cameras in the world coordinates to efficiently match image in sequential order. Vocabulary tree matching matches every image according to its visual nearest neighbors, therefore, this method indexes the images ranked by similarity of its visual vocabulary.

RANSAC is an abbreviation for Random Sample Consensus. RANSAC is an iterative method that detects outliers and estimates a mathematical model with no outlier influence. This method is used to detect and discard outliers of matched features. RANSAC can remove false matches based on the epipolar geometry of the image pair. The majority of good samples allow a single model, however bad samples don't consistently agree with a single model, therefore, instead of detecting the bad inconsistent samples, the objective is to group samples that look the same and fit a model only to group the inliers.

This method randomly selects the minimal points to sample in order to provide a concept of inliers. RANSAC algorithm adopts the following structure: Random sampling, model building, thresholding and inlier counting. For a certain number of iterations, it starts by selecting randomly a subset of data, if the objective is to find a line, the data needs to be at least 2 points, if it is an affine transformation, 3 points, if it is an homography, it is 4 points. Afterwards the translation and rotation of the transformation needs to be known. The data is tested against a model and the inliers determined based on a threshold. If the distance between the data and the model is less than the threshold, then the data fits and the points are inliers. On the other hand, if the new model is better than the best model, by including more inliers, then the best model is replaced by

the new model, on the contrary, the best model is kept if maintains being better than the new model. These measures are repeated until the best model is found.

The number of samples is chosen to have at least one inlier in the data. The following mathematical expression corresponds to the selected number of samples:

$$N = \frac{\log(1-p)}{\log(1-(1-e)^s)} \qquad 4.21$$

where N is the number of samples, e is the probability that a point is an outlier, s is the number of points in a sample and p is the desired probability to choose a good sample. The number of points (s) depends on the problem that is being solved but in case of homography, the data corresponds to 4 points and the fundamental matrix depends on 8 points to be computed. The number of samples is usually chosen such that $p = 0{,}99$ in order to have at least one inlier.

The expression $(1-e)^s$ is the joint probability of the points being inliers of the s points, therefore the complement $(1-(1-e)^s)^N$ corresponds to the outliers points from N samples. And so, the probability that a point is outlier can be calculated by the following expression:

$$p = 1-(1-(1-e)^s)^N \qquad 4.22$$

4.2.3 Pose Location Estimation

This is the last step to build a sparse point cloud and consists in determining the camera parameters of each image, including intrinsic parameters and extrinsic parameters based on all of the information gathered until this stage. Besides, computing the camera parameters, this step also determines the 3D locations of the point by using the feature points matched and the camera parameters through a triangulation algorithm.

4.2.3.1 Camera motion estimation

Epipolar geometry [1] is the projective geometry between two views. Assuming a static scene with a minimum of 2 views and a set of point correspondences between both views with known intrinsic parameters of the camera of both views. Epipolar geometry aims to find the 3D points based on the parameters. It determines the extrinsic parameters, algebraically, and the 3D coordinates. Considering a point in 3D world as X, where its projection onto the first view results in a 2D coordinate x_1 and its projection onto the second view results in a 3D coordinate x_2. Epipolar geometry can describe the correspondences between the matched feature point of the image pair which are: The essential matrix and the fundamental matrix.

The essential matrix [1] describes the correspondences between the camera coordinates of the matched features and contains the extrinsic parameters. In mathematical terms, the essential matrix is the central

part of the epipolar constraint without the 2D coordinates (equation 4.25), written as the multiplication of translation, T_x, with rotation, R :

$$E = [t]_x R \qquad \in IR^{3\times3} \qquad 4.23$$

With the translation expressed by a 3×3 skew symmetric matrix:

$$[t]_x = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix} \qquad 4.24$$

The essential matrix was introduced by Longuet-Higgius and its properties were explained in detail by Huang and Fougeras [99], The essential matrix was chronologically introduced before the fundamental matrix, however, it is considered a specialization of the fundamental matrix for a normalized image coordinates case, referring to a variation of the problem where the camera calibration is known.

The essential matrix has 5 degrees of freedom. The rotation and translation have both 3 degrees of freedom and since the essential matrix is of homogeneous quantity, it has an overall scale ambiguity. Also a 3×3 matrix is an essential matrix if two of its singular values are equal and the third is zero. The relation between the image of a point in one camera to its image in the other camera, given the translation and rotation, can be written as [1]:

$$x_2{}^T E x_1 = 0 \qquad 4.25$$

The fundamental matrix [1] relates a point from the image plane from one camera to another and contains the intrinsic and extrinsic parameters. The fundamental matrix can be implemented in problems with uncalibrated cameras. The fundamental matrix was introduced by Longuet Higgins (1981) as an essential matrix, however, in 1992, Hartley and Fougeras identified it as the fundamental matrix.

The difference between the fundamental matrix and essential matrix lies on the type of coordinates. Although both relate corresponding points in two images, the fundamental matrix adapts the points in pixel coordinates and the essential matrix is computed based on points that are in “normalized image coordinates”. Therefore, to calculate the essential matrix it is mandatory to know or estimate the intrinsic parameters. So, the fundamental matrix relates the image coordinates of the 3D point and the essential matrix relates 3D coordinate of two images.

There are various methods to compute the camera parameters. The Five-point [39] algorithm is used to calculate the extrinsic parameters while the eight-point algorithm [40] estimates the intrinsic and extrinsic parameters. Usually, the Five-point algorithm is computed when the intrinsic parameter is one of the inputs on the 3D reconstruction process.

Normalized 8-point algorithm [40] is a method that solves the fundamental matrix, assuming it can only be solved if there are 8 point correspondences to compute the fundamental matrix. The objective is to compute the fundamental matrix such that $x_i{}' F x_i = 0$. Begins by normalizing the image since the points need to be

normalized to have the same image sizes. The images can be normalized by applying some transformation T. The centroid of each point and its range is determined, and the points are normalized between 0 and 1. The eigenvector corresponding to the smallest eigenvalue is calculated. Afterwards $\hat{F}$ can be constructed and normalized in order to apply the SVD method.

$$T = \begin{bmatrix} a_x & 0 & d_x \\ 0 & a_y & d_y \\ 0 & 0 & 1 \end{bmatrix} \quad 4.26$$

$$\hat{F} = \begin{bmatrix} \mathrm{f}_1 & \mathrm{f}_2 & \mathrm{f}_3 \\ \mathrm{f}_4 & \mathrm{f}_5 & \mathrm{f}_6 \\ \mathrm{f}_7 & \mathrm{f}_8 & \mathrm{f}_9 \end{bmatrix} \quad 4.27$$

The SVD method is a process of factorizing F in 3 matrices: the diagonal matrix, U matrix and V matrix. The diagonal matrix is formed by diagonal elements that correspond to the singular value of the inverse of eigenvalues. In order to force the matrix to have a rank deficiency, it is necessary to make a rank enforcement of $\sigma_3 = 0$ (equation 4.28). This step is fundamental because the Fundamental matrix that was obtained previously doesn't correspond to the real Fundamental matrix since it doesn't have a rank deficiency, and so, it is crucial to artificially make the third element of the diagonal matrix zero and then multiply it by U and V'. By de-normalizing the Fundamental matrix and applying the transformation T, this algorithm can be expressed as follows:

$$\check{F} = \mathrm{U} \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathrm{V}' \qquad (\sigma_1 \geq \sigma_2 \geq 0) \quad 4.28$$

$$F = T' \check{F} T \quad 4.29$$

Similarly to the 8-point algorithm, the five-point algorithm [39] considers the 5 epipolar constraints of each five-point correspondences and stacks the vectors of these 5 image points in a 5×9 matrix and the four vectors, $\hat{X}$, $\hat{Y}$, $\hat{Z}$, $\widehat{W}$, present in the essential matrices can be computed through SVD.

A Perspective n-point problem consists in analyzing the correspondences between the 3D coordinates of an object onto the 2D coordinates of an image, in order to estimate the pose of the camera in relation to the set of n 3D points. PnP problem was also introduced to estimate the position and orientation of the camera with respect to a scene or object. The most common methods that solve PnP problems are the P3P method and the EPnP method. Both methods are frequently developed in open source software.

A camera pose has 6 degrees of freedom: the rotation includes the roll, pitch and yaw and the translation with the 3 coordinates(x, y, z). Consequently, a problem without any features or correspondences$(n = 0)$ is a P0P has 6 degrees of freedom. A P1P corresponds to one feature detection alternative with 4 degrees of freedom. A P2P is a problem with 2 degrees of freedom, thus a P3P with 3 identified points has zero degrees of freedom.

The perspective n-point model for a camera can be written based on the intrinsic camera parameters, its translation and rotation, following the expression:

$$s\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}f_x & \gamma & u_0\\0 & f_y & v_0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}r_{11} & r_{12} & r_{13} & t_1\\r_{21} & r_{22} & r_{23} & t_2\\r_{31} & r_{32} & r_{33} & t_3\end{bmatrix}\begin{bmatrix}x\\y\\z\\1\end{bmatrix} \quad 4.30$$

where $[x \quad y \quad z \quad 1]^T$ is the homogeneous world point, $[u \quad v \quad 1]^T$ is the corresponding homogeneous image point, $\begin{bmatrix}f_x & \gamma & u_0\\0 & f_y & v_0\\0 & 0 & 1\end{bmatrix}$ corresponds to intrinsic camera parameters matrix, f_x and f_y are the focal length, γ is the skew factor for the image point, $\begin{bmatrix}r_{11} & r_{12} & r_{13}\\r_{21} & r_{22} & r_{23}\\r_{31} & r_{32} & r_{33}\end{bmatrix}$ is the 3D rotation and $[t_1 \quad t_2 \quad t_3]^T$ is the 3D translation matrix of extrinsic parameters. The PnP problem can return multiple solutions and usually require a post-processing algorithm, such as RANSAC to obtain a robust solution and avoid outliers.

P3P problem [37] is the less complex PnP problem and it is considered a special case of a PnP problem. P3P problems have 3 points correspondences known (n=3) which makes it able to generate 4 possible solutions. Geometrically the problem is represented as it is in Figure 23:

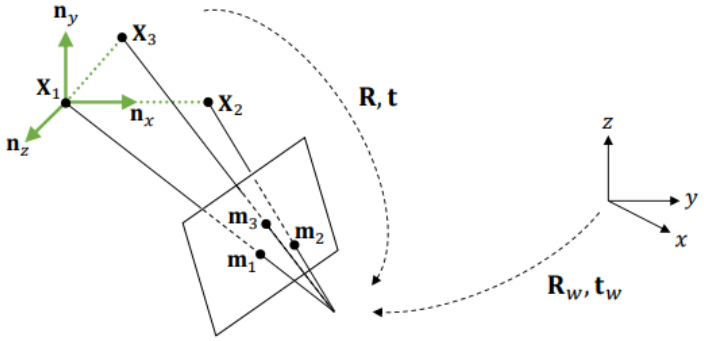


Figure 23: The Perspective-three point problem

In order to determine where the triangle is located, it is necessary to solve polynomial equations. Generally, there are 8 solutions for the system of equations, formed by 4 solutions behind the focal point and 4 in front of the focal point. Therefore, only 4 solutions are in front of the camera. In case $n \geq 3$, there are still some redundancies since it returns solutions as the n gets higher and not coplanar.

There are a substantially amount of literature that estimates the camera pose based on 3 points. This method can be classified as iterative or non-iterative and with known or unknown camera parameters. The P3P problem returns 4 solutions, however these solutions were estimated based on 3 points that can induce inaccurate estimations. Assuming P is the center of the projection of the camera, A, B and C are the 3D world points with corresponding image points u, v and w. Being $X = |PA|$, $Y = |PB|$, $Z = |PC|$, $\alpha =< BPC$, $\beta =< APC$, $\gamma =< APB$, $P = 2cos\alpha$, $q = 2cos\beta$,$r = 2cos\gamma$, $a' = |AB|$, $b' = |BC|$, $c' = |AC|$. Forming triangles: PBC; PAC; PAB. The polynomial equation to solve is the following:

$$\begin{cases} Y^2 + Z^2 - YZp - b'^2 = 0 \\ Z^2 + X^2 - XZp - c'^2 = 0 \\ X^2 + Y^2 - XYr - a'^2 = 0 \end{cases} \quad 4.31$$

Efficient Perspective n-points is a method developed by Lepetit et al. to also solve PnP problems with $n \geq 4$ [100]. This approach expresses the n 3D points as a weighted of four virtual control points to determine the camera pose in an efficient way, assuming the 3D to 2D coordinates correspondences and camera parameters are known. The problem can be written as follows:

$$p_i{}^w = \sum_{j=1}^{4} \alpha_{ij} c_j{}^w \quad 4.32$$

where $p_i{}^w = [X^W, Y^W, Z^W]^T$ is a 3D point in world coordinate system, α_{ij} is the homogeneous barycentric coordinates and $c_j{}^w = [X^W, Y^W, Z^W]^T$ is a 3D control point in world coordinates. The previous equation can also be written in terms of camera coordinate system.

The control points can be chosen by taking the centroid of the selected points and consider it as a control point. The rest is chosen in order to be aligned with the principal directions of the data. Theoretically these control points could be chosen arbitrarily, however, this approach leads to a more stable method since it equates into a linear system of equations. The point coordinates needs to be normalized and the solution is the weighted sum of eigenvectors as follows [100]:

$$w_i \begin{bmatrix} u_i \\ 1 \end{bmatrix} = A p_i{}^c = A \sum_{j=1}^{4} a_{ij} c_j{}^c \quad 4.33$$

where A is the internal camera calibration matrix, w_i is the scalar projective parameters and $[u_i]_i = 1, .. n$ is the 2D projections of the reference points. The previous expression can be expanded and rewritten as follows:

$$w_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \sum_{j=1}^{4} a_{ij} \begin{bmatrix} x_j{}^c \\ y_j{}^c \\ z_j{}^c \end{bmatrix} \quad 4.34$$

where $[u_i \quad v_i]^T$ are the 2D projections of u_i, f_u, f_v are the focal length coefficients, (u_c, v_c) is the principal point and $[x_j{}^c \quad y_j{}^c \quad z_j{}^c]^T$ are the 3D coordinates of each control point. Considering there are 4 central points, there is a total of 12 unknown control point coordinates that can be generated by solving this linear system:

$$\sum_{i=1}^{4} \alpha_{ij} f_u x_i{}^c + \alpha_{ij} (u_c - u_i) z_j{}^c = 0 \quad 4.35$$

This linear expression can also be written as $Mx = 0$. Where M is a $2n \times 12$ matrix with all known coefficients and $x = [c_1{}^c \quad c_2{}^c \quad c_3{}^c \quad c_4{}^c]^T$ is vector of the 12 unknowns. The solution of this linear system lies on the null space, or kernel of M in which v_i are the N null singular values of M that correspond to the column of

the right singular vector of M. However, M is transformed into a small constant matrix $M^T M$ of 12×12 size in order to compute the eigenvectors. Although this step is the most computationally complex and the most time consuming, it is where EPnP efficiency lies.

$$N = 1: \quad x = \beta_1 v_1 \tag{4.36}$$

$$N = 2: \quad x = \beta_1 v_1 + \beta_2 v_2 \qquad (\dots) \tag{4.37}$$

This approach assumes that the distance between control points in the camera coordinate system equate to the ones computed in the world coordinate system, such as in expression 4.38:

$$\left\| {c_i}^c - {c_j}^c \right\|^2 = \left\| {c_i}^w - {c_j}^w \right\|^2 \tag{4.38}$$

These geometric constraints allow to correctly find β_i, assuming expression, for example like for $N = 1$. Since the distance $\left\| {c_i}^w - {c_j}^w \right\|^2$ is known, β can be computed as follows:

$$\beta = \frac{\sum [i,j] \in [1:4] \left\| v^{[i]} - v^{[j]} \right\| . \left\| {c_i}^w - {c_j}^w \right\|}{\sum [i,j] \in [1:4] \| v^{[i]} - v^{[j]} \|^2} \tag{4.39}$$

This solution has $0(n)$ complexity and it can be applied to planar and non-planar control points. The most efficient optimization is the Gauss-Newton algorithm to refine the coefficients to compute the rotation and translation matrix. This process needs to be inverted by first computing the control points coordinates in the camera frame reference and the coordinate of the 3D points in camera frame reference to extract the R matrix and T vector.

There is another well-known PnP method introduced by A.Penõte et al's, the UPnP [101], which is an uncalibrated PnP method. As it was suggested by its name, UPnP is a PnP method that estimates the camera pose and focal length without calibration. Similar to EPnP, the unknown parameters of this problem are the weights of the linear combination of the eigenvectors from a set of n 3D to 2D point correspondences. In contrast to EPnP, UPnP has polynomial of 4 degrees instead of 3 degrees.

It is assumed a camera with square pixel size and with the principal point (u_0, v_0) at the center of the image. Furthermore, it is also considered a set of 3D to 2D correspondences of n points ${p_1}^w, \dots, {p_n}^w$ in world coordinate system, w and their 2D projections $u_i, \dots, u_n$ in the image plane. These considerations allow the problem to be formulated in order to return the focal length f of the camera, the rotation matrix R and the translation vector t. This problem is solved by first minimizing an objective function based on the reprojection error:

$$min_{f,R,t} \sum_{i=1}^{n} \| u_i - \tilde{u}_i \|^2 \tag{4.40}$$

where $\tilde{u}_i$ is the projection of ${p_i}^w$. Identically to EPnP method, the 3D points are the rewritten in terms of the barycentric coordinates of the 4 control points. At this stage, the problems is expressed as a linear system of 2n equations of 12 unknown parameters. However, in UPnP method, the perspective projection equation

differs from the EPnP approach, since the focal length needs to be considered. The resulting linear system can also be represented by the following linear system: $Mx = 0$. Where M is a $2n \times 12$ Matrix computed from the principal point, the 2D points u_i and the coefficients α_{ij}.x is a vector formed by the 12 unknowns, which contain the control points 3D coordinates with respect to the camera frame and the focal length dividing the x term.

$$x = \left[{x_1}^c, {y_1}^c, {z_1}^c/_f, \dots, {x_4}^c, {y_4}^c, {z_4}^c/_f\right]^T \quad 4.41$$

The solution lies on the null space, or Kernel of M. Therefore, the singular value decomposition (SVD) method is the approach to solve this problem.

$x = \sum_{k=1}^{N} \beta_k v_k$ (weighted sum of the null eigenvectors v_k of $M^T M$

Where v_k is the N null eigenvalue of M, in which N is the rank of the kernel of $M^T M$ and β_k is the unknown weights. Similar to EPnP method, before calculating the eigenvectors M is transformed into a small constant matrix $M^T M$ of size 2×12. The value of β_i is computed by introducing the same distance constraints as to the prior method:

$$\left\|{c_j}^c - {c_j}^c\right\|^2 = {d_{ij}}^2 \quad 4.42$$

Where c_i and c_j are a pair of control points and d_{ij} is the known Euclidean distance between the control points in world coordinate system. This expression can also be written in terms of β_k coefficients as follows:

$${c_j}^c = \begin{bmatrix} {x_j}^c \\ {y_j}^c \\ {z_j}^c \end{bmatrix} = \sum_{k=1}^{N} \begin{bmatrix} \beta_k {v_{k,x}}^{[j]} \\ \beta_k {v_{k,y}}^{[j]} \\ \beta_k {v_{k,z}}^{[j]} \end{bmatrix} \quad 4.43$$

This polynomial system is solved through a closed form of linearization techniques resulting in the combination of the equation (1) and the 6 distance constraints of equation (2). In case $N = 1$ (2 variables need to be estimated) a standard linearization method can be executed, however for $N = 2$ the number of equations remain the same while the unknown variables increase, therefore an exhaustive linearization approach can be an effective method to solve the respective problem. In order to exemplify the linearization approach, the case of $N = 1$ is applied in the following formulation of the method:

$$Lb = d \quad 4.44$$

Where $b = [\beta_{11} \quad \beta_{ff11}]^T = [{\beta_1}^2 \quad f^2 {\beta_1}^2]^T$, and L is a 6×2 matrix formed by the known elements of the first eigenvector column v_1, and d is a 6 vector of squared distance between the points. The least squared method is applied to estimate the magnitudes of β_1 and f_1 by substituting:

$$\beta_1 = \sqrt{\beta_{11}} \qquad f = \sqrt{\frac{|\beta_{ff11}|}{|\beta_1|}} \tag{4.45}$$

Once β is known, the camera pose can be estimated. The control points coordinates are first computed in the camera frame reference and then the 3D points coordinate in camera frame reference are computed to extract the rotation matrix R and the translation vector t.

4.2.3.2 Triangulation

Once the camera parameters are known it is possible to compute the 3D locations of the matched points by utilizing the feature points with respective correspondences and the camera parameters and positions of each image. The DLT [1] is an algorithm that formulates a homogeneous linear system and solves a set of variables, carrying similarity relations to solve triangulation problems. This method can be applied to solve simple linear triangulation problems. However, before applying the DLT algorithm, the input data needs to be normalized to ensure that the algorithm is invariant to arbitrary noise in scale and coordinate frame, therefore, the image space coordinates are translated to their centroid to be located at the origin and then scaled so that the average distance to the origin is $\sqrt{3}$ and the average point is equal to $(1,1,1,1)$ [102]. Afterwards, this problem can be solved by assuming a stereo rig problem in which a set of points are visible to more than one camera, each input image has a measurement x in the 2D camera coordinates of a world point and a second measurement x' which corresponds to the same point in a second camera coordinates, shown in the expressions 4.46 and 4.47

$$x = PX \tag{4.46}$$

$$x^{'} = P'X \tag{4.47}$$

where x represents the 3D world coordinate, P and P' are the projection of the 2D camera coordinates into 3D world coordinates. These equations can be combined in the form of $AX = 0$, as in expression 4.48.

$$PX = \begin{bmatrix} p^T X \\ p^{2T} X \\ p^{3T} X \end{bmatrix} \Rightarrow A = \begin{bmatrix} xp^{3T} - p^T X \\ xp^{3T} - p^{2T} \\ x'p'^{3T} - p'^T X \\ x'p'^{3T} - p'^{2T} \end{bmatrix} \tag{4.48}$$

The method singular value decomposition or SVD solves A in order to have all 3D coordinates of the known point from X.

4.2.3.3 Model parameters correction

Before generating the sparse cloud of the scene, most 3D reconstruction pipelines correct the camera parameters of each image and the 3D points obtained in the previous step through the Bundle adjustment method or its variation. Bundle Block adjustment or Bundle adjustment was originated in the

photogrammetry field with the objective of minimizing the reprojection error between the location of the image measured points and the predicted image points. This approach focuses in optimizing the camera pose, the intrinsic calibration and the location of the 3D point to obtain an optimal 3D reconstruction. This optimization problem is solved based on a non-linear least-squares algorithm.

The image projection is usually corrupted by noise due to the measurement in the data. Therefore, the reprojection error is the difference between the image projection and the image measurements:

$$e = \tilde{m} - m = \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \qquad 4.49$$

Outliers can be identified with at least 4 views per point, however, with 5 to 6 observations per point can yield a good estimation of the gross errors.

Bundle adjustment focus in minimizing the error measured in the image coordinate system. Since m is derived from the homogeneous projection of the point X into the image through the camera projection system to obtain measurements in terms of the homogeneous coordinates of that point. Their coordinates are the back projection of the 3D point divided by w. In general, this method adjusts the rotation of the camera, the camera center and the 3D point locations:

$$e = \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} - \begin{bmatrix} x/w \\ y/w \end{bmatrix}, \qquad where \begin{bmatrix} u \\ v \\ w \end{bmatrix} = P \begin{bmatrix} X \\ 1 \end{bmatrix} = KR[I_{3\times3} - C] \begin{bmatrix} X \\ 1 \end{bmatrix} \qquad 4.50$$

The objective is to minimize the projection error by minimizing the measurements in the image $[\tilde{x} \quad \tilde{y}]^T$, the rotation and the camera center with respect to the three-dimensional vector:

$$minimize \left\| \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} - \begin{bmatrix} u(R,C,X)/w(R,C,X) \\ v(R,C,X)/w(R,C,X) \end{bmatrix} \right\|^2 \qquad 4.51$$

The error function is a non-linear least square problem. The rotation matrix can be represented in different forms, but the most common among software is the quaternion representation.

$$minimize_{q,C,X} \left\| \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} - \begin{bmatrix} u(R(q),C,X)/w(R(q),C,X) \\ v(R(q),C,X)/w(R(q),C,X) \end{bmatrix} \right\|^2 = minimize \|b - f(R(q),C,X\|^2 \qquad 4.52$$

Where q is the quaternion vector.

The solution of this nonlinear square problem consists in 2 steps: Primarily, the error function is extracted and its derivative function is computed with respect to the variable X and set it to zero, in order to minimize this nonlinear least square problem because the gradient is zero at a local maxima. The second step revolves in computing the Taylor expansion of the nonlinear function $f(x)$ to identify the direction and distance. The present direction is calculated by computing Δx.

$$\frac{dE}{dx} = 2\frac{df(x)}{dx}^T f(x) - 2\frac{df(x)}{dx}^T b = 0 \qquad 4.53$$

$$f(x+\Delta x) \approx f(x) + \frac{df(x)}{dx}\Delta x, \qquad where\ \Delta x = (J^T J)^{-1} J^T (b - f(x)) \tag{4.54}$$

The number of rows of the Jacobian matrix corresponds to the number of the constraints and the number of columns is associated to the number of variables that are searched. The normal equation retrieves an iterative solution to find the local minima of the respective nonlinear function.

$$f(R(q), C, X) = \begin{bmatrix} u/w \\ v/w \end{bmatrix}, \qquad where \begin{bmatrix} u \\ v \\ w \end{bmatrix} = KR[X - C] = \tag{4.55}$$

$$= \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} [X - C]$$

The Jacobian is derived for this system of projection equations to compute Δx from the normal equation. The u, v, w are homogeneous coordinates in the image space, which consists in the rotation matrix elements r_{ij}, a known vector X and a known camera center C in the following form:

$$\begin{cases} u = [fr_{11} + p_x r_{31} \quad fr_{12} + p_x r_{32} \quad fr_{13} + p_x r_{33}][X - C] \\ v = [fr_{21} + p_y r_{21} \quad fr_{22} + p_y r_{32} \quad fr_{23} + p_y r_{33}][X - C] \\ w = [fr_{31} \quad r_{32} \quad r_{33}][X - C] \end{cases} \tag{4.56}$$

The Jacobian can be divided into 3 columns according to the unknown or known variables. The 3 groups are established, first through the quaternion for the rotation matrix which specifies how the cameras are oriented formed by 4 dimensional vectors and so 4 columns. Secondly, the camera center in $[x, y, z]$ consists in 3 columns of known variables. Thirdly, the known position for the 3D point coordinate X results in a 3 column in the Jacobian. Overall, the Jacobian matrix has a total of 10 columns and 2 rows, since there are 2 constraints for each point projected to the image. The symbolic representation of the derivative functions is the following:

$$J = \begin{bmatrix} \frac{df(R(q), C, X)}{dq} & \frac{df(R(q), C, X)}{dC} & \frac{df(R(q), C, X)}{dX} \end{bmatrix} \tag{4.57}$$

It is also possible to decompose the derivative function in respect to the quaternion by two parts: The derivative function with respect to the rotation matrix formed by a 9×4 matrix. In the case of multiple views of the same dimensional objects, the non linear least square problem is updated into a different Jacob matrix where it takes into consideration all views. The following expression is an example of the Jacobian matrix if there were 2 views:

$$J = \begin{bmatrix} 1^{st}\ view\ Jacobian & 0_{2\times 7} & 3D\ point \\ 0_{2\times 7} & 2º\ view\ Jacobian & 3D\ point \end{bmatrix} \tag{4.58}$$

Where the 1^{st} view and 2º view jacobian are the camera orientation and camera center, respectively, with respect to the 3D point and because the 1^{st} view is affected by the 2º view motion there is a part of the Jacobian which corresponds to the 2º view but in the 1^{st} view the rows are zero and vice-versa.

A very established library is Ceres *Solver* [103]. Ceres Solver is an open source C^{++} library for modeling and solving large optimization problems such as a nonlinear least squares problems. This library consists in two different parts, the modeling API which formulates an optimization problem and a solver API which controls the minimization algorithm. The modeling API formulates the non-linear least squares problem as follows:

$$min\frac{1}{2}\sum_{i}\rho_i(\|f_i(x_i,\dots,x_{ik})\|^2) \qquad s.t.\, l_j \le x_j \le u_j \tag{4.59}$$

where $\rho_i(\|f_i(x_i,\dots,x_{ik})\|^2)$ is a residual block, f_i is a cost function that depends on the parameter block $\{x_i,\dots,x_{ik}\}$. The parameter block consists in small groups of scalar together, such as, the components of the quaternion and the components of the translation vector that define the pose of the camera. ρ_i is a loss function which is a scalar function that reduces the influences of outliers on the result of the optimization problem. If the non-linear, least square problem is unconstrained then $\rho_i(x) = x$, $l_j = -\infty$ and $u_j = \infty$, resulting in the expression $\frac{1}{2}\sum_i \rho_i(\|f_i(x_i,\dots,x_{ik})\|^2)$. A cost function is computed to determine the vector of residues and Jacobian matrices for each objective function. The cost function is usually formulated as follows:

$$J_i = \frac{d}{dx_i} f(x_i,\dots,x_k) \qquad \forall\, i \in \{1,\dots,k\} \tag{4.60}$$

where $f(x_i,\dots,x_k)$ is the function that depends on the parameter blocks $[x_i,\dots,x_k]$.

The second part of Ceres solver is the solver of non-linear least square problems. Assuming there is an optimization problem such as:

$$argmin\frac{1}{2}\|F(x)\|^2 \qquad l \le x \le U \tag{4.61}$$

where $x \in R^n$ and n is the n-dimensional vector of variables, and $F(x) = [f_i(x),\dots,f_m(x)]^T$ be a m-dimensional function of x. l and U are the lower and upper bound to the parameter x.

Since $J(x)$ is the Jacobian of $F(x)$, where $J_{ij}(x) = djf_i(x)$ and the gradient vector is $g(x) = \nabla\frac{1}{2}\|F(x)\|^2 = J(x)^T F(x)$. For non-linear least squares, an approximation is the linearization of $F(x+\Delta x) \approx F(x) + J(x)\Delta x$, leading to a linear least square problem:

$$min\frac{1}{2}\|J(x)\Delta x + F(x)\|^2(2) \tag{4.62}$$

The approximation needs to be computed for each iteration, in order to determine a correction Δx to the vector x. Depending on the size of step Δx, which controls the convergence of the algorithm, the non-linear optimization algorithms can be divided into two categories: The trust region and the line search. The trust region approach lies in the objective function approximation through a model function of trust region which is a known space. The model function minimizes the objective function of the trust region, if not, it's

contracted and the optimization problem needs to be solved again. The line search approach computes the descent direction through the gradient descent method, Newton's method or the Quasi-Newton method and then computes a step size in which is decided the distance it should move.

Ceres Solver implements multiple algorithms in both approaches. For the trust region it implements algorithms such as: The Levenberg-Marquardt method, the Dogleg method and inner iterations. The line search can only solve unconstrained problems supporting a backtracking and interpolation Wolfe condition live search algorithm.

The trust method algorithm has the following procedures: First it is given an initial point x and a trust region radius μ. Then it is solved the linear least square problem such that $\|D(x)\Delta x\|^2 \leq \mu$ and $L \leq x + \Delta x \leq u$. Afterwards, it is calculated a variable that measures the quality of the step Δx.

$$\rho = \frac{\|F(x + \Delta x)\|^2 - \|F(x)\|^2}{\|J(x)\Delta x + F(x)\|^2 - \|F(x)\|^2} \qquad 4.63$$

The variable ρ determine if the linear model accurately predicted the decrease of the non-linear, however, if $\rho \geq \eta_1$ then$\mu = 2\mu$ or $\rho \geq \eta_2$ then $\mu = 0{,}5\mu$. This process repeats itself, restarting by solving the linear square problem again and checking ρ value for bigger and smaller radius.

Nevertheless, the main computational step is to solve the linear square problem, through the expression 4.63. There are different ways in solving it, in which the most common is the Levenberg-Marquart method.

The Line search method in Ceres Solver follows the forward procedure: First it is given an initial point x. Afterwards, it is calculated the direction:

$$\Delta x = -H^{-1}(x)g(x) \qquad 4.64$$

where H(x) is the hessian of the objective function and $g(x)$ is the gradient at x. it is calculated the expression 4.64 in order to compute $x = x + \mu\Delta x$. Where μ is the trust region radius and D(x) is a matrix defining the matrix on F(x) domain.

Another approach is the Multicore Bundle Adjustment [104]. This method consists in applying CPU and GPU parallelism to achieve a faster bundle adjustment method and solve larger problems. This approach implements the Lavenberg- Marquardt algorithm without storing any matrices (Hessian, Jacobian or Shur complement) in memory and therefore saving space and time.

A multicore system has some issues such as the mismatch between the processor speed and rate at which the data is retrieved from the memory. Also, multiple threads may be written to the same memory location in multithreaded systems, thus, to apply an optimal multicore bundle adjustment there are certain requisitions: The maximization of the processor occupancy and the optimization of the memory access to reduce contention [104].

The Jacobian matrices are stored in a Block compressed sparse row for both CPU and GPU, when needed. Focusing in CPU parallelization, it is crucial to notice that Multicore CPUs can run 10s of threads, simultaneously [104]. In terms of CPU, the computational tasks are divided into several threads that are applicable for the number of CPU cores available to efficiently use all of the computation power and therefore maximize the processor occupancy. In order to optimize the memory access, the fundamental indexing structures are stored to allow the threads to run independently [104]. Also, shuffled copies of data are stored in order to the most frequently used function can have memory access patterns.

GPUs have a much smaller access to RAM as compared to CPUs, therefore the application of the implicit-hessian, the implicit Schur and the matrix-free version with Jacobian in order to get rid of the storage of the augmented Hessian, the schur complement matrices and the Jacobian and thus optimize memory access pattern.

Once the model parameters are corrected, the sparse point cloud can be built, and the 3D reconstruction process can proceed to a Dense reconstruction process.

4.3 Dense Point Cloud Reconstruction

Dense point cloud is the representation of depth maps. Depth maps are images represented in gray pixels of intensity between 0 to 255 value, in which, 0 corresponds to the black pixels and the farthest away from the source that took the images and 255 corresponds to the white pixels that are near the sensor. The Dense Cloud reconstruction is implemented though Multi-view stereo approaches such as CMVS/PMVS and SGM. CMVS/PMVS can be divided into two steps: The Clustering Views for Multi-View Stereo (CMVS) and the Patch-based Multi-View Stereo (PMVS). The CMVS is adopted when the 3D points and the camera pose are known to group similar image with the intention of optimizing the MVS process, afterwards the PMVS can be computed to generate the 3D model (Figure 24).

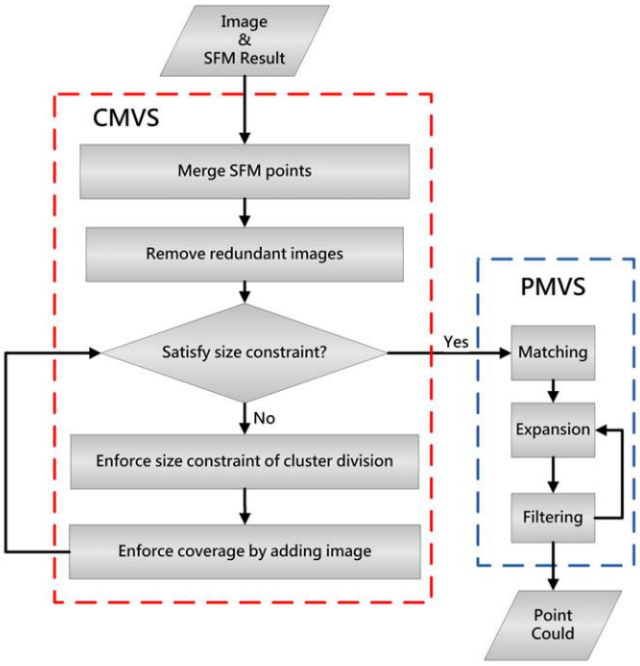


Figure 24: Flowchart of CMVS and PMVS [54]

CMVS method [105] aims to decompose the set of input images into clusters that have small overlap. Once the images have been processed by SFM algorithms for the camera poses and the 3D points to be known, CMVS algorithm finds overlapping image clusters with manageable size in order for each point to be reconstructed by at least one of the clusters, as it is shown in Figure 25. CMVS is suitable for large input data.

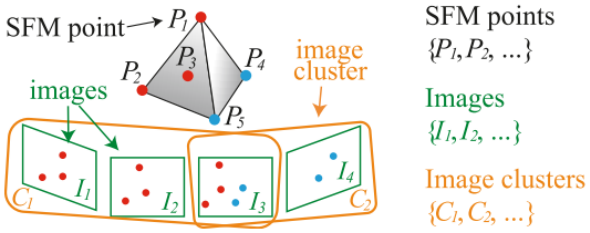


Figure 25: Clustering algorithm [105].

The CMVS algorithm [105] is divided into four main steps, in which the first two stages are pre-processing and the last two are iterative loops. The first step is a SFM filter that aggregates visible data over a local neighborhood and merges the point in that neighborhood. The position of the merged points is the average of its neighbors. The point and its neighbor are removed from the input set. This process is repeated until the input set is empty to reduce the number of points and improve the running time. The second pre-processing step is the image selection in which is performed a removal test for all images in increasing order of image resolution in order to remove low resolution images first. This removal test works according a coverage constraint to discard image and speed up the following steps. The third step is the cluster division

that divides an image cluster into smaller components by enforcing a size constraint to split the cluster. These steps are repeated until the size constraint is satisfied for all clusters. The final step is the image addition in which images are added to the process and this action is evaluated according its effectiveness of increasing coverage.

The PMVS method [56] can be decomposed in three steps: matching, expansion and filtering. The matching step builds sparse patches from the detected features, through the Difference of Gaussian and Harris operators. The expansion step ensures that the patches are dense enough for reconstruction by reconstructing at least one patch in every image cell by taking the existing patches and generating new ones, and so the patches are expanded. The filtering step is used to remove errors from the patches. There are three filters: A filter that considers a patch outlier if 2 patches are not neighbors and are stored in the same cell. The second filter tests if a patch is visible to a depth map to filters out the outlier if the number is less than a certain threshold. The third filter considers an outlier patches whose proportion is lower than 0.25.

SGM or Semi-global matching is a stereo method that supports pixelwise stereo matching based on a global cost function that expresses a smoothness constraint [57]. This process computes the similarity between each pixel in stereo image and each pixel of another stereo image to get the disparity maps. SGM performs a line optimization in different direction, the bigger the number of directions corresponds to a better quality, on the other hand, the less number the faster is the process but less quality, while it is recommended 16 paths for a good coverage, the minimum is 8. The objective of SGM is to minimize the approximation of the global cost function shown in [57].

4.4 Surface Reconstruction

Surface reconstruction states a problem of conversion, initiating with a dense point cloud set as an input to a surface. This process recovers the topology and geometry of the object or scene that is being reconstructed. An important property to consider is to have a watertight surface reconstruction with an adjustment between fitted data and smoothness of the surface to avoid noise or outliers. This property creates assumptions to choose the best candidate shape for the reconstruction problem. There are two main methods applied to reconstruct the surface of the Limestone block: The Poisson Surface Reconstruction and the Delaunay Triangulation.

The Poisson Surface reconstruction [74] is a method that transforms oriented points into a spatial Poisson problem. In order to obtain the reconstructed surface, it is computed a 3D indicator function X which attributes 1 to the points inside the model and 0 to the points outside to obtain an isosurface, shown in Figure 26. The surface can be detected based on the gradient of the indicator function since this implicit function is constant everywhere excepting at points near the surface. The standard Poisson reconstruction

computes the scalar function X whose gradient diverges to a vector field $\vec{V}$, determined by the samples. Thus, the indicator function needs to be computed from the samples: $\Delta X = \nabla . \nabla_x = \nabla . \vec{V}$

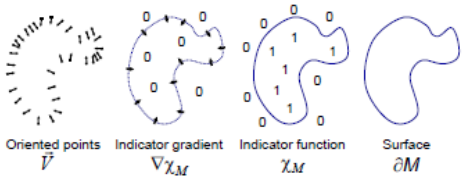


Figure 26:Illustration of Poisson Reconstruction in 2D [74]

The gradient field is defined through the convolution between the indicator function X with the smoothing filter to compute the smoothed function. This approach is essential to avoid computing unbounded values at the surface boundary in a vector field. Knowing that a vector field can be obtained by smoothing the surface normal field. Afterwards the gradient field is estimated from the approximation of the surface integral with a discrete summation [74]:

$$\nabla(X_M * F)(q) = \sum_{s \in S} \int \tilde{F}_p(q) \tilde{N}_{dM}(p) dp \approx \sum_{s \in S} |P_s| \tilde{F}_{s.p}(q)\, s.\vec{N} \equiv \vec{V}(q) \qquad 4.65$$

Where M is a solid Model with boundary dM, X_M is the indicator function of M, $\tilde{N}_{dM}(p)$ denotes the inward surface normal to $p \in dM$. $\tilde{F}_p(q)$ is the smoothing filter for which $F_p(q) = \vec{F}(q - p)$ is the translation to the point p. S is the set of point to divide dM into various patches P_S.

The smoothing filter F needs to satisfy two conditions: A narrow filter to not oversmoothed the data and a wide enough in order to approximate well the patches P_s. Therefore, it is applied a Gaussian filter. Furthermore, once the vector field V is known, the Poisson equation is solved to find the best least-square approximate solution. The goal is to solve the scalar function which can be obtained by solving the Poisson equation:

$$\nabla X = \Delta . \vec{V} \qquad 4.66$$

To sum up, the watertight mesh can be achieved by transforming the point samples into a vector field in 3D, in order to reach to a scalar function whose gradient fits the vector field and to extract the isosurface.

Screened Poisson surface reconstruction [75] is a technique similar to the Poisson surface reconstruction but this approach uses a sparse set of points to be incorporated as interpolation constraint, generalizing it to a screened Poisson equation. Screened Poisson surface reconstruction forces the reconstructed isosurface to pass through the input points. Poisson surface reconstruction technique is known for its tendency to over-smooth the data compared to the Screened Poisson surface reconstruction that preserves better the details of the scene [75].

Delaunay triangulation [63] is the Dual data structure of the Voronoi Diagram. The Delaunay triangulation subdivides the sample convex Hull, forming simplices on all faces. In contrast, the Voronoi diagrams associate each sample point to a subdivision of space, called convex cell. Delaunay triangulation is a triangulation of a convex hull, a set of points P, maximizing the minimum angle of all the angles from the triangles created by the triangulation whose points cannot be inside of the circumcircle of any triangle. Hence, if there are two Voronoi cells such that share a Voronoi edge, a nonempty intersection, then the points are connected. The Voronoi vertices are the center of the circumcircles created once there are 2 triangles in which the main property is that the circumcircles don't contain any of the rest of the vertices of the triangulation and therefore, they are empty. Both representations can be seen in Figure 27.

The Delaunay triangles can be reconstructed if the sample is dense enough. The definition of a dense enough sample can vary with 3 characteristics: The Medial axis, the local feature size and the epsilon-sampling. A Voronoi diagram is the set of points with the nearest neighbors in P, forming edges if the set is built from two nearest neighbors and vertices with three or more nearest neighbors.

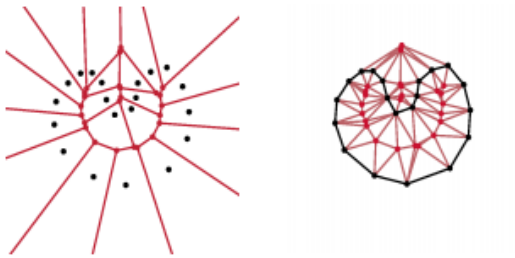


Figure 27: The Voronoi Diagram and the Delaunay triangulation [66]

In sum, the Delaunay triangulation is a set of triangles made from a discrete set of vertices so that no vertex lies inside the circumcircle of any triangle. Delaunay triangulation can be computed through the Bowyer-Watson algorithm which first creates a super-triangle that includes all points, and then a point is selected. If this point is inside a circumcircle formed by a triangle then it can't be a Delaunay triangle and the edges of the former triangle are connected to the point to make a new set of triangles. This process is repeated for all points. In the end the vertices of the super triangles are deleted, and a triangular mesh is created to form the surface of the object or scene.

4.5 Textured Reconstruction

Textured reconstruction [95], as the last step of 3D reconstruction, offers color information mapped onto the geometry. This step generates texture parametrization to input in the 3D surface model, as illustrated in Figure 28. The color information is obtained from the calibrated cameras to generate a new texture mapping through detailed color encoding "per-vertex". The texture mapping consists in a parametrization that associates a pixel in a 2D image to each point on the 3D surface and the color is stored in an image as

texture map. This is a standard solution for texturing surfaces that requires the 3D geometry to have a correspondence between a point on the 3D mesh and a point on the 2D image.

Figure 28: Example of color mapping [95].

5 Results and Discussion

For each output stage of the reconstruction the best preprocessed set of images is introduced on each software to further analyze the models. The results are divided in 4 stages already mentioned in problem formulation: The first stage compares different software, the second stage determines the best algorithm combination within the previously concluded best software, the third stage achieves a 3D model with known intrinsic parameters for comparison as well as the fourth stage that accomplish a 3D model with known intrinsic and extrinsic parameters This chapter presents the results for comparison and further discussions.

5.1 Stage 1

This first stage compares the different types of software to conclude the most suitable pipeline in order to reconstruct 3D models of limestones. For all open source software, it was used similar procedures with the objective of comparing each pipeline The four output stages considered were: Sparse point Reconstruction, Dense point Reconstruction, Surface Reconstruction and Textured Reconstruction.

5.1.1 Sparse point Cloud

This step is the most complex phase of 3D reconstruction and corresponds to the Structure from Motion phase. The sparse point cloud is the output of the Structure from motion process by representing the 3D position of the tie points. This point cloud can be generated by *VisualSFM*, *COLMAP*, *Meshroom* and *Agisoft Metashape*. Any of these pipelines receives the preprocessed images, respectively, the original images in which the gamma was corrected by 0.4 and a histogram equalization was implemented, as input to generate a sparse cloud.

Based on the sparse point cloud generated from *VisualSFM*, it was concluded that this software wasn't viable to generate a 3D model of the quarry stones scene due to miscalculation of the camera poses. As it is shown in Figure 49, the sparse cloud has no sparse points on the left side of the quarry stone line, although, there were features detected in all 68 images and all features were matched between their correspondent image features, the pipeline estimated only 34 camera poses on the right side of the scene, instead of 34 cameras on the left and 34 on the right side. Since the estimation and refinement of the camera poses, through PnP and Multicore bundle adjustment with a robust estimation based on RANSAC, produced incorrect extrinsic parameters of the camera, the triangulation was also incorrect, returning only sparse points on the right part of the scene and unreliably positioned. Nevertheless, this pipeline was able to detect a similar amount of features for each images, according to Figure 45.

Contrarily to *VisualSFM*, *COLMAP* is able to generate a sparse point cloud with similar geometry to the actual scene, as it is displayed in Figure 29. *COLMAP* detected features in all images with a count ranging between 8,268 and 13,675 features per images. It was observed that the number of features detected in the images taken from the right camera had almost no variation, recording more than 12,000 features per image. And, the images taken from the camera on the left showed a big variation of detected features, with almost half the number of features calculated in certain images, according to Figure 46.

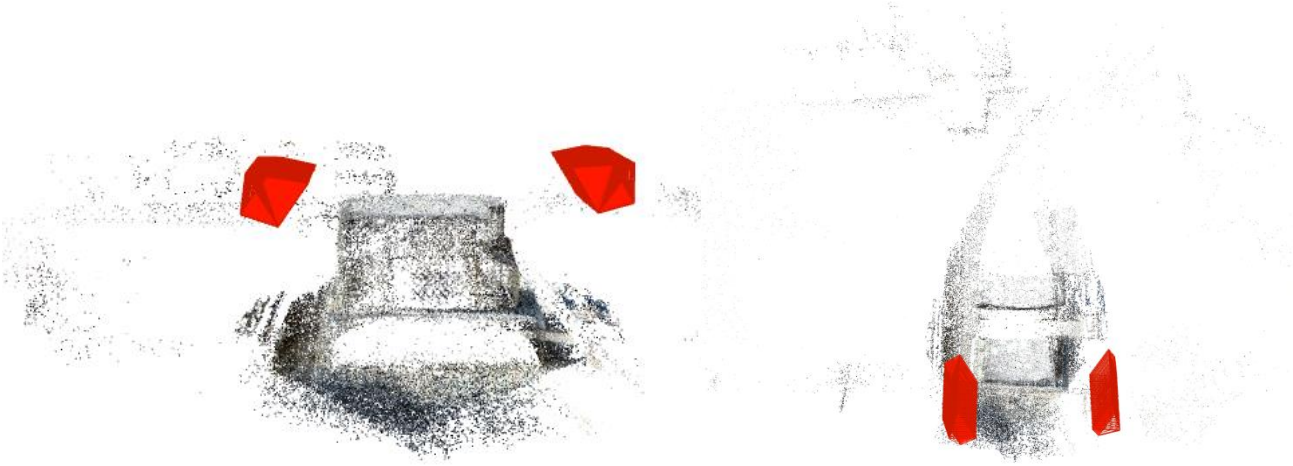


Figure 29: Sparse point cloud from COLMAP (left: front view, right: top view)

The main factors that differentiate the images, from right and left side in Figure 50 and Figure 51, are the lighting conditions and the background. Although the images were preprocessed to have similar brightness, the left side of the quarry stone line is illuminated by direct sunlight producing different quality of images. The background of the images from the right side have more texture, contrast and three-dimensional characteristics than the images taken from the left side, contributing to the difference of the number of features computed. Nevertheless, the amount of features per image and the camera poses estimated proved to be accurate enough to build a sparse cloud of tie points with similar structure to the real scene, as it is seen in Figure 29, Figure 50 and Figure 51. *COLMAP*'s sparse point cloud doesn't filter the background features detected for each image, therefore, this pipeline computes tie points for all structures in the scene, not only the foreground as it is seen in the sparse cloud generated.

It was confirmed that *Meshroom* is capable of generating a 3D cloud with geometry similar to the real scene, shown in Figure 30. This pipeline computes similar number of features for all images, approximately, 6000 key points, displayed in Figure 47. In comparison to *COLMAP*, *Meshroom* is able to build enough tie points of the quarry stone line foreground from the preprocessed images without having so many tie points in the background. This proves it is unessential to detect more than 6000 features per image, if the features correspond to the main object of the forefront. Similarly to *COLMAP*, *Meshroom* estimated 68 cameras with positions identical to the actual setting.

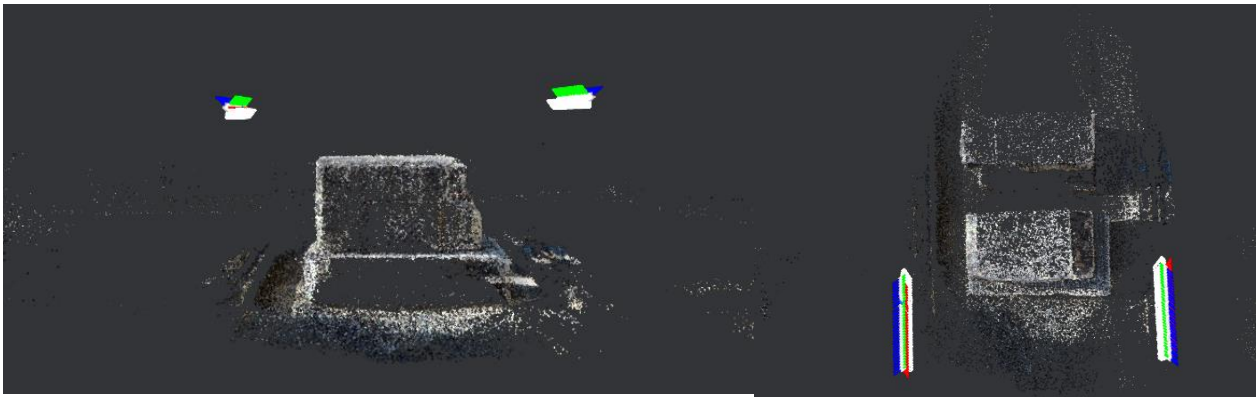


Figure 30: Sparse point cloud from Meshroom (left: front view, right: top view)

Agisoft Metashape Professional created a sparse cloud with a similar shape to the *COLMAP* and *Meshroom* cloud. In contrast to the other pipelines, this software isn't able to generate a sparse point cloud based on the original images through a standard process. The only procedure to build a 3D model from the original images is by separating the 68 images in two chunks (the 34 images from the right cameras and the 34 images from the left camera) in order to adjust the brightness and contrast to further apply the alignment and finally be merged. However, the preprocessed images achieve a tie point cloud seen in Figure 31 without adjusting brightness and contrast on the software. The alignment of the preprocessed image was executed without limiting the number of detected features and tie points, but also the quality was set on high to prevent the process of downscaling the images. The number of features detected can vary between 72,987 and 102,333 without a distinct disparity between the images from the right and left side of the limestone line, shown in Figure 48. The amount of tie points is visibly higher compared to the open source pipelines and the estimated number of cameras is 68 with similar positions to the real scene.

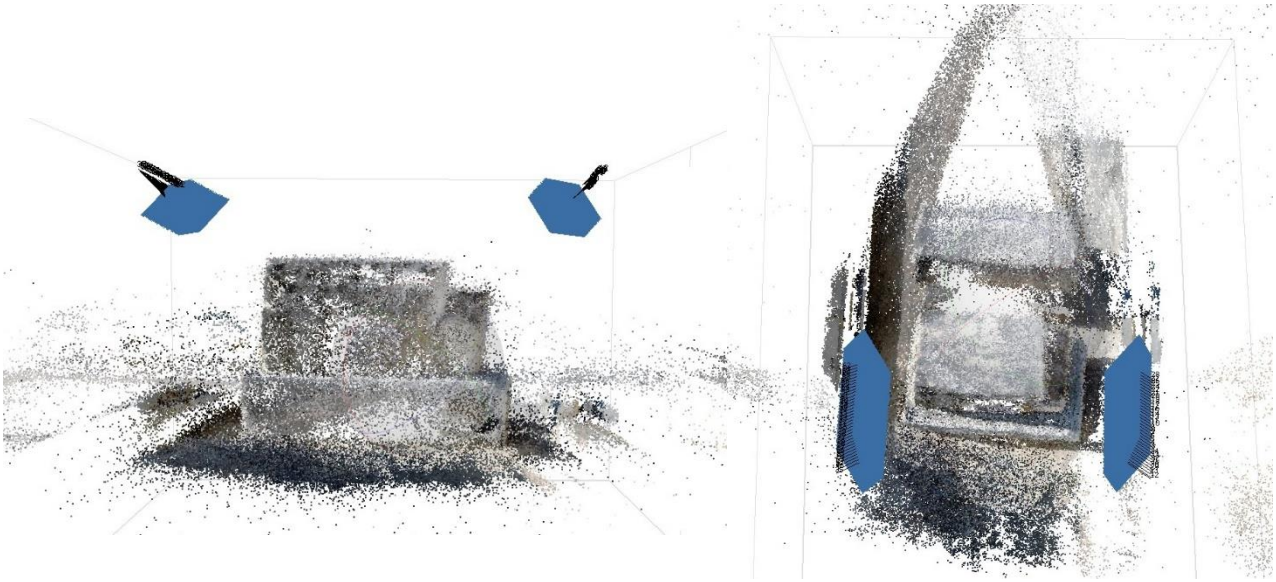


Figure 31: Sparse point cloud from Agisoft Metashape (left: front view, right: top view)

Regarding the duration of the sparse point cloud generation, *VisualSFM* achieved the fastest feature detection of the 4 pipelines analyzed, however, *VisualSFM* is an ineffective software at producing a tie point cloud due to the misestimation of the camera poses, shown in Figure 49, and therefore nonviable to continue the analysis. Based on Figure 32 and Table 6, the feature detection step took similar time through *COLMAP* and *Agisoft Metashape* implementation, however *Meshroom* performed 12.5 times slower than the other pipelines. Although feature matching was implemented through an exhaustive brute force algorithm in the open-source pipelines, *COLMAP* was 1.87 times slower than *Meshroom*. *Meshroom* achieved the fastest feature matching step since it was also 58% faster than *Agisoft*. The sparse point cloud reconstruction time corresponds to the time of the alignment process, the time aggregation of the feature detection, matching and camera pose estimation. Overall, *Meshroom* was the fastest software by taking 46.6% less time than *COLMAP* and 58.8% less than *Agisoft*. *COLMAP* accomplished a 22.8% faster performance compared to Agisoft Metashape.

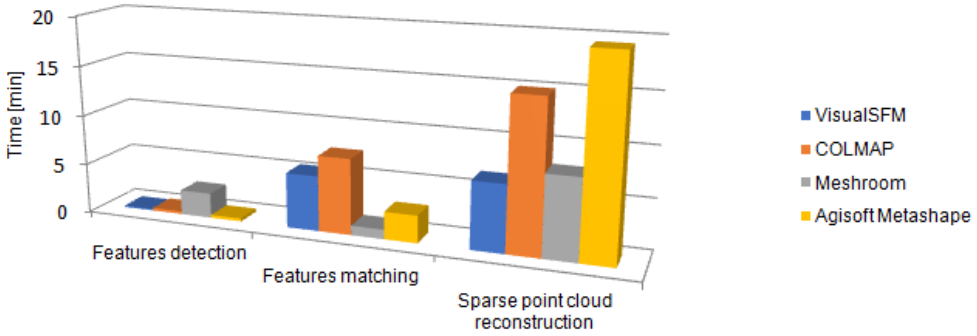


Figure 32: Sparse point cloud elapse time

Although, *VisualSFM* proved to be nonviable, this pipeline was able to detect more features than *Meshroom* since the miscalculation only occur on the last step of SFM. According to Figure 33, *COLMAP* detected 1.93 times more features than *Meshroom* but only 14% features compared to *Agisoft Metashape*. Similarly to the number of features detected, Agisoft Metashape produced a cloud with distinctly more points than the other pipelines, creating a sparse point cloud with 5.5 times more tie points than *COLMAP* and 8.5 times more than *Meshroom*.

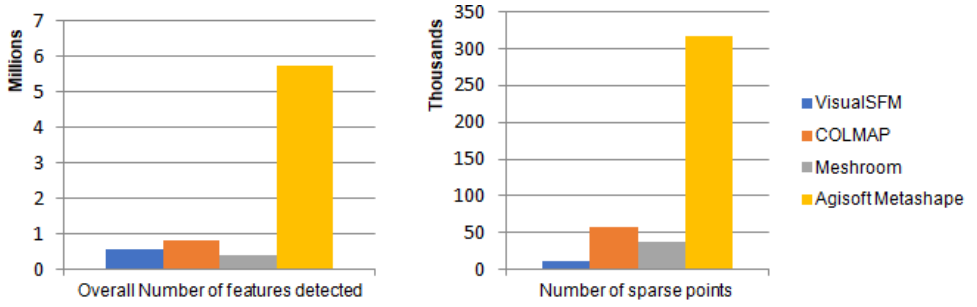


Figure 33: Number of features and sparse points

5.1.2 Dense Point Cloud

The dense point cloud's input corresponds to the output of the sparse point cloud reconstruction, consequently, the dense cloud reconstruction performance depends on the results of the previous step.

According to Figure 34, *COLMAP* produced a very detailed dense cloud with a structure resembling the actual limestone line. This phase displays an upgraded cloud compared to the sparse point cloud, it is already possible to visualize the incapability of reconstructing the backside of the limestones due to lack of information. Furthermore, the frontside of the second stone in the line shows the absence of dense points, probably, due the direct light on the left side of the block. The main downside of the dense point cloud in *COLMAP* is the amount of irrelevant dense points present on the background of the scene, shown in Figure 52.

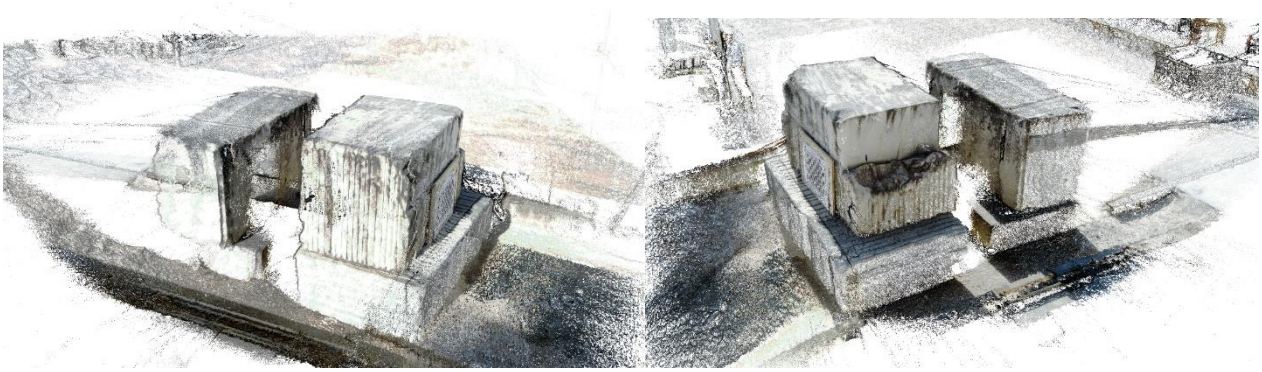


Figure 34: Dense point cloud from COLMAP

Meshroom graphical user interface doesn't display the dense point cloud due to dense cloud process considerations. The depth maps are computed in this phase however the fusion of the depth maps is only applied in the mesh generation process.

Agisoft Metashape achieves an even more complete dense reconstruction compared to *COLMAP*, based on Figure 35 and Figure 53. It is also evident the need for more information in order to build the posterior of the limestones and the front of the second stone. Another apparent detail is the focus of dense points only on the relevant forefront of the limestone line, eliminating all the information from the background.

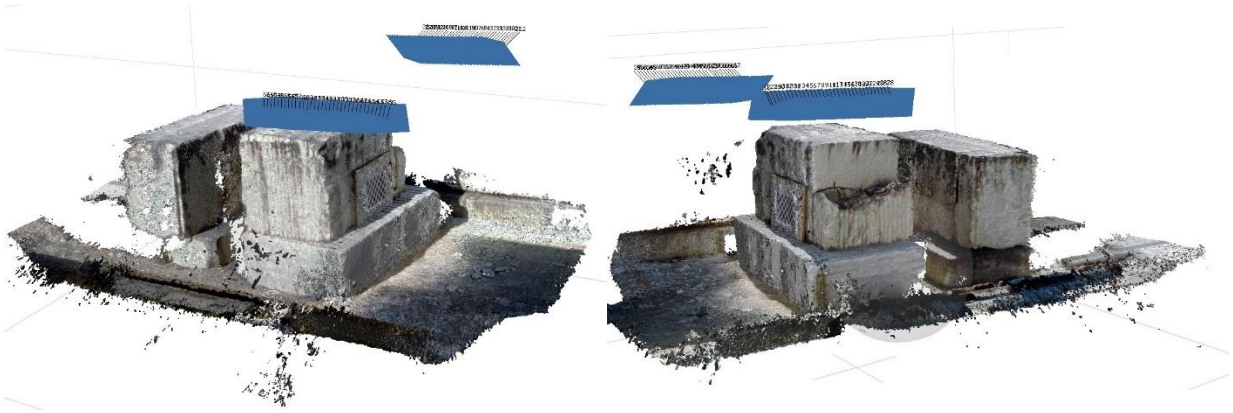


Figure 35: Dense point Cloud from Agisoft Metashape

Since Meshroom software does not display details of the dense point cloud generation, it was not possible to know the total time of this step and the number of dense points. Therefore, *Meshroom* performance can only be compared in the other stages of reconstruction. *COLMAP* was 3.5 times slower and calculated 60% less points than *Agisoft Metashape*, as it is shown in Table 8 and Figure 36.

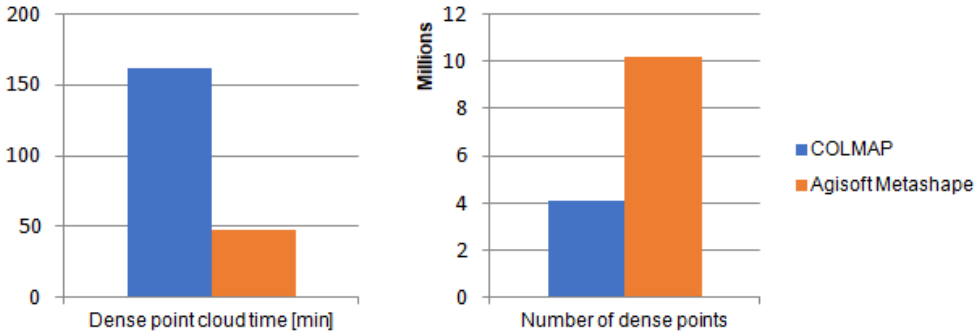


Figure 36: Dense point cloud reconstruction comparison

5.1.3 Surface Model

COLMAP's software can compute surface reconstructions but its graphical user interface is unable to show surface models. There are two methods that can be executed in *COLMAP*: Screened Poisson surface reconstruction and Delaunay triangulation-based surface reconstruction. The surface model generated based on the Poisson reconstruction, shown in Figure 55, displayed an undesirable model with acceptable structure but a substantially amount of noise.

On the other hand, Surface reconstruction based on Delaunay triangulation retrieved a model more accurate and less noisy than the Poisson surface reconstruction. but still unusable due to the imprecision of the structure compared to the real limestones, as it is shown in Figure 57. For instance, the fissure in the middle of the first limestone doesn't conform to the real scene. Based on Figure 54, a Delaunay triangulation based surface reconstruction can estimate a model resembling the ground truth on the foreground but still produces false faces and noise on the background.

MeshLab can create surfaces based on dense point cloud through the process of Screened Poisson surface reconstruction. MeshLab's input was the dense point cloud produced by *COLMAP*. Based on Figure 56, the extent of noise present in the dense point cloud from *COLMAP* was too large to generate a suitable model, creating an unusable 3D mesh that is even less akin to the ground truth than the model built by *COLMAP* with identical method.

Meshroom was able to produce an acceptable 3D mesh of the scene since most of the noise was filtered by the software, achieving a 3D model of the limestone identical to the ground truth. Contrarily to the other open source software, the holes in the back of the stones weren't closed, therefore, the formation of a structure on the back that is a misrepresentation of the real limestones was prevented but the opening on the front of both limestones in line was its consequence, observed in Figure 37.

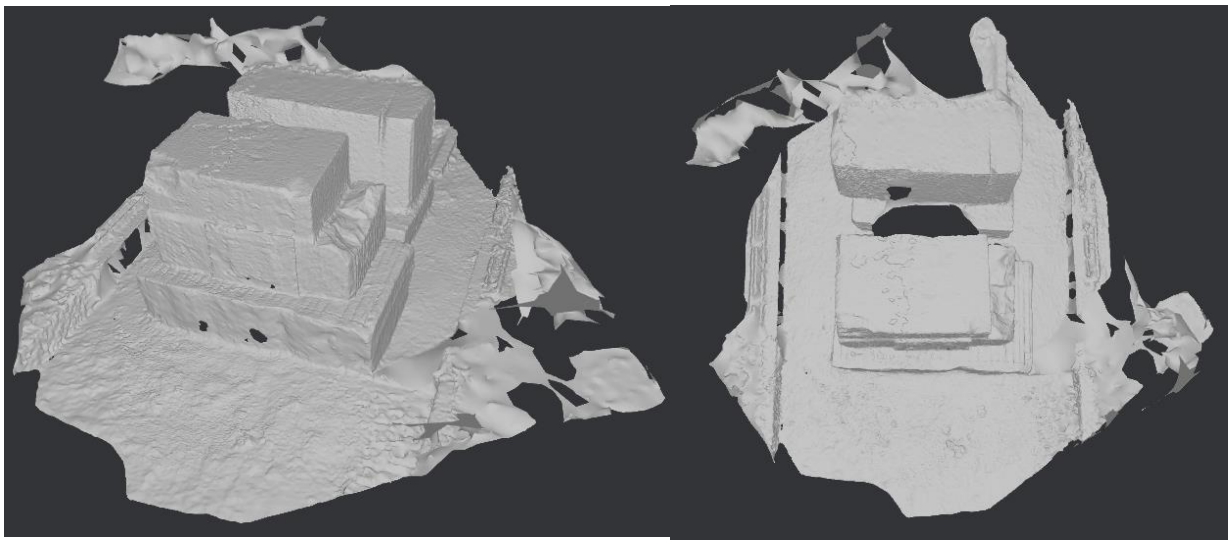


Figure 37:Delaunay triangulation based surface reconstruction from Meshroom (left: perspective from the right side; right: top view of the limestone line)

Agisoft Metashape generated a 3D surface identical to the 3D mesh created by Meshroom, regarding the nonexistence of background noise and the stone's geometry. From Figure 38, it is apparent that *Agisoft's* model has less clutter compared to Meshroom and the openings on the limestones are closed. As expected, this built a distortion on the back of each limestone, but the front openings were closed.

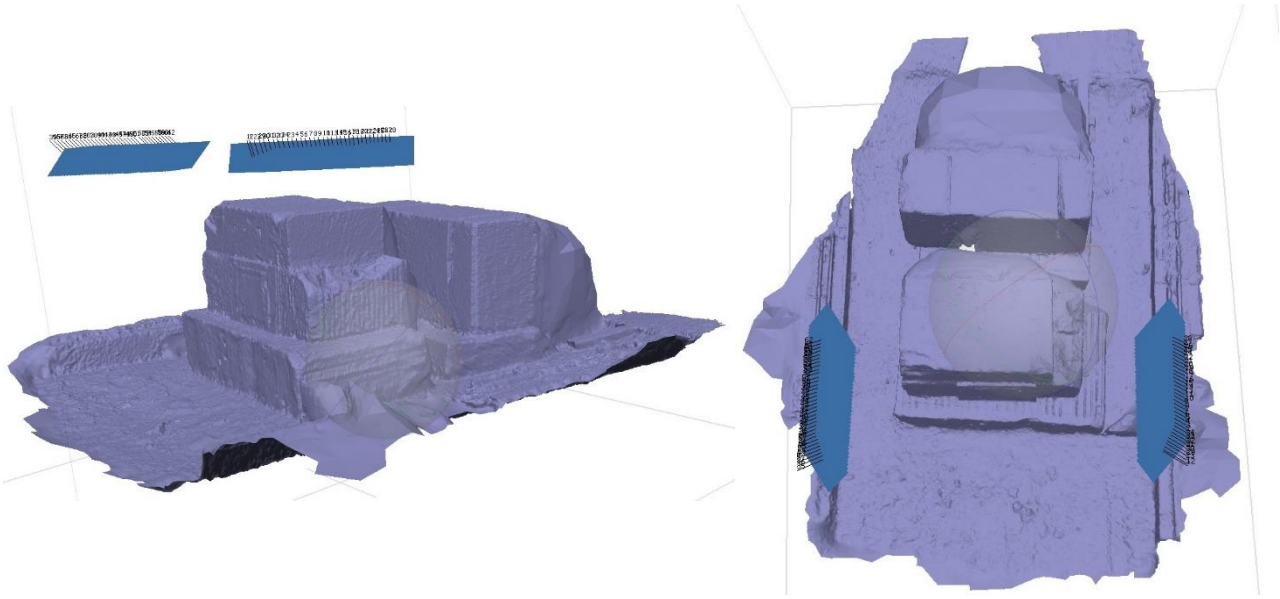


Figure 38:Surface reconstruction from Agisoft Metashape (left: perspective from the right side; right: top view of the limestone line)

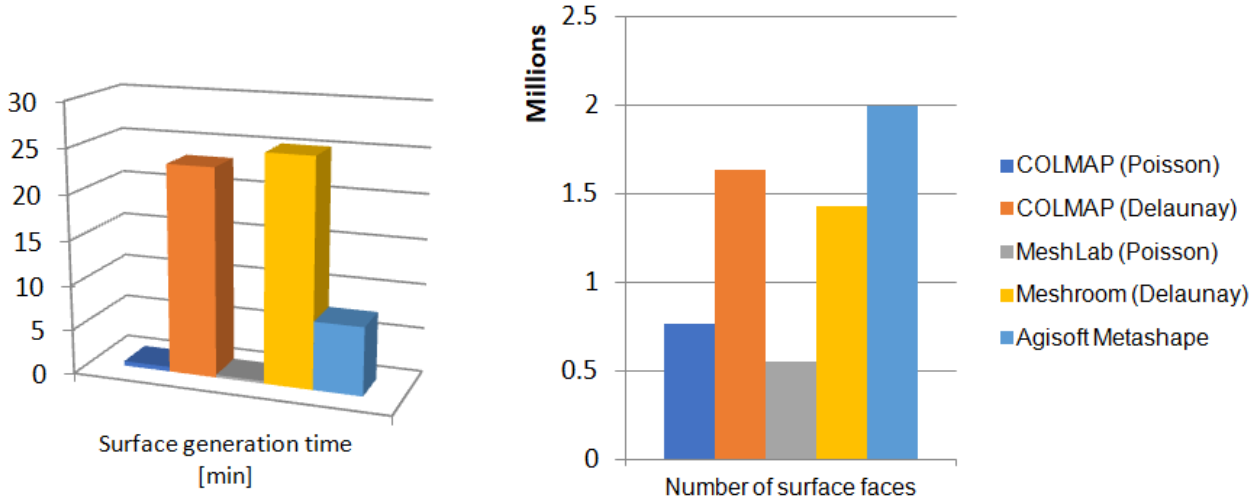


Figure 39: Surface reconstruction comparison

Despite the short amount of elapsed time of a Poisson reconstruction, it is evident that both *COLMAP* and *MeshLab* couldn't produce an applicable 3D mesh for Quarry industry use, based on the number of faces produced by this method and the images of the 3D surfaces. The Delaunay triangulation-based surface reconstruction implemented by *Meshroom* and *COLMAP* had similar number of faces and duration time. However, contrarily to Meshroom, most of the faces generated by *COLMAP* are clutter from the background and, even though, it created a better model compared to the Poisson reconstruction from *COLMAP* and MeshLab, it still has severely less detailed compared to *Meshroom* and *Agisoft Metashape*. *Agisoft Metashape* achieved a 3D surface of the quarry stone line 70% faster and a surface with 1.3 times more faces than *Meshroom*. Therefore, the open source *Meshroom* accomplished almost the same amount of

detail compared to the commercial pipeline, *Agisoft Metashape*, since there are no faces to close the openings on the surfaces, displayed in Figure 39.

5.1.4 Textured Model

The texture model reconstruction is the last phase of a 3D reconstruction, although the 3D mesh of the limestone can be sufficient to identify the proper regions of cutting stone, textured model adds information for an exact decision. *COLMAP* doesn't create a texture file for the mesh, only uses the vertex color to apply texture, therefore it isn't compared to the other pipelines.

From Figure 40, Figure 41 and Figure 58, it can be observed that *Meshroom* and *Agisoft Metashape* improved the 3D surface reconstructed previously, through texture implementation. While it was applied texture on the clutter in *Meshroom* and the opening stayed the same, *Agisoft Metashape* developed a model that textures most of the closing structure from the backside and frontside of the stones

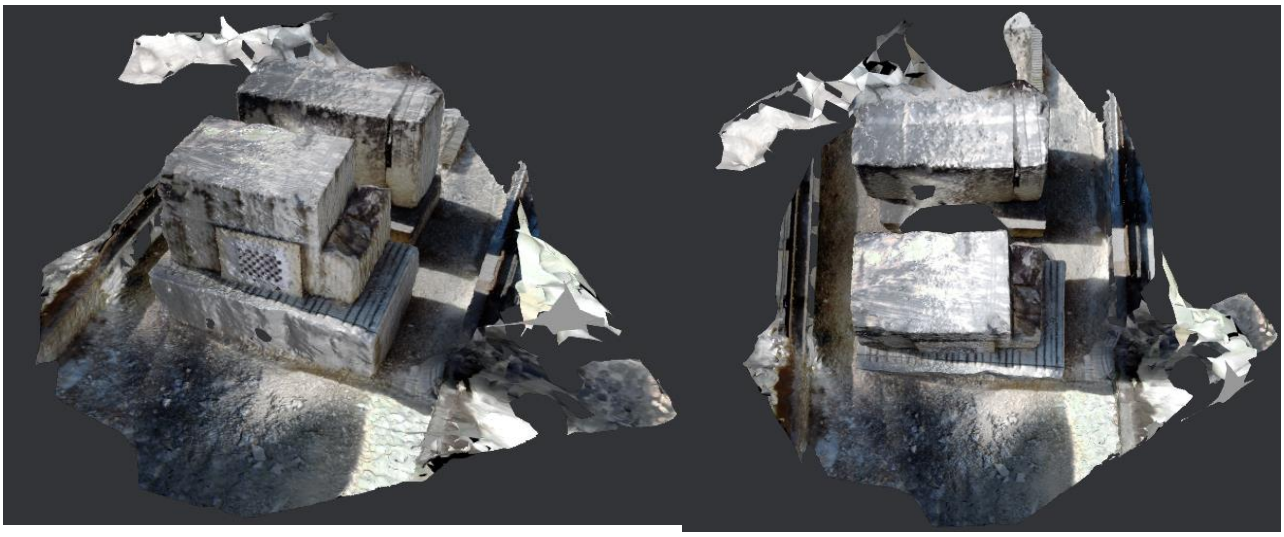


Figure 40: Texture reconstruction from Meshroom (left: perspective from the right side; right: top view of the limestone line)

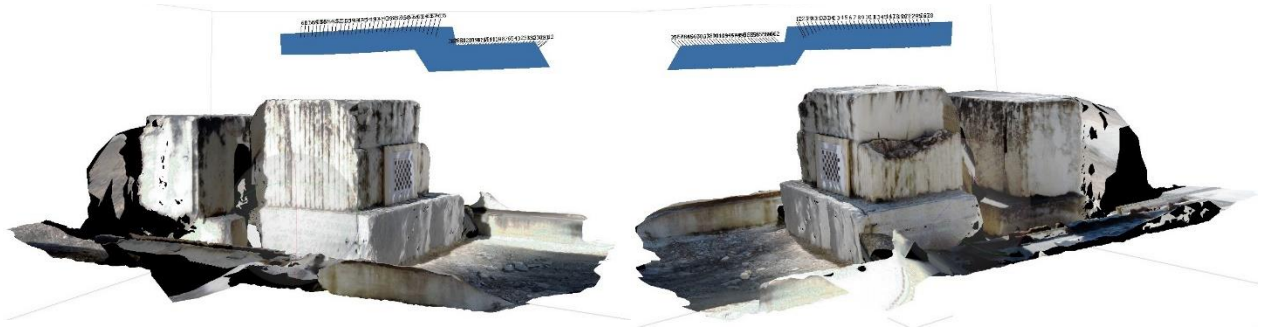


Figure 41: Texture reconstruction from Agisoft Metashape (left: perspective from the left side; right: perspective from the right side)

These pipelines required distinct amounts of time to achieve results. *Meshroom* was able to perform a texture reconstruction more than 2 times faster than *Agisoft*, according to Figure 42.

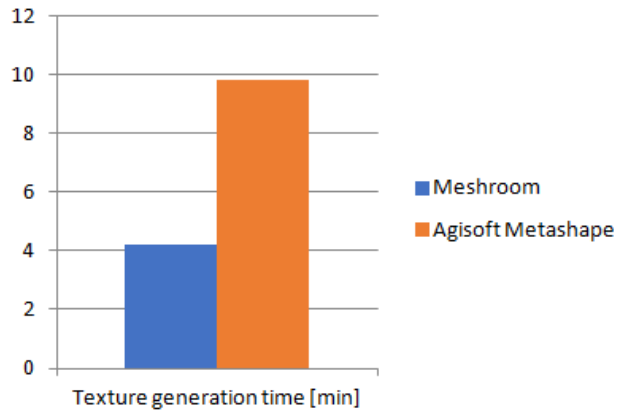


Figure 42: Textured reconstruction elapsed time

Overall, *Meshroom* took 59.71 minutes while *Agisoft Metashape* took 84.727 minutes achieving similar final 3D models. *Agisoft* produces a mesh with 30% more mesh faces than *Meshroom*, possibly, due to the existence of faces to cover the posterior of each stone in *Agisoft*. Ultimately *Meshroom* had the best performance by achieving 30% faster textured reconstruction in similar detail compared to *Agisoft Metashape.*

5.2 Stage 2

Meshroom has different algorithms to perform sparse cloud reconstruction. For dense, surface and textured reconstruction, this pipeline only has one method, therefore, the analysis to improve the current *Meshroom* reconstruction is focused in the SFM stage. *Meshroom* has different algorithms to perform feature detection, however, SIFT is the only one applied since it creates robust and accurate feature descriptors essential to produce a good reconstruction, but also, for the elapsed time being already negligible. Image matching and feature matching can be evaluated between various algorithm combinations, but camera pose estimation and correction can only be accomplish by the same established methods. Therefore, the image and feature matching algorithms are compared.

There were three image matching and three feature matching methods combined: The exhaustive, vocabulary tree and sequential methods for images matching and the brute force, ANN and Cascade Hashing for feature matching. Based on Table 13, Table 14 and Table 15, any combination built a sparse cloud with similar durations and number of sparse points, however, the method cascade hashing produced worse results in terms of number of tie points combined with any image matching process. Comparing exhaustive with brute force and exhaustive with ANN, both obtained, approximately, 37,000 tie points but Brute force in an exhaustive matching took less 0.772 minutes than ANN, displayed in Table 13. Vocabulary tree with brute force achieved a sparse cloud with slightly more points but slower than with ANN and

Sequential with ANN method had scarcely less tie points but a faster process than brute force matching, present in Table 14, Table 15, respectively. Subsequently, the number of tie points is the most valuable aspect to build a 3D model in relation to such small differences in time, therefore the best matching solution was feature matching through brute force and exhaustive image matching technique.

5.3 Stage 3

A correctly scaled 3D model can be built if the intrinsic parameters are precise and known. In order to generate a scaled model, the same set of preprocessed images were provided as input to *Meshroom* software, as well as three different types of intrinsic for comparison analysis: The intrinsic parameters without distortion coefficients and the intrinsic parameters with distortion coefficients by [12] and the intrinsic parameters generated by *Meshroom* in the previously concluded best sequence of matching algorithms. *Meshroom* doesn't have the feature to connect the stage of camera calibration of the pipeline in the UI and so, the camera models and parameters need to be manually copied to the GUI settings. The number of tie points, the time to build a sparse cloud and the overall geometry of the model were the analyzed components.

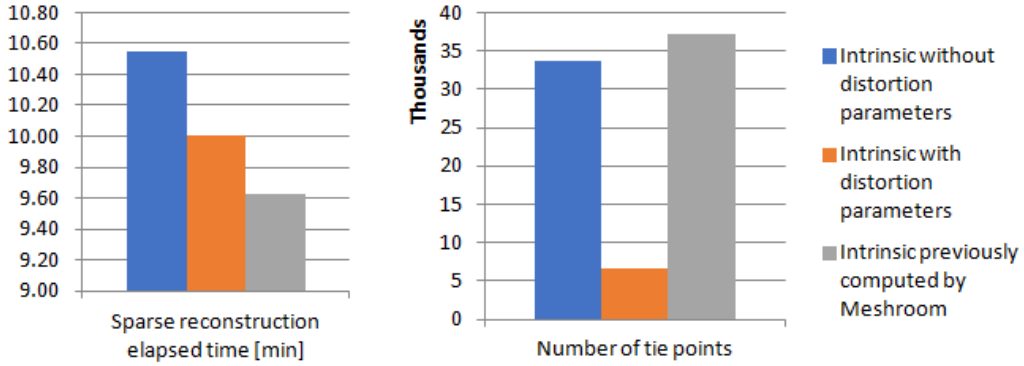


Figure 43: Intrinsic parameters input in Meshroom

Table 11 and Figure 43 show that the best apparent sparse point cloud was generated by the intrinsic parameters produced by *Meshroom* resulting in a greater number of tie points in a shorter amount of time. As shown in the Figure 60 and Figure 59, the model built from the intrinsic with distortion parameters is a disorganized cloud of sparse points while the model based on the intrinsic without distortion parameters produced a cloud with tie points captured from the right camera unaligned with the tie points originated by the images from the left. Subsequently, the intrinsic calculated by [12] proved to be unviable and the only possible application is the intrinsic by *Meshroom*, based on the distribution of the tie point in Figure 61. By giving the intrinsic previously calculated, the number of tie points increased an insignificant amount, however, the elapsed time to generate a sparse point cloud was 10% faster.

5.4 Stage 4

The final study to improve the 3D model generation quality and speed is to input the intrinsic and the extrinsic parameters along with the images in the software. From the previous analysis it was determined that only the intrinsic parameters computed by *Meshroom* are viable and therefore it is the only intrinsic used on this last analysis. The extrinsic parameters obtained by [12] were compared to the parameters computed by *Meshroom* in the same condition of the intrinsic comparison analysis. The number of tie points, the time to build a sparse cloud and the overall geometry of the model were the analyzed components.

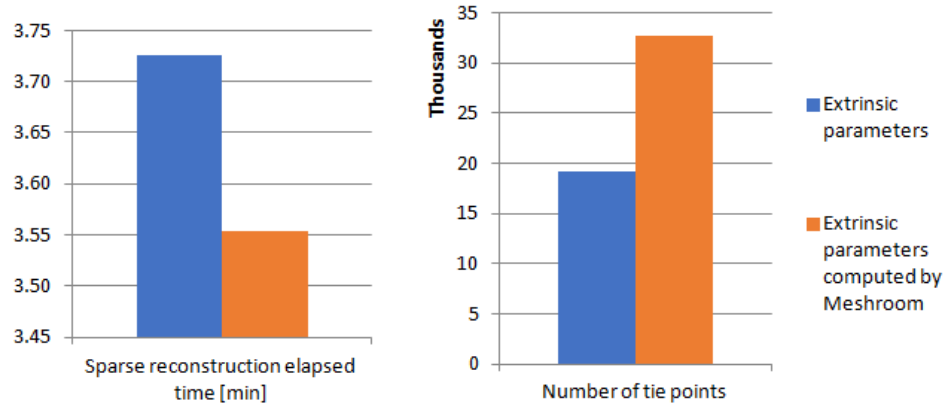


Figure 44: Intrinsic and extrinsic parameters input in Meshroom

As evidenced in Table 12 and Figure 44 the intrinsic and extrinsic parameters computed by *Meshroom* generated more tie points and was slightly faster than the parameters obtained by [12]. As shown in Figure 62 and Figure 63, The intrinsic parameters calculated by *Meshroom* with the extrinsic parameters obtained by [12] produced a sparse cloud with unaligned points from the left and right cameras introducing an unapplicable option. While the 3D model obtained by the parameters computed in *Meshroom* has a geometry similar to the real object. Although, by giving the intrinsic and extrinsic parameter previously computed by *Meshroom* the number of tie point decreased 22%, the elapsed time was 33% faster and the models still looks similar to the ground truth.

6 Conclusion and Future Work

This dissertation presented two possible pipelines to effectively reconstruct a 3D model of a quarry limestones: *Meshroom* and *Agisoft Metashape*. *Agisoft Metashape* produced the best results in terms of detail of the model and Meshroom achieved similar results but in a faster process. In order to accomplish these results, it was first conducted a research on the different 3D reconstruction pipelines in the market. There are a lot of Structure from Motion pipelines, however, a lot of these software are already outdated and the main researches on 3D model generation pipelines, reviewed on the background, concluded that *COLMAP* obtains the best results, followed by *VisualSFM*. Although, COLMAP is able to compute surface reconstruction, it is incapable of showing the model, and so MeshLab is used to not only open the surface model but also to build a 3D model based on its dense cloud. *Meshroom* is another renowned software that stands out for being the open source software based on the commercial pipeline *AliceVision* framework, also, for its update in 2019. Based on the background search, *Agisoft Metashape* is one of the most well-known commercial software to generate close range 3D models. Therefore, these five software, *VisualSFM*, *COLMAP*, *MeshLab*, *Meshroom* and *Agisoft Metashape*, were chosen to be compared and to conclude the best pipeline implementation of quarry stone images in an uncontrolled environment. Most of the algorithms implemented on the open-source pipelines are known, therefore it was chosen a standard sequence of methods for all software to achieve a 3D model for proper comparison. SIFT was the feature detection approach for all pipelines, even Agisoft applies a variation of SIFT. The features were matched through brute force in an exhaustive approach for all open source pipelines. The method PnP followed by bundle adjustment were executed to estimate and correct the camera poses, respectively. All open-source pipelines adopt RANSAC for outlier removal. *VisualSFM* and *COLMAP* use CMVS/PMVS to produce a Dense point cloud. On the other hand, *Meshroom* applies SGM. *VisualSFM* doesn't generate surface models, yet, *COLMAP* produces 3D meshes through one of two different methods: Screened Poisson surface reconstruction and Delaunay triangulation based surface reconstruction. *MeshLab* also implements Screened Poisson surface reconstruction and *Meshroom* achieves a surface model through the Delaunay triangulation based surface reconstruction.

Agisoft is the only commercial software to be compared and assumed to be one of the most capable. Regardless, it was still unfit to build a 3D model based on the images captured in the uncontrolled environment of a quarry. The images had to, previously, be adjusted in terms of brightness and contrast. Therefore, it was analyzed a set of preprocessing methods to first output the best preprocessed images in terms of brightness and contrast enhancement. The preprocessing methods implemented in *Python* were an averaging of brightness and contrast, histogram equalization, CLAHE, Gamma correction and the combination of gamma correction with histogram equalization and gamma correction with CLAHE. Considering *COLMAP* as one of the best open source pipelines, it was executed the same, previously described, algorithms during the SFM process, in order to analyze each pre-processing method. Once the images were computed, *COLMAP* was used to analyze each set of preprocessed images. The Gamma

correction method was the fastest method but retrieved a sparse cloud with less points, therefore with less detail, and CLAHE was the slowest method but still produced a point cloud with less points than other methods, consequently, both methods were excluded from being viable approaches in producing the best preprocessed images. Evidently, an averaging of brightness and contrast method was another ineffective method, with similar results to the original images. The best 3 preprocessing methods were the histogram equalization, the gamma correction with histogram equalization and gamma correction with CLAHE. Histogram equalization was the second slowest approach and the second most efficient in terms of number of detected points and sparse points built. Gamma correction with histogram equalization approach was 22.3% faster and only returned 0.9% less sparse points than the histogram equalization method. Therefore, histogram equalization was the worse between these three approaches. Finally, gamma correction with histogram equalization method obtained, only, 1.3% less sparse points compared to gamma correction with CLAHE method, however, 8.8% less time to generate a sparse cloud. Accordingly, the best preprocessed images and the input of the different 3D reconstruction software were the images obtained by the gamma correction with histogram equalization approach.

The evaluated parameters in the sparse point cloud generation were the elapse time, the number of features detected and the number of sparse points built. For the sparse point cloud analysis, *VisualSFM* obtained a cloud with tie points only of the right side of the line quarry stone due to the miscalculation and correction of the camera poses. Therefore, *VisualSFM* was an unviable pipeline for the 3D reconstruction of a limestone, proving the importance of the method bundle adjustment. COLMAP, *Meshroom* and *Agisoft Metashape* achieved good results. *Meshroom* was the fastest software by taking 46.6% less time than *COLMAP* and 58.8% less than *Agisof*t. *COLMAP* achieved a performance of 14% compared to *Agisoft Metashape*, in terms of number of sparse points, but almost, the double of the tie points in relation to *Meshroom*. The amount of sparse points doesn't indicate how good the model is, considering that most of the sparse points in *COLMAP* are the forefront but also the background, and most of the tie points in *Meshroom* are only the essential foreground.

Regarding the dense point cloud generation, *Meshroom* doesn't return enough information for comparison. Nevertheless, the output of *COLMAP* and *Agisoft Metashape* were compared concerning the number of dense points and the duration of each process. *COLMAP* obtained significantly worse results, achieving 60% less dense points and 3.5 times slower compared to *Agisoft*.

The analyze of the surface reconstruction was accomplished through the elapsed time of the process and the number of faces of the final model. *COLMAP* can generate a surface through Screened Poisson surface reconstruction and Delaunay triangulation-based surface reconstruction. MeshLab can also compute a surface based on a dense point cloud, that in this case was the output, the dense point cloud reconstructed in *COLMAP*. The Poisson produced a poor quality mesh including substantial amounts of clatter, for both *MeshLab* and *COLMAP*, despite *COLMAP* obtaining a better model than *MeshLab*, it still wasn't a viable option to clearly detect stone damage. *Meshroom* can also generate a 3D mesh based on the Delaunay

triangulation algorithm, therefore comparing the surface reconstruction of *Meshroom* and *COLMAP* through Delaunay triangulation show that *Meshroom* achieves a better performance with less noise than *COLMAP*, in spite of, *COLMAP* obtaining a model with more faces and slightly faster. The best 3D meshes obtained were the *Meshroom* and the *Agisoft* surfaces. *Agisoft* obtained a surface 70% faster and only 1.3 times more faces than *Meshroom*

The texture reconstruction was only possible to be executed by *Meshroom* and *Agisoft* and it was analyzed in duration of the process and appearance. In contrast to the other stages, Agisoft was 2 times slower than Meshroom.

Overall, *Meshroom* and *Agisoft* achieved the best performances. *Meshroom* generated a 3D textured model with 1,428,422 faces in 59.71 minutes while *Agisoft Metashape* took 84.727 minutes to compute the same model with 1,989,573 faces. The detail between both models have meaningless variations, the most apparent was the *Agisoft* attempt to close the openings on the meshes, creating a distortion on the back of the limestone, while *Meshroom* has openings in the stones. However, *Meshroom* was 30% faster than *Agisoft Metashape*. Since *Meshroom* enables the user to choose each method used in every step, it was possible to iteratively try different algorithms and conclude the fastest and complete output. Image and feature matching were applied in various algorithm combinations between the image matching methods, Exhaustive, Vocabulary tree and sequential and the feature matching methods brute-force, ANN and Cascade Hashing. Cascade hashing produced the worse results in terms of time and number of tie points and, as expected, brute-force and ANN achieved overall more tie points and lower time consumption, respectively. The result variations were trivial, nevertheless, the combination of exhaustive image matching with brute-force feature matching confirmed to be the fastest and most complete to generate a sparse cloud. Further, the intrinsic and extrinsic parameters were introduced to build the sparse point cloud in order to achieve the fastest generation of a model without compromising the model detail, based on the concluded sequence of best matching algorithms. Before introducing the extrinsic, different versions of intrinsic parameters were compared: The intrinsic parameters concluded in [12] with and without distortion coefficients and the intrinsic parameters previously computed by *Meshroom*. The number of tie points in the cloud, the elapse time and the geometry of the model were analyzed to conclude the unviability of both intrinsic parameters version in [12] but the successful reconstruction from the intrinsic parameter of *Meshroom* for decreasing the time by 10% while building a cloud with similar geometry to the ground truth. Since only the intrinsic parameters computed by *Meshroom* are applicable, these intrinsic parameters were introduced with the extrinsic concluded in [12] and along with the extrinsic previously computed by *Meshroom* for comparison. This final analysis determined that the extrinsic parameters in [12] weren't fit to produce a model with geometry similar to the real limestones while the parameter computed by *Meshroom* achieved a similar cloud structure with less 22% tie points in less 33% time than the reconstruction without the intrinsic and extrinsic parameters. Overall, the preprocess of the images captured in an uncontrolled environment proved to be crucial to build a 3D model based on open source or commercial software, further, the input of the intrinsic and extrinsic parameters to improve the time duration of a scaled sparse point cloud

the input of the intrinsic and extrinsic parameters proved to be important to reconstruct a scaled 3D model of the limestones in less 11% than without the camera parameters. It was concluded that it is possible to reconstruct a 3D model of the limestone in an uncontrolled environment through *Meshroom*, even though, it took 55.25 minutes, around 25.25 minutes more than the time needed for the cutting machine to go back to its original position in order to start cutting.

The consequence of only having imagery information on the front, left and right side, of the limestones can be seen in all 3D models. As expected, the stones are hollow on the bottom and back side of each structure. The bottom side of the limestone is unnecessary to be reconstructed, however, the back side of the stones is critical to be built for stone damage analysis. An understandable approach would be adding two more cameras with similar angle, altitude and facing the stones in the opposite position to the present cameras, not only to add information on the reconstructed faces, but also to reconstruct the back side of the stones.

Another obstacle was the direct sun light exposition during capture. Although the preprocess of the images gives the ability to reconstructing the scene, the light and shadow are still the cause for creating holes in the surface of the reconstructed model and contribute to the incorrect color of the model. This issue could be solved by creating a structure, such as a marquee, that would revolve the limestone line and respective cutting machine to avoid shadows and direct light onto the stone.

The images taken have a lot of detail, providing the model with not only limestone's details but also the details of surrounding environment. This excessive information can lead to an increase of time of process to generate a model. Either a background removal or a plain non textured structure revolving the limestone line could be possible approaches to remove the background. The background removal can be executed in two different ways: By taking the same pictures of the background without the limestones and pre-processing the images by removing the background images or by removing the background through computer vision algorithms to only filter the limestones.

The intrinsic and extrinsic were already introduced in the software to generate a model, however, these intrinsic parameters were computed by *Meshroom* and thus it does not scale the model to its true measurements or place the cameras in the exact positions. An improvement would be to recalibrate the cameras for better intrinsic parameters and recalculate the exact camera positions to introduce correct extrinsic parameters. Once the model has the correct intrinsic and extrinsic parameters, it is possible to determine real metric distances. Another method, supported by *Agisoft Metashape*, is to place markers along the scene with their real known positions and introduce them on the software to have a scaled 3D model.

The last clear improvement is to upgrade the hardware. The hardware used in this study employed an outdated GPU. While sufficient to activate the CUDA and perform the computations, it has low compute capability, therefore upgrading to a better GPU would reduce processing time.

7 References

[1] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Second Edi. 2004.

[2] L. G. Roberts, “Machine perception of three-dimensional solids,” *Doctoral thesis of Philosophy in the Electrical Engineering Department of Massachusetts Institute of Technology*, 1963, [Online]. Available: http://dspace.mit.edu/handle/1721.1/11589.

[3] J. Huang, T. Pretz, and Z. Bian, “Intelligent solid waste processing using optical sensor based sorting technology,” *Proceedings - 2010 3rd International Congress on Image and Signal Processing, CISP 2010*, vol. 4, pp. 1657–1661, 2010, doi: 10.1109/CISP.2010.5647729.

[4] S. Lopes and D. Jayaswal, “A Methodical Approach for Detection and 3-D Reconstruction of Brain Tumor in MRI,” *International Journal of Computer Applications*, vol. 118, no. 17, pp. 37–43, 2015, doi: 10.5120/20840-3580.

[5] S. Prakash, B. A. de Boer, J. Hagoort, Q. D. Gunst, J. M. Ruijter, and M. J. B. van den Hoff, “Considerations for Measurement of Embryonic Organ Growth,” *Anatomical Record*, vol. 302, no. 1, pp. 49–57, 2019, doi: 10.1002/ar.23908.

[6] J. Janai, F. Güney, A. Behl, and A. Geiger, “Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art,” *Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art*, 2020, doi: 10.1561/9781680836899.

[7] B. Renuka, B. Sivaranjani, A. M. Lakshmi, and N. Muthukumaran, “Automatic Enemy Detecting Defense Robot by using Face Detection Technique,” *Asian Journal of Applied Science and Technology*, vol. 2, no. 2, pp. 495–501, 2018.

[8] B. G. Abdelaty, M. A. Soliman, and A. N. Ouda, “Reducing Human Effort of the Optical Tracking of Anti-Tank Guided Missile Targets via Embedded Tracking System Design,” *American Journal of Artificial Intelligence*, vol. 2, no. 2, pp. 30–35, 2018, doi: 10.11648/j.ajai.20180202.13.

[9] Teledyne DALSA, “Genie Nano Cameras,” 2017, [Online]. Available: http://info.teledynedalsa.com/acton/attachment/14932/f-0602/1/-/-/-/-/Genie_Nano_Family_datasheet.pdf.

[10] GOYO OPTICAL INC., “Industrial Lens: Item No . GM12HR58018MCN.” .

[11] Neousys technology, “Nuvo-5000E / P Series,” 2018.

[12] F. F. Monteiro, “Integration of an optical system for 3D reconstruction,” *Master thesis of science degree in Mechanical Engineering*, no. January, p. Instituto Superiror Técnico, 2021.

[13] “OpenCV.” https://opencv.org/ (accessed Oct. 08, 2020).

[14] Z. Al-Ameen, G. Sulong, A. Rehman, A. Al-Dhelaan, T. Saba, and M. Al-Rodhaan, “An innovative technique for contrast enhancement of computed tomography images using normalized gamma-corrected contrast-limited adaptive histogram equalization,” *Eurasip Journal on Advances in Signal Processing*, vol. 2015, no. 1, pp. 1–12, 2015, doi: 10.1186/s13634-015-0214-1.

[15] S. El Hazzat, A. Saaidi, and K. Satori, “Euclidean 3D reconstruction of unknown objects from multiple images,” *Journal of Emerging Technologies in Web Intelligence*, vol. 6, no. 1, pp. 59–63, 2014, doi: 10.4304/jetwi.6.1.59-63.

[16] E. Vural and A. A. Alatan, “Outlier removal for sparse 3D reconstruction from video,” *2008 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video, 3DTV-CON 2008 Proceedings*, pp. 341–344, 2008, doi: 10.1109/3DTV.2008.4547878.

[17] Y. Furukawa and C. Hernández, “Multi-View Stereo : A Tutorial,” *Foundations and Trends in Computer Graphics and Vision*, vol. 9, no. 1–2, pp. 1–148, 2013.

[18] G. A. Christie, “Computer Vision for Quarry Applications,” *Medical Molecular Morphology*, vol. 42, no. 3, pp. 162–166, 2009, doi: 10.1007/s00795-009-0455-x.

[19] P. Rossi, F. Mancini, M. Dubbini, F. Mazzone, and A. Capra, “Combining nadir and oblique uav imagery to reconstruct quarry topography: Methodology and feasibility analysis,” *European Journal of Remote Sensing*, vol. 50, no. 1, pp. 211–221, 2017, doi: 10.1080/22797254.2017.1313097.

[20] H. Huang *et al.*, “Field Imaging and Volumetric Reconstruction of Riprap Rock and Large-Sized Aggregates: Algorithms and Application,” *Transportation Research Record*, vol. 2673, no. 9, pp. 575–589, 2019, doi: 10.1177/0361198119848704.

[21] C. Robiati, M. Eyre, C. Vanneschi, M. Francioni, A. Venn, and J. Coggan, “Application of remote sensing data for evaluation of rockfall potential within a quarry slope,” *ISPRS International Journal of Geo-Information*, vol. 8, no. 9, 2019, doi: 10.3390/ijgi8090367.

[22] S. Li, “A review of feature detection and match algorithms for localization and mapping,” *IOP Conference Series: Materials Science and Engineering*, vol. 231, no. 1, 2017, doi: 10.1088/1757-899X/231/1/012003.

[23] D. G. Low, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, pp. 91–110, 2004, [Online]. Available: https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf.

[24] E. Karami, S. Prasad, and M. Shehata, “Image Matching Using SIFT, SURF, BRIEF and ORB: Performance Comparison for Distorted Images,” *in Proceedings of the 2015 Newfoundland Electrical and Computer Engineering Conference, St. John’s, Canada, November,2015*.

[25] L. C. Chiu, T. S. Chang, J. Y. Chen, and N. Y. C. Chang, “Fast SIFT design for real-time visual feature extraction,” *IEEE Transactions on Image Processing*, vol. 22, no. 8, pp. 3158–3167, 2013, doi: 10.1109/TIP.2013.2259841.

[26] M. Güzel, “A Hybrid Feature Extractor using Fast Hessian Detector and SIFT,” *Technologies*, vol. 3, no. 2, pp. 103–110, 2015, doi: 10.3390/technologies3020103.

[27] Y. Ke and R. Sukthankar, “PCA-SIFT: A more distinctive representation for local image descriptors,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 2–9, 2004, doi: 10.1109/cvpr.2004.1315206.

[28] P. Mainali, G. Lafruit, Q. Yang, B. Geelen, L. Van Gool, and R. Lauwereins, “SIFER: Scale-invariant feature detector with error resilience,” *International Journal of Computer Vision*, vol. 104, no. 2, pp. 172–197, 2013, doi: 10.1007/s11263-013-0622-3.

[29] A. Leonardis, H. Bischof, and A. Pinz, *Computer Vision ECCV 2006*, vol. Part I. 2006.

[30] P. Y.Simard, L. Bottou, P. Haffner, and Y. LeCun, “Boxlets: a Fast Convolution Algorithm for Signal Processing and Neural Networks,” *Journal of Visual Languages & Computing*, p. 7, 1998.

[31] Y. Rubner, C. Tomasi, and L. J. Guibas, “Earth mover’s distance as a metric for image retrieval,” *International Journal of Computer Vision*, vol. 40, no. 2, pp. 99–121, 2000, doi: 10.1023/A:1026543900054.

[32] J. L. Bentley, “Multidimensional Binary Search Trees Used for Associative Searching,” *Communications of the ACM*, vol. 18, no. 9, 1975, doi: 10.1145/361002.361007.

[33] A. Gionis, P. Indyk, and R. Motwani, “Similarity Search in High Dimensions via Hashing,” *Proceedings of the 25th International Conference on Very Large Data Bases*, pp. 518–529, 1999, [Online]. Available: http://dl.acm.org/citation.cfm?id=645925.671516.

[34] A. Guttman, “R-trees: A dynamic index structure for spatial searching,” *ACM SIGMOD Record*, vol. 14, no. 2, pp. 47–57, 1984, doi: 10.1145/971697.602266.

[35] D. Comer, “Ubiquitous B-Tree.,” *Computing surveys*, vol. 11, no. 2, pp. 121–137, 1979.

[36] J. Cheng, C. Leng, J. Wu, H. Cui, and H. Lu, “Fast and accurate image matching with cascade hashing for 3D reconstruction,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2014, doi: 10.1109/CVPR.2014.8.

[37] X. S. Gao, X. R. Hou, J. Tang, and H. F. Cheng, “Complete solution classification for the perspective-three-point problem,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 8, pp. 930–943, 2003, doi: 10.1109/TPAMI.2003.1217599.

[38] P. D. Fiore, “Efficient linear solution of exterior orientation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 140–148, 2001, doi: 10.1109/34.908965.

[39] D. Nistér, “An efficient solution to the five-point relative pose problem,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 756–770, 2004, doi: 10.1109/TPAMI.2004.17.

[40] R. I. Hartley, “In Defense of the Eight-Point Algorithm,” *EEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, vol. 19, no. 6, pp. 580–593, 1996.

[41] M. Zhang, G. Wang, H. Chao, and F. Wu, “Fast and robust algorithm for fundamental matrix estimation,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9164, no. July 2015, pp. 316–322, 2015, doi: 10.1007/978-3-319-20801-5_34.

[42] B. Triggs, A. Zisserman, and R. Szeliski, *Vision Algorithms: Theory and Practice*. 1999.

[43] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon, “Bundle Adjustment — A Modern Synthesis,” *International Workshop on Vision Algorithms, Sep 2000, Corfu, Greece. pp.298–372, 10.1007/3-540-44480-7_21 . inria-00548290*.

[44] M. I. A. Lourakis and A. A. Argyros, “Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment?,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. II, pp. 1526–1531, 2005, doi: 10.1109/ICCV.2005.128.

[45] Y. Chen, Y. Chen, and G. Wang, “Bundle Adjustment Revisited,” *Computer Vision and Pattern Recognition*, 2019, [Online]. Available: http://arxiv.org/abs/1912.03858.

[46] M. A. Fischler and R. C. Bolles, “Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography,” *Graphics and Image Processing*, vol. 24, no. 6, pp. 381–395, 1981, doi: 10.1145/358669.358692.

[47] O. Chum and J. Matas, “Matching with PROSAC - Progressive sample consensus,” *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, vol. I, pp. 220–226, 2005, doi: 10.1109/CVPR.2005.221.

[48] B. J. Tordoff and D. W. Murray, “Guided-MLESAC: Faster image transform estimation by using matching priors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1523–1535, 2005, doi: 10.1109/TPAMI.2005.199.

[49] K. Ni, H. Jin, and F. Dellaert, “GroupSAC: Efficient consensus in the presence of groupings,” *Proceedings of the IEEE International Conference on Computer Vision*, no. Iccv, pp. 2193–2200, 2009, doi: 10.1109/ICCV.2009.5459241.

[50] D. Nistér, “Preemptive RANSAC for live structure and motion estimation,” *Machine Vision and Applications*, vol. 16, no. 5, pp. 321–329, 2005, doi: 10.1007/s00138-005-0006-y.

[51] S. Bianco, G. Ciocca, and D. Marelli, “Evaluating the performance of structure from motion pipelines,” *Journal of Imaging*, vol. 4, no. 8, pp. 1–18, 2018, doi: 10.3390/jimaging4080098.

[52] O. Chum, J. Matas, and J. Kittler, “Locally optimized RANSAC,” *Lecture Notes in Computer*

Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 2781, pp. 236–243, 2003, doi: 10.1007/978-3-540-45243-0_31.

[53] P. Moulon, R. Marlet, and P. Monasse, “Adaptive Structure from Motion with a Contratio Model Estimation,” *Proceedings of the 11th Asian conference on Computer Vision - Volume Part IV*, pp. 434–448, doi: 10.1007/978-3-642-37447-0.

[54] M. Der Yang, C. F. Chao, K. S. Huang, L. Y. Lu, and Y. P. Chen, “Image-based 3D scene reconstruction and exploration in augmented reality,” *Automation in Construction*, vol. 33, pp. 48–60, 2013, doi: 10.1016/j.autcon.2012.09.017.

[55] C. Koch, S. Paal, A. Rashidi, Z. Zhu, M. König, and I. Brilakis, “Achievements and challenges in machine vision-based inspection of large concrete structures,” *Advances in Structural Engineering*, vol. 17, no. 3, pp. 303–318, 2014, doi: 10.1260/1369-4332.17.3.303.

[56] Y. Furukawa and J. Ponce, “Accurate, dense, and robust multiview stereopsis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 8, pp. 1362–1376, 2010, doi: 10.1109/TPAMI.2009.161.

[57] H. Hirschmüller, “Stereo processing by semiglobal matching and mutual information,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, 2008, doi: 10.1109/TPAMI.2007.1166.

[58] A. Khaloo and D. Lattanzi, “Hierarchical dense structure-from-motion reconstructions for infrastructure condition assessment,” *Journal of Computing in Civil Engineering*, vol. 31, no. 1, pp. 1–13, 2016, doi: 10.1061/(ASCE)CP.1943-5487.0000616.

[59] W. Yuan, S. Chen, Y. Zhang, J. Gong, and R. Shibasaki, “An aerial-image dense matching approach based on optical flow field,” *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, vol. 41, no. July, pp. 543–548, 2016, doi: 10.5194/isprsarchives-XLI-B3-543-2016.

[60] F. C. Yang, C. H. Kuo, J. J. Wing, and C. K. Yang, “Reconstructing the 3D solder paste surface model using image processing and artificial neural network,” *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, pp. 3051–3056, 2004, doi: 10.1109/ICSMC.2004.1400799.

[61] M. Abdel, I. Taha, A. Hussein, and M. Rawan, “An enhanced algorithm for surface reconstruction from cloud points,” *Communications in Computer and Information Science*, pp. 42–49, 2013, doi: 10.1007/978-3-642-41647-7_6.

[62] J. Barhak and A. Fischer, “Parameterization and reconstruction from 3D scattered points based on neural network and PDE techniques,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 7, no. 1, pp. 1–16, 2001, doi: 10.1109/2945.910817.

[63] H. K. Zhao, S. Osher, and R. Fedkiw, “Fast surface reconstruction using the level set method,” *Proceedings - IEEE Workshop on Variational and Level Set Methods in Computer Vision, VLSM 2001*, pp. 194–199, 2001, doi: 10.1109/VLSM.2001.938900.

[64] R. Goldenthal and M. Bercovier, “Design of Curves and Surfaces Using Multi-Objective Optimization,” pp. 1–12.

[65] A. Saeedfar and K. Barkeshli, “Shape reconstruction of three-dimensional conducting curved plates using physical optics, NURBS modeling, and genetic algorithm,” *IEEE Transactions on Antennas and Propagation*, vol. 54, no. 9, pp. 2497–2507, 2006, doi: 10.1109/TAP.2006.880662.

[66] N. Amenta, M. Bern, and M. Kamvysselis, “A new voronoi-based surface reconstruction algorithm,” *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1998*, pp. 415–422, 1998, doi: 10.1145/280814.280947.

[67] T. G. Ni and Z. H. Ma, “A fast surface reconstruction algorithm for 3D unorganized points,” *ICCET*

2010 - 2010 International Conference on Computer Engineering and Technology, Proceedings, vol. 7, pp. 15–18, 2010, doi: 10.1109/ICCET.2010.5485908.

[68] X. Xu and K. Harada, “Automatic surface reconstruction with alpha-shape method,” *Visual Computer*, vol. 19, no. 7–8, pp. 431–443, 2003, doi: 10.1007/s00371-003-0207-1.

[69] K. Zhou, M. Gong, X. Huang, and B. Guo, “Data-parallel octrees for surface reconstruction,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 5, pp. 669–681, 2011, doi: 10.1109/TVCG.2010.75.

[70] H. Xie, K. T. McDonnell, and H. Qin, “Surface reconstruction of noisy and defective data sets,” *IEEE Visualization 2004 - Proceedings, VIS 2004*, pp. 259–266, 2004, doi: 10.1109/visual.2004.101.

[71] P. Z. Wen, X. J. Wu, Y. Zhu, and X. W. Peng, “LS-RBF network based 3D surface reconstruction method,” *2009 Chinese Control and Decision Conference, CCDC 2009*, pp. 5785–5789, 2009, doi: 10.1109/CCDC.2009.5195232.

[72] P. Breitkopf, H. Naceur, A. Rassineux, and P. Villon, “Moving least squares response surface approximation: Formulation and metal forming applications,” *Computers and Structures*, vol. 83, no. 17–18, pp. 1411–1428, 2005, doi: 10.1016/j.compstruc.2004.07.011.

[73] X. Li, W. Wan, X. Cheng, and B. Cui, “An improved poisson surface reconstruction algorithm,” *ICALIP 2010 - 2010 International Conference on Audio, Language and Image Processing, Proceedings*, pp. 1134–1138, 2010, doi: 10.1109/ICALIP.2010.5685081.

[74] M. Kazhdan, M. Bolitho, and H. Hoppe, “Poisson Surface Reconstruction,” *ACM Transactions on Graphics*, vol. 32, no. 3, 2013, doi: 10.1145/2487228.2487237.

[75] M. Kazhdan and H. Hoppe, “Screened poisson surface reconstruction,” *ACM Transactions on Graphics*, vol. 32, no. 3, pp. 1–13, 2013, doi: 10.1145/2487228.2487237.

[76] D. N. Ostrov, “Boundary conditions and fast algorithms for surface reconstructions from synthetic aperture radar data,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 37, no. 1 II, pp. 335–346, 1999, doi: 10.1109/36.739066.

[77] S. Osechinskiy and F. Kruggel, “PDE-based reconstruction of the cerebral cortex from MR images,” *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC’10*, pp. 4278–4283, 2010, doi: 10.1109/IEMBS.2010.5626179.

[78] J. Fayer, B. Durix, S. Gasparini, and G. Morin, “Texturing and inpainting a complete tubular 3D object reconstructed from partial views,” *Computers and Graphics (Pergamon)*, vol. 74, pp. 126–136, 2018, doi: 10.1016/j.cag.2018.05.012.

[79] J. L. Schonberger and J. M. Frahm, “Structure-from-Motion Revisited,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 4104–4113, 2016, doi: 10.1109/CVPR.2016.445.

[80] J. De Reu, P. De Smedt, D. Herremans, M. Van Meirvenne, P. Laloo, and W. De Clercq, “On introducing an image-based 3D reconstruction method in archaeological excavation practice,” *Journal of Archaeological Science*, vol. 41, pp. 251–262, 2014, doi: 10.1016/j.jas.2013.08.020.

[81] Z. Ma and S. Liu, “A review of 3D reconstruction techniques in civil engineering and their applications,” *Advanced Engineering Informatics*, vol. 37, no. March, pp. 163–174, 2018, doi: 10.1016/j.aei.2018.05.005.

[82] S. Izadi *et al.*, “KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera,” *UIST’11 - Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, pp. 559–568, 2011, doi: 10.1145/2047196.2047270.

[83] B. Curless, “From Range Scans to 3D Models,” *ACM SIGGRAPH Computer Graphics*, vol. 33, no.

4, 1999.

[84] E. Mostafa Abdel-Bary, “3D Laser Scanners Techniques Overview,” *International Journal of Science and Research*, vol. 4, no. 10, pp. 323–331, 2015, [Online]. Available: www.ijsr.net.

[85] J. Butime, I. Gutierrez, L. G. Corzo, and C. F. Espronceda, “3D reconstruction methods, a survey,” *VISAPP 2006 - Proceedings of the 1st International Conference on Computer Vision Theory and Applications*, vol. 2, pp. 457–463, 2006, doi: 10.5220/0001369704570463.

[86] S. G. Sowmya and S. Dixit, “3D Reconstruction Methodologies : a Review,” *International Journal of Advanced Networking & Applications (IJANA)*, pp. 2–4, 2016.

[87] M. Siudak and P. Rokita, “A survey of passive 3D reconstruction methods on the basis of more than one image,” *Machine Graphics and Vision*, vol. 23, no. 3–4, pp. 57–117, 2014.

[88] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 1998, doi: 10.1109/34.888718.

[89] Agisoft, “Agisoft Metashape User Manual,” p. 160, 2020, [Online]. Available: https://www.agisoft.com/pdf/metashape-pro_1_5_en.pdf.

[90] “Meshroom Manual — Meshroom v2020.1.0 documentation.” https://meshroom-manual.readthedocs.io/en/latest/ (accessed Nov. 12, 2020).

[91] “VisualSFM : A Visual Structure from Motion System.” http://ccwu.me/vsfm/ (accessed Oct. 05, 2020).

[92] J. L. Schonberger, “COLMAP — COLMAP 3.7 documentation.” https://colmap.github.io/ (accessed Jan. 28, 2021).

[93] J. L. Schönberger, E. Zheng, J. M. Frahm, and M. Pollefeys, “Pixelwise view selection for unstructured multi-view stereo,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9907 LNCS, pp. 501–518, 2016, doi: 10.1007/978-3-319-46487-9_31.

[94] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, “MeshLab: An open-source mesh processing tool,” *6th Eurographics Italian Chapter Conference 2008 - Proceedings*, no. February 2015, pp. 129–136, 2008, doi: 10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.

[95] M. Callieri, G. Ranzuglia, M. Dellepiane, P. Cignoni, and R. Scopigno, “Meshlab as a Complete Open Tool for the Integration of Photos and Colour with High- Resolution 3D Geometry Data Marco,” *Computer Applications and Quantitative Methods in Archaeology 1990*, no. 565, 2011.

[96] D. R. Finley, “HSP Color Model - Alternative to HSV (HSB) and HSL,” 2006. http://alienryderflex.com/hsp.html (accessed Mar. 29, 2021).

[97] E. A. Fedorovskaya, M. E. Miller, and P. D. Snyder, “Image specific perceived overall contrast prediction,” US 6,983,083 B2, 2006.

[98] T. Tommasini, A. Fusiello, E. Trucco, and V. Roberto, “Making good features track better,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, no. 1, pp. 178–183, 1998, doi: 10.1109/CVPR.1998.698606.

[99] Q. T. Luong and O. D. Faugeras, “The fundamental matrix: Theory, algorithms, and stability analysis,” *International Journal of Computer Vision*, vol. 17, no. 1, pp. 43–75, 1996, doi: 10.1007/BF00127818.

[100] V. Lepetit, F. Moreno-Noguer, and P. Fua, “EPnP: An accurate O(n) solution to the PnP problem,” *International Journal of Computer Vision*, vol. 81, no. 2, pp. 155–166, 2009, doi: 10.1007/s11263-008-0152-6.

[101] A. Penate-Sanchez, J. Andrade-Cetto, and F. Moreno-Noguer, “Exhaustive linearization for robust camera pose and focal length estimation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 10, pp. 2387–2400, 2013, doi: 10.1109/TPAMI.2013.36.

[102] Z. Zhao, D. Ye, X. Zhang, G. Chen, and B. Zhang, “Improved direct linear transformation for parameter decoupling in camera calibration,” *Algorithms*, vol. 9, no. 2, 2016, doi: 10.3390/a9020031.

[103] “Ceres Solver — A Large Scale Non-linear Optimization Library.” http://ceres-solver.org/ (accessed Feb. 25, 2021).

[104] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz, “Multicore bundle adjustment,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, no. 1, pp. 3057–3064, 2011, doi: 10.1109/CVPR.2011.5995552.

[105] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, “Towards internet-scale multi-view stereo,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, no. June, pp. 1434–1441, 2010, doi: 10.1109/CVPR.2010.5539802.

8 Annex

Result tables and images

The number of features detected in each pipeline is displayed in Figure 45, Figure 46, Figure 47 and Figure 48 based on the same algorithm SIFT. The camera on the right corresponds to the images between 1 and 34 and the camera on the left to the images between 35 and 68.

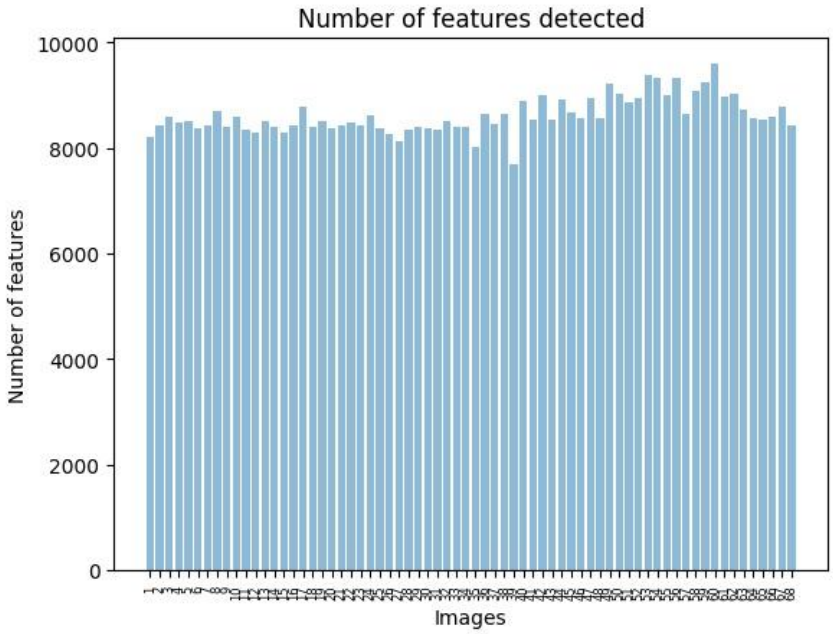


Figure 45: Number of features detected in VisualSFM

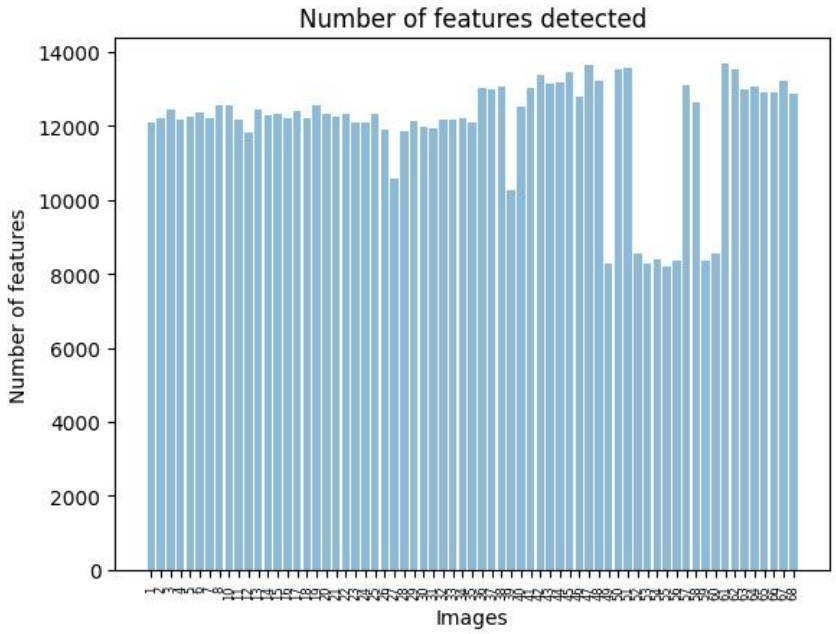


Figure 46: Number of features detected in COLMAP

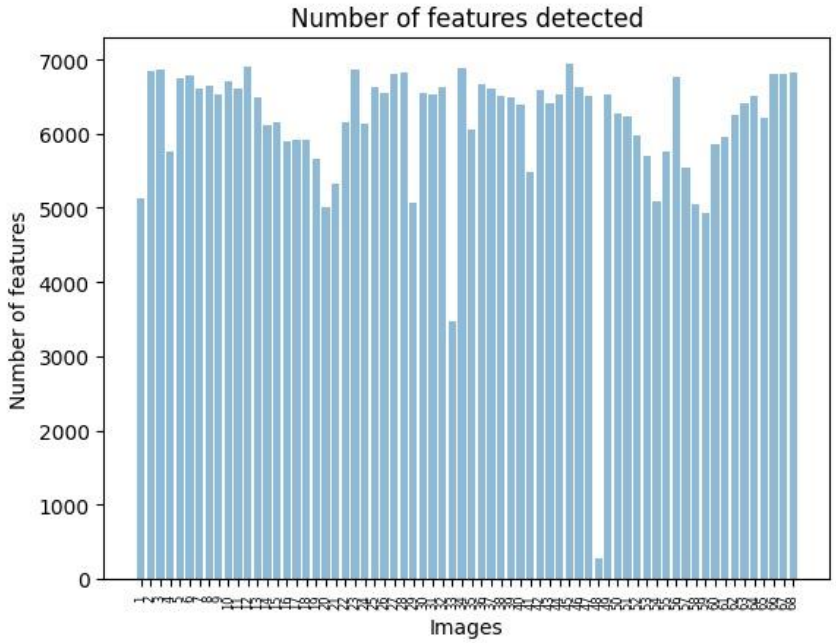


Figure 47: Number of features detected in Meshroom

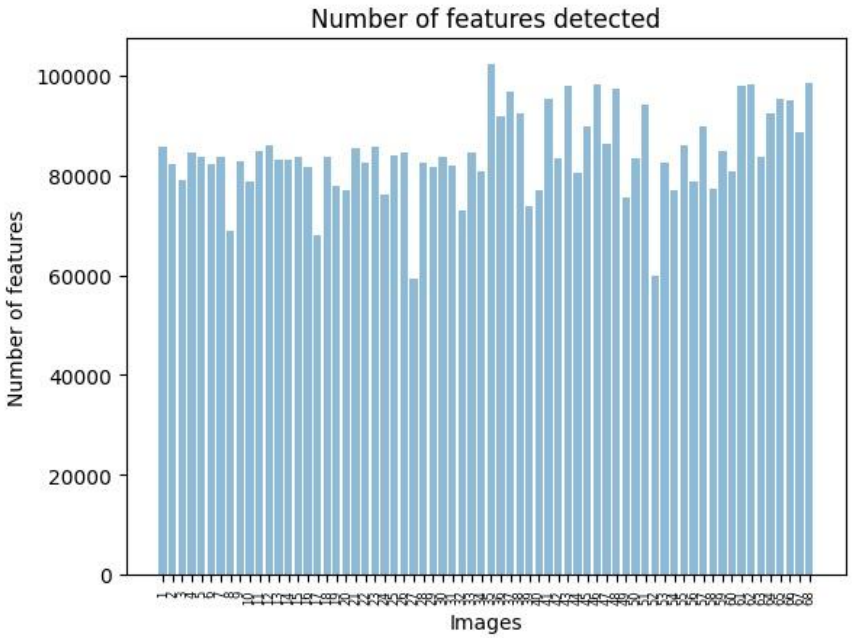


Figure 48: Number of features detected in Agisoft Metashape

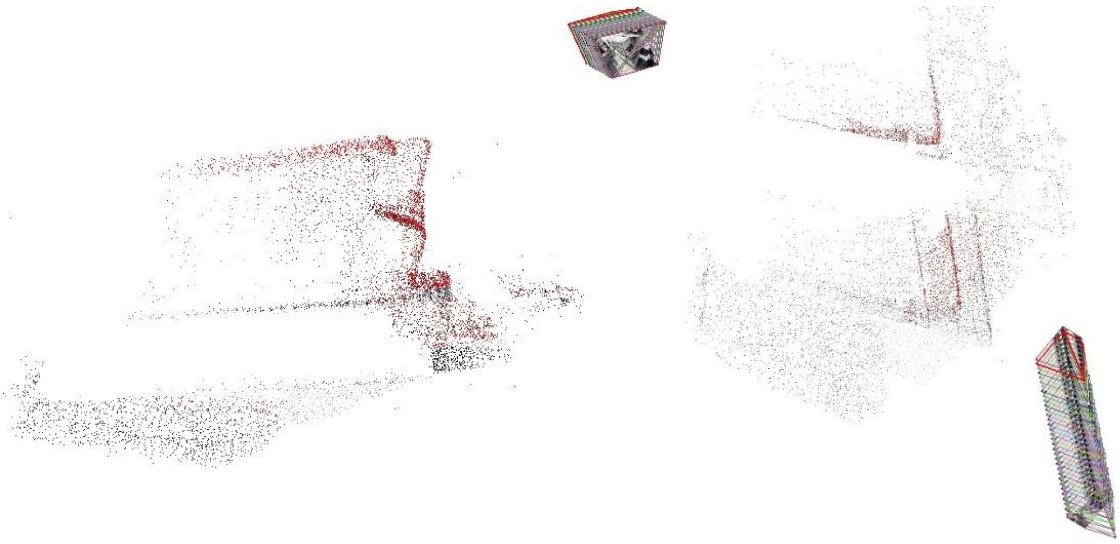


Figure 49: Sparse point cloud from VisualSFM (left: front view, right: top view)

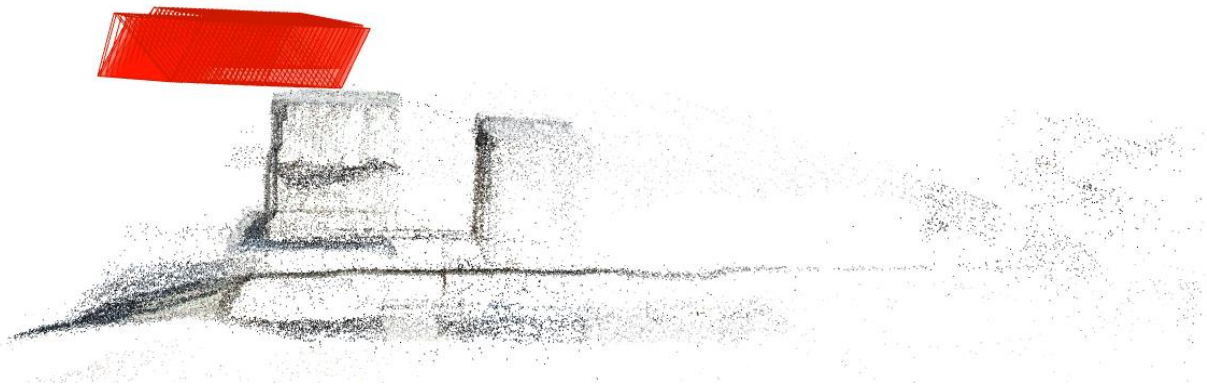


Figure 50: Right side of the sparse point cloud from COLMAP

Figure 51: Left side of the sparse point cloud from COLMAP

Figure 52:Top view of the dense point cloud from COLMAP

Figure 53: Top view of dense point cloud from Agisoft Metashape

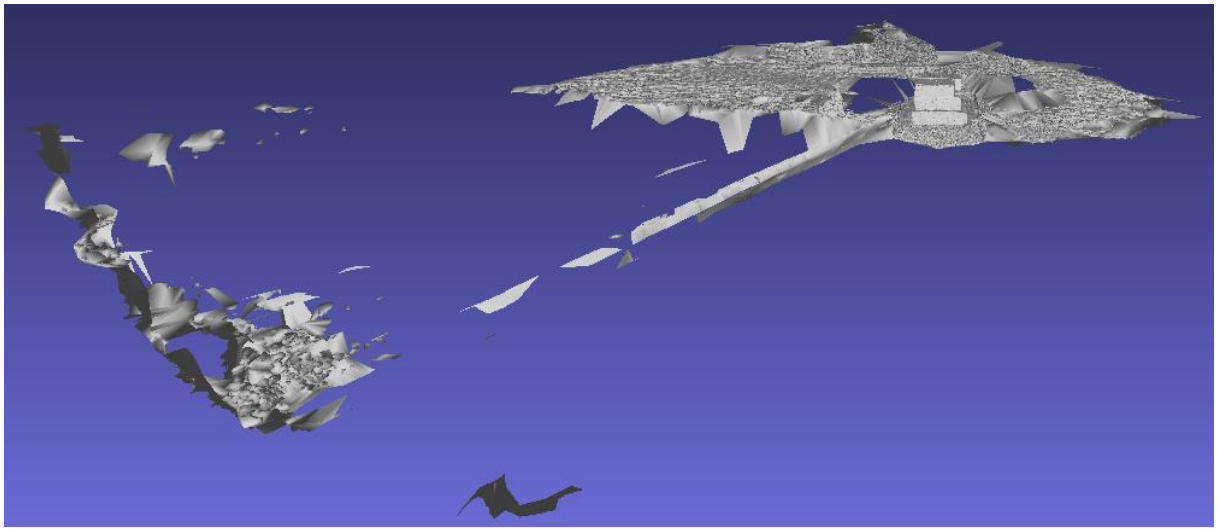


Figure 54: Front view of the Delaunay triangulation based surface reconstruction from COLMAP

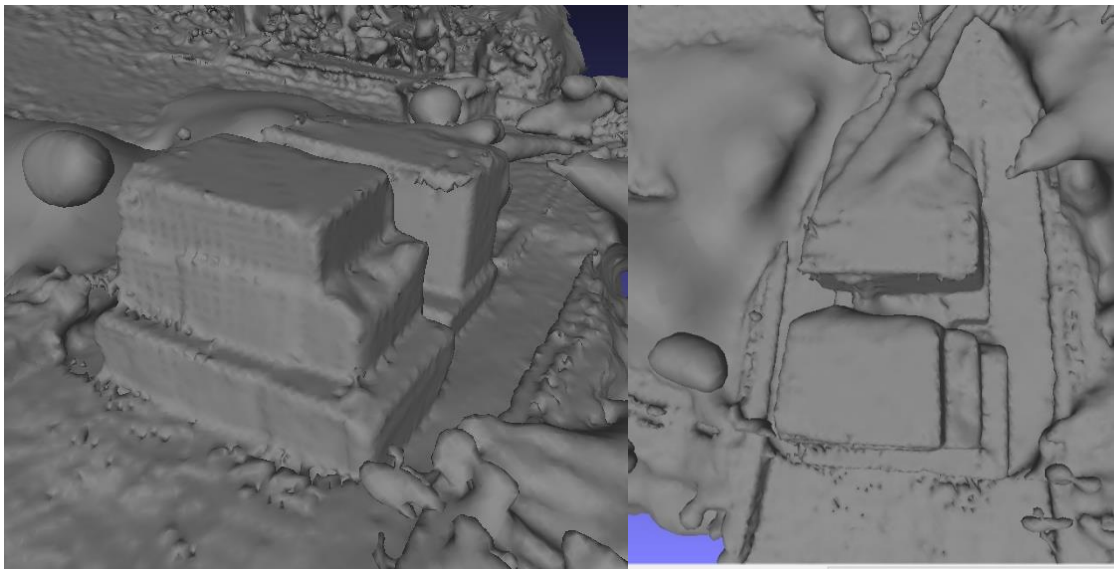


Figure 55: Screened Poisson surface reconstruction from COLMAP (left: perspective from the right side; right: top view of the limestone line)

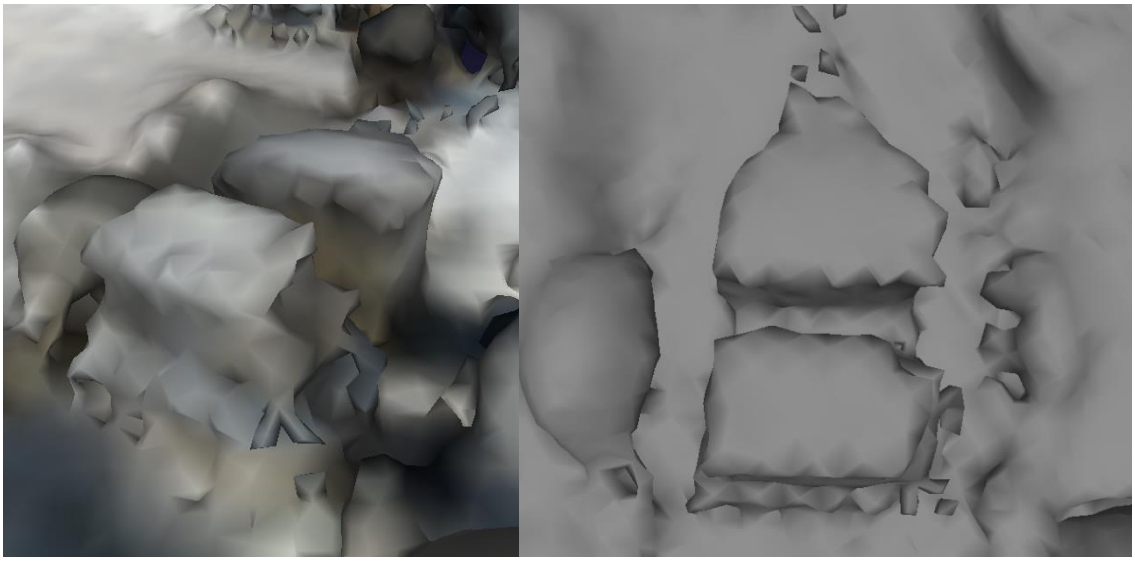


Figure 56: Screened Poisson surface reconstruction from MeshLab (left: perspective from the right side; right: top view of the limestone line)

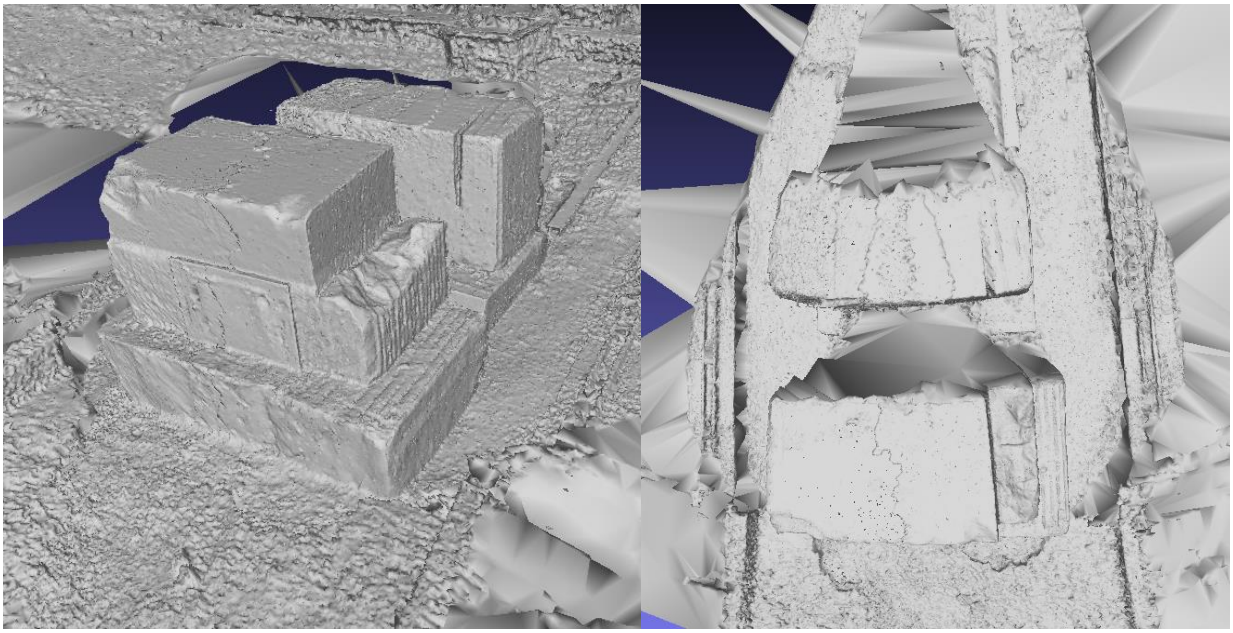


Figure 57: Delaunay triangulation based surface reconstruction from COLMAP (left: perspective from the right side; right: top view of the limestone line)

Figure 58: Top view of the textred model from Agisoft Metashape

These following tables correspond to the information portrayed in charts along with the discussions in chapter 5. Figure 32 shows a chart with the information in Table 6, as well as Figure 33 corresponds to Table 7, Figure 36 with Table 8, Figure 54 with Table 9, Figure 42 with Table 10, Figure 43 with Table 11 and Figure 44 with Table 12.

Table 6: Sparse point cloud elapse time

		VisualSFM	COLMAP	Meshroom	Agisoft Metashape
time[min]	**Features detection**	0.167	0.281	2.512	0.293
	Features matching	5.510	7.552	0.954	2.733
	Sparse point cloud reconstruction	6.683	15.005	8.013	19.443

Table 7: Number of features and sparse points

	VisualSFM	COLMAP	Meshroom	Agisoft Metashape
Overall Number of features detected	585,949	805,687	417,350	5,730,545
Number of sparse points	11,287	57,532	37,086	316,482

Table 8: Dense point cloud reconstruction comparison

	COLMAP	Meshroom	Agisoft Metashape
Dense point cloud time [min]	161.884	22.314+unknown time of fusion	47.833
Number of dense points	4,071,155	Unknown	10,155,454

Table 9: Surface reconstruction comparison

	COLMAP (Poisson)	COLMAP (Delaunay)	MeshLab (Poisson)	Meshroom (Delaunay)	Agisoft Metashape
Surface generation time [min]	0.52	23.261	0.297	25.2	7.617
Number of surface faces	760,739	1,635,633	548,284	1,428,422	1,989,573

Table 10: Textured reconstruction elapsed time

	Meshroom	Agisoft Metashape
Texture generation time [min]	4.183	9.834

Table 11: Intrinsic parameters input in Meshroom

	Intrinsic without distortion parameters	Intrinsic with distortion parameters	Intrinsic previously computed by Meshroom
Sparse reconstruction elapsed time [min.]	10.545	10.004	9.621
Number of tie points	33,578	6,602	37,124

Table 12: Intrinsic and extrinsic parameters input in Meshroom

	Extrinsic parameters	Extrinsic parameters computed by Meshroom
Sparse reconstruction elapsed time [min.]	3.725	3.553
Number of tie points	19,117	32,682

Table 13: Sparse cloud reconstruction with exhaustive image matching in Meshroom

	Exhaustive-Brute Force	Exhaustive-ANN	Exhaustive-Cascade Hashing
Sparse reconstruction elapsed time [min.]	10.525	11.297	11.030
Number of tie points	37,086	37,058	35,829

Table 14: Sparse cloud reconstruction with vocabulary tree image matching in Meshroom

	Vocabulary tree-Brute Force	Vocabulary tree-ANN	Vocabulary tree-Cascade Hashing
Sparse reconstruction elapsed time [min.]	10.529	10.378	10.209
Number of tie points	37,013	36,676	35,565

Table 15: Sparse cloud reconstruction with sequential image matching in Meshroom

	Sequential-Brute Force	Sequential-ANN	Sequential-Cascade Hashing
Sparse reconstruction elapsed time [min.]	11.084	10.973	10.782
Number of tie points	36,875	36,798	35,538

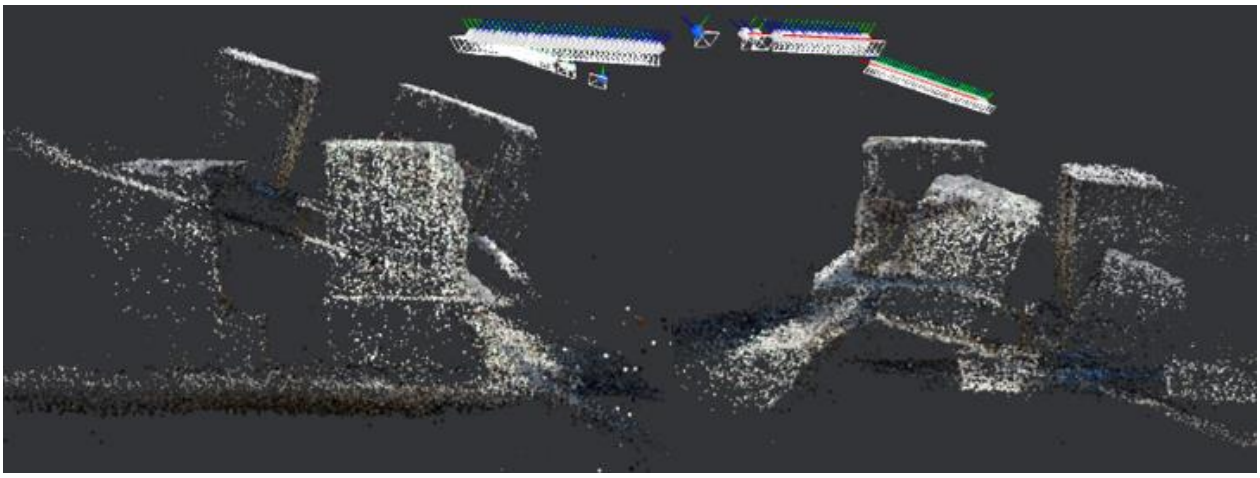


Figure 59: Sparse Cloud given intrinsic parameters (without distortion)

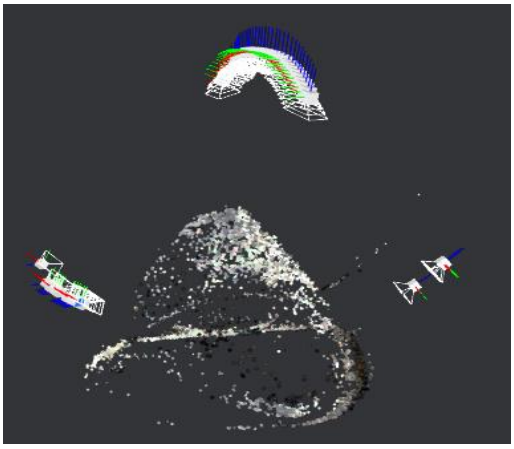


Figure 60: Sparse Cloud given intrinsic parameters (with distortion)

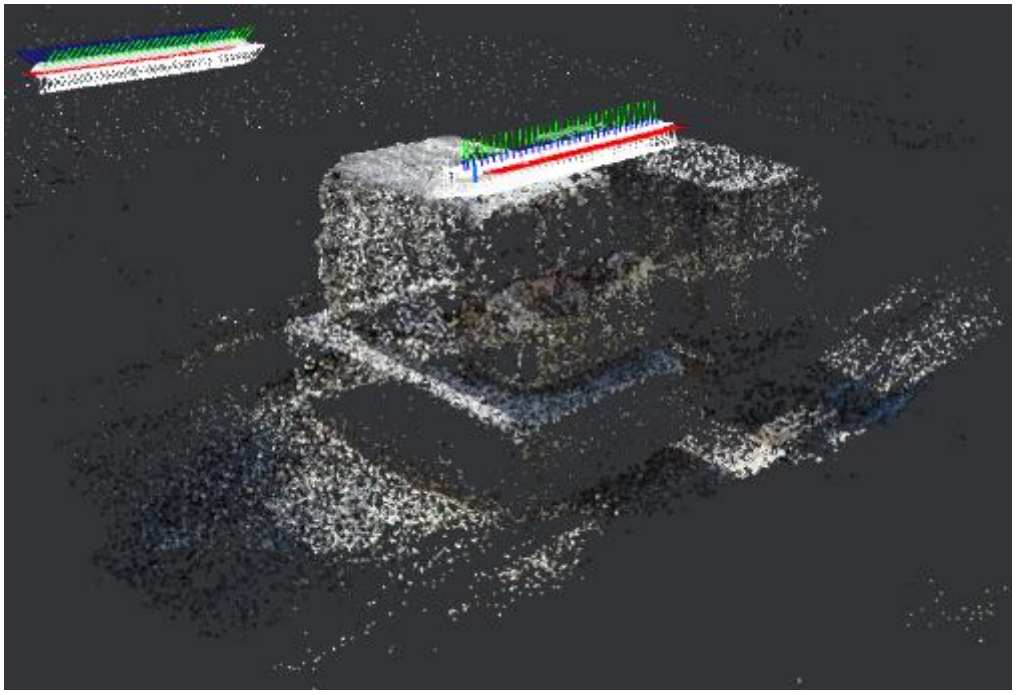


Figure 61: Sparse Cloud given intrinsic parameters from Meshroom

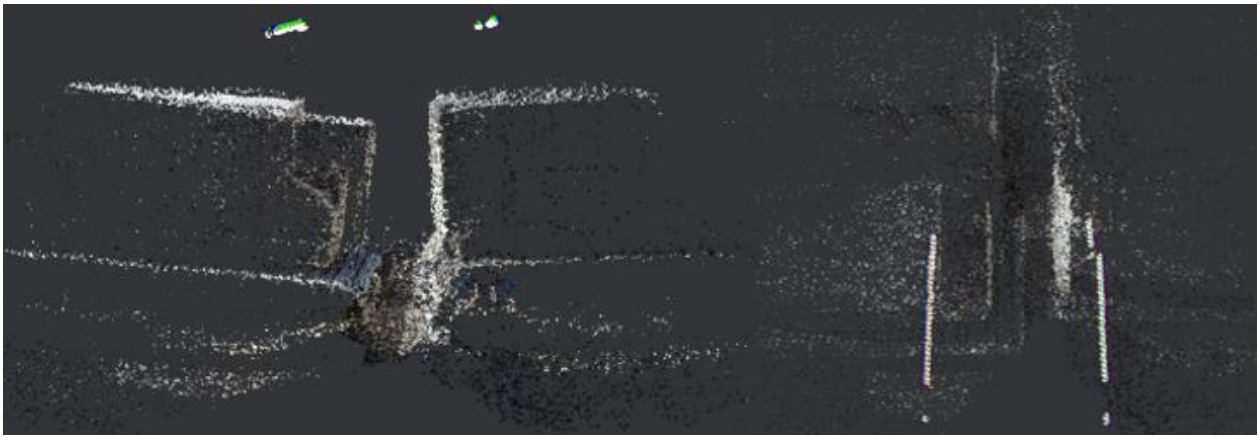


Figure 62: Sparse Cloud given extrinsic parameters (intrinsics by Meshroom)

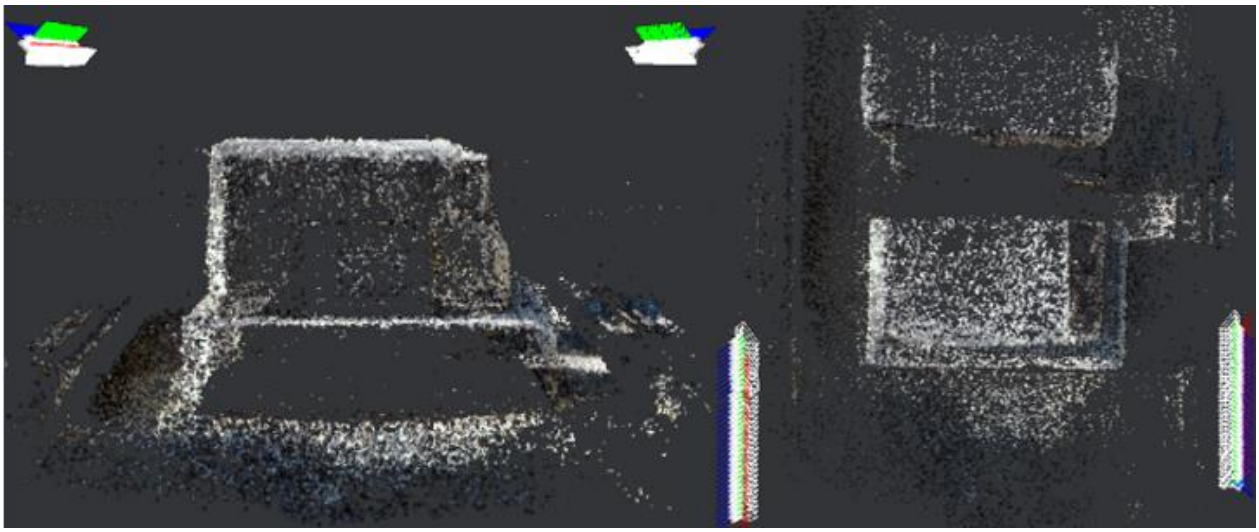


Figure 63: Sparse Cloud given intrinsic and extrinsic parameters by Meshroom