Institute for Computer Science VII
Robotics and Telematics

Master's thesis

Benchmarking structure from motion algorithms with video footage taken from a drone against laser-scanner generated 3D models

Angel Alfredo Martell

September 2017

First supervisor
Prof. Dr. Andreas Nüchter
Julius-Maximilians-Universität Würzburg

Second supervisor
Dr. Anita Enmark
Luleå University of Technology

Abstract

Structure from motion is a novel approach to generate 3D models of objects and structures. The dataset simply consists of a series of images of an object taken from different positions. The ease of the data acquisition and the wide array of available algorithms makes the technique easily accessible. The structure from motion method identifies features in all the images from the dataset, like edges with gradients in multiple directions, and tries to match these features between all the images and then computing the relative motion that the camera was subject to between any pair of images. It builds a 3D model with the correlated features. It then creates a 3D point cloud with colour information of the scanned object. There are different implementations of the structure from motion method that use different approaches to solve the feature-correlation problem between the images from the data set, different methods for detecting the features and different alternatives for sparse reconstruction and dense reconstruction as well. These differences influence variations in the final output across distinct algorithms.

This thesis benchmarked these different algorithms in accuracy and processing time. For this purpose, a terrestrial 3D laser scanner was used to scan structures and buildings to generate a ground truth reference to which the structure from motion algorithms were compared. Then a video feed from a drone with a built-in camera was captured when flying around the structure or building to generate the input for the structure from motion algorithms. Different structures are considered taking into account how rich or poor in features they are, since this impacts the result of the structure from motion algorithms. The structure from motion algorithms generated 3D point clouds, which then are analysed with a tool like CloudCompare to benchmark how similar it is to the laser scanner generated data, and the runtime was recorded for comparing it across all algorithms. Subjective analysis has also been made, such as how easy to use the algorithm is and how complete the produced model looks in comparison to the others.

In the comparison it was found that there is no absolute best algorithm, since every algorithm highlights in different aspects. There are algorithms that are able to generate a model very fast, managing to scale the execution time linearly in function of the size of their input, but at the expense of accuracy. There are also algorithms that take a long time for dense reconstruction, but generate almost complete models even in the presence of featureless surfaces, like COLMAP modified PatchMacht algorithm. The structure from motion methods are able to generate models with an accuracy of up to 3 cm when scanning a simple building, where Visual Structure from Motion and Open Multi-View Environment ranked among the most accurate. It is worth highlighting that the error in accuracy grows as the complexity of the scene increases. Finally, it was found that the structure from motion method cannot reconstruct correctly structures with reflective surfaces, as well as repetitive patterns when the images are taken from mid to close range, as the produced errors can be as high as 1 m on a large structure.

Zusammenfassung

Structure From Motion ist ein neuartiger Ansatz um 3D-Modelle von Objekten und Strukturen zu erstellen. Der Datensatz besteht aus einer Serie von Bildern eines Objektes, die aus unterschiedlichen Positionen aufgenommen sind. Die einfache Akquirierung der Daten, sowie die große Vielfalt an vorhandenen Algorithmen macht diese Technik leicht einsetzbar. Die Structure From Motion Methode identifiziert Merkmale, wie Kanten, in allen Bildern des Datensatzes und versucht die Merkmale unterschiedlicher Bilder einander zuzuordnen. Daraus wird die Relativbewegung der Kamera zwischen den jeweiligen Bildpaaren errechnet und ein 3D-Modell der korrelierten Merkmale wird erstellt. Die Methode erzeugt eine 3D-Punktwolke mit den Farbinformationen des gescannten Objekts. Es gibt verschiedene Implementierungen der Structure From Motion Methode die unterschiedliche Zugänge nutzen um das Merkmal-Korrelations-Problem zwischen den Bildern zu lösen, unterschiedliche Methoden zum Detektieren der Merkmale verwenden und verschiedene Alternativen für die Rekonstruktion der Punktwolke sowie deren Verdichtung zum 3D-Modell anwenden.

Diese Arbeit bewertet die verschiedenen Algorithmen hinsichtlich ihrer Genauigkeit und Rechenzeit. Dafür wurde ein terrestrischer 3D-Laserscanner genutzt um Strukturen und Gebäude zu vermessen und eine akkurate Referenz zu erstellen, mit der die Ergebnisse der Structure From Motion Algorithmen verglichen werden sollen. Dann wurde ein Video beim Umfliegen der Struktur oder des Gebäudes aufgenommen, wozu eine Drohne mit eingebauter Kamera genutzt wurde, um die Bilddaten für die structure of motion Algorithmen zu generieren. Es wurde unterschiedliche Strukturen betrachtet, damit berücksichtigt ist, wie die Reichhaltigkeit der Merkmale die Ergebnisse der Structure From Motion Algorithmen beeinflusst. Mit den Structure From Motion Algorithmen wurden 3D-Punktwolken erzeugt, wobei die Laufzeit zum Vergleich aufgezeichnet wurde. Zur Bewertung wie ähnlich die 3D-Punktwolken zu den vom 3D-Laserscanner generierten Daten sind wurden sie mit einem Programm, wie CloudCompare, analysiert. Eine subjektive Analyse, wie zum Beispiel hinsichtlich der Benutzerfreundlichkeit der Algorithmen und der Vollständigkeit des erstelltes Modelles im Vergleich mit den anderen, wurde ebenso durchgeführt.

Der Vergleich zeigt, dass es keinen, bezüglich aller Teilaspekte des Vergleichs, besten Algorithmus gibt, da sie sich jeder in anderen Bereichen hervorhebt. Es gibt Algorithmen die in der Lage sind ein Modell besonders schnell, aber auf Kosten der Genauigkeit, erstellen können, wobei deren Laufzeit linear zur Größe des eingegebenen Datensatzes ist. Ebenso gibt es Algorithmen die besonders lange für die Verdichtung der Punktwolke benötigen, aber dabei, trotz Flächen ohne Merkmale, fast vollständige Modelle erzeugen. Ein Beispiel hierfür ist COLMAP, ein modifizierter PatchMatch Algorithmus. Die structure from motion Methoden sind in der Lage Gebäudemodelle mit einer Genauigkeit von bis zu 3 cm zu erzeugen, wobei Visual Structure from Motion und Open Multi-View Environment zu den genauesten zählen. Erwähnenswert ist dabei, dass die Abweichungen mit zunehmender Komplexität des Gebäudes und seiner Umgebung zunehmen. Letztendlich kann ausgesagt werden, dass die structure from motion Methode Strukturen mit spiegelnden Oberflächen oder sich wiederholenden Mustern nicht korrekt rekonstruieren kann, falls die Bilder aus geringen bis mittleren Abständen aufgenommen wurden. Bei großen Gebäuden können die dabei erzeugten Abweichungen bis zu 1 m betragen.

Contents

List of Figures

Chapter 1

Introduction

3D mapping and digitalization of objects and environments has a wide variety of uses for technical and scientific applications. It is useful for accurately surveying unknown areas for topography or construction applications or also for surveying areas with non intrusive methods, which is of special interest for archaeology and geology [57]; is it used to study structures or environments of difficult or dangerous access for humans like tunnels or mines, as well on remote areas like other planets; and it is also employed by robots for navigating through their surroundings and to estimate their position as well. One special application of interest is also disaster response, where the known environment has changed and a fast rescue response is required. This response can be aided with a fast method of digitalizing the affected area, to plan and coordinate effectively the rescue forces without risking human lives with a dangerous exploration by foot.

Various techniques exist for 3D mapping and digitalization of environments. The most popular ones include laser mapping, structured light mapping and photogrammetry [8]. Laser mapping senses the environment by projecting a pulsating beam of light to the object and measuring the return time of reflected pulses to calculate the distance from the projector to the object. A laser scanner usually launches beams of light in multiple directions to generate a 2D or 3D measurement. Structured light mapping works by projecting a known pattern onto the object and recording the illuminated surface from a different angle with a camera. The distortion of the pattern due to the object structure where it is projected and the different perspective of the camera is then used to calculate a model of the object [28]. Photogrammetry techniques attempt to estimate geometric information from photographs. Two approaches exists for photogrammetry: either an architectural modelling is employed [44], where the image is decomposed into primitives (points, lines and planar polygons) and 3D information is extracted by describing simpler shapes in the image, like cubes, cylinders or pyramids; or the stereophotogrammetry approach is used for point cloud generation. This approach estimates 3D coordinates by employing measurements from two or more images of the same object taken from different perspectives, then common points are identified across the images and by the means of triangulation of the projection lines of the points, their 3D position can be determined [21].

In this work structure from motion will be studied and its implementations will be benchmarked. Structure from motion is a novel approach to stereophotogrammetry to generate 3D models of objects and structures. The dataset simply consists of a series of images of an object

taken from different positions with a camera with known intrinsic parameters (focal length, sensor size, pixel size, aspect ratio), although they can be estimated as well. The ease of the data acquisition and the wide array of available algorithms makes the technique easily accessible. The structure from motion method identifies features in all the images from the dataset, like edges with gradients in multiple directions, and tries to match these features between all the images and then computing the relative motion that the camera was subject to between any pair of images with matches between them. It builds a 3D model with the correlated features when the camera positions are determined and the features triangulated. Then the algorithms perform a dense 3D reconstruction that creates a point cloud with color information of the scanned object. These algorithms, both freely available and others with commercial licenses, use different approaches to solve the feature-correlation problem between the images from the data set, different methods for detecting the features and different alternatives for dense reconstruction and triangulation as well. These differences influence variations in the final output across distinct algorithms and influence the execution time of the complete process.

This work will benchmark some of the available structure from motion algorithms, in both accuracy and processing time, trying to include the most popular for image reconstruction. For this purpose, a commercially available terrestrial 3D laser scanner will be used to scan structures and buildings to generate a ground truth reference to which the structure from motion algorithms will be compared against. Then a video feed from an amateur drone with a builtin camera will be captured when flying around the structure or building to generate the input dataset for the structure from motion algorithms. Different structures will be considered taking into account how rich or poor in features they are, since this impacts the result of the structure from motion algorithms. For example, a building with single-color flat walls will produce much less features than a building with brick walls. The structure from motion algorithms will generate a 3D point cloud, which then will be analyzed with a specialized tool like CloudCompare to benchmark how similar it is to the laser scanner generated data, and the runtime will be recorded for comparing it across all algorithms. It will also be discussed in a subjective fashion the usability of these algorithms: how easy or complicated it is to install them in a Linux system, to operate them and to export the results to a point cloud; and as well how the produced models look in completeness and noise.

1.1 Motivation and objectives

Structure from motion is a very new technique into the photogrammetry family of solutions, with algorithms being not older than ten years and is still an active topic of development. Therefore not much work has been done in elaborating an exhaustive comparison across the most popular implementations of the structure from motion technique. For benchmarking these structure from motion implementations, this work looks for fulfilling the following.

- To evaluate how the different structure from motion implementations behaves with models with known limitations, such as texture-less surfaces, reflective surfaces and changing structures, like smoke.
- To compare the execution time that the different implementations take in order to generate

a 3D model, given the same input.

- To assert the accuracy of the structure from motion techniques by comparing it to a proven accurate frame of reference, such as laser scanner produced models.
- To describe subjectively the installation and usability of the different implementations of the structure from motion technique.
- To describe subjectively the generated models: how sharp or noisy do the structures look, how accurate are the colours, what kind of artefacts are generated if any.

It is also of interest to evaluate if a structure from motion approach with video taken from a drone would be useful for disaster response coordination. When a complex catastrophic event occurs, like an earthquake or a landslide that causes the collapse of buildings, big fires in industrial plants or in the forests, or also large crashes of vehicles in a highway; the optimal use of the first minutes after the event are vital to find survivors inside damaged structures or trapped by fires. This process can be aided if the rescue team has knowledge of the terrain where the search is to be conducted, as to plan where to focus the search efforts more effectively, as exploration by foot can be too difficult or even dangerous. Therefore this work also looks for identifying which implementation, if any, is best suited for a fast and accurate generation of a map.

1.2 Previous work

Since structure from motion is a very recent and constantly changing topic, very few research has been made in asserting the accuracy of different algorithms, and compare which ones perform better and under what circumstances this occurs. There are a number of articles where the accuracy of structure from motion techniques is tested, but these works rarely compare different algorithms or a variety of scenarios.

Most comparisons between different structure from motion algorithms have been made in close range measurements of small objects, like ornaments or sculptures. Nikolov et al. [40] tested six different commercial structure from motion software packages (including PhotoScan by Agisoft), and digitalized six different objects. They chose objects with varying characteristics, like featureless surfaces, repetitive patterns and glossiness. The produced models are compared to an undisclosed ground truth reference, and they found that most algorithms provided sub-millimetre accuracy for close range photos, and that PhotoScan in particular was consistent with the results taken from different lighting conditions but failed to resolve finer detail, as it over-smoothed the surfaces.

Jaud et al. [23] compared PhotoScan against MicMac, deveolped by the French National Institute of Geographic and Forestry Information, by building a model of a landslide on the Réunion island with images taken from an hexacopter. The tested datasets consist of between 109 to 120 images, and reference GPS markers are used for correctly estimating the scale of the generated models. They compared the constructed models against laser scanner measurements for assessing their accuracy. They found that both of them achieve in average errors from 3 to 4 cm of accuracy when compared to the laser, but in steep slopes this can rise up to 17 cm. They also found that PhotoScan provides a cleaner model for rugged terrain.

Another work that uses a reference frame from laser scanners was implemented by Pangagiotidis et al. [42]. They use again PhotoScan for digitalizing tree stems as a cheaper alternative for gathering forestry metrics, and they found errors of up to 11 cm in the digitalized models when compared to the laser data. Insead of using a Gaussian distribution for error description, they used a Weibull distribution that fits this kind of error data better, as errors are unsigned.

Work from Mancini et al. [36] and Genchi et al. [17] has also used drones for capturing data and generating models for topography, and later comparing them to laser measurements. In the first work they used as well PhotoScan and when compared to laser data, they found errors of up to 19 cm. The second work used Visual Structure from Motion, and found an accuracy of 7 cm.

1.3 Outline

This work will aim to give the reader a brief understanding on what structure from motion is and how does it work, what solution exists for the technique and how and why the structures used in this work where selected.

Therefore on the second chapter the reader will be introduced to the complete structure from motion pipeline. The process on how a camera projects light into its sensor will be explained, as the inverse process is required for understanding how the triangulation steps work. Feature detection and matching will also be discussed, as well as how structure from motion generates the point cloud. It is important to highlight that the general idea of each step in the structure from motion pipeline will be explained, sometimes with the aid of a particular implementation as to provide a more detailed view of the specifics of the expected functionality to the reader; but by no means is this work an exhaustive documentation of all the existing implementation for this steps.

In the third chapter it will be enumerated what algorithms will be benchmarked in this work and it will be discussed how they implement each step in the structure from motion pipeline, in accordance with their respective documentation. The most remarkable improvements made by each algorithm at the time of their release will be highlighted.

In the fourth chapter the methodology of how this work aims to conduct the benchmarking task is explained. It is also stated what objects will be scanned and why are they chosen. The equipment employed for this work is also described, and how the metrics for the benchmark are obtained and interpreted.

In the fifth chapter the generated models are presented. The performance counters used for the final analysis are also quantified in this section, such as point count, execution time and error distribution. Subjective parameters are also discussed, such as the quality of the generated models. In the final chapter, an overview analysis of the results is made and conclusions are drawn. Future work is proposed, as to what improvements can be made for both structure from motion algorithms as well as future benchmarking works.

In the final pages of this work the reader will find Appendix A and B. Appendix A contains different views of each generated point cloud by each pipeline in full colour, and in Appendix B the same views are shown with a colour map that is bound to the magnitude of the error between the generated model and the frame of reference obtained by the laser scanner.

Chapter 2

Structure from Motion

The ability to recover 3D information from 2D images is of high interest in the field of robotics and image processing in general. This operation is unfortunately not possible by using a single image due to the fact that depth information cannot be extracted from it without additional information. But using different views of the same scene, this information can be reconstructed by means of triangulation. Structure from motion is a photogrammetric technique to compute simultaneously camera positions and projections from multiple views of a scene, to produce a 3D model of the scene using only correspondences between the different views. In a broad sense these correspondences are found when tracking common features between the views, which can be detected as edges with colour gradients on multiple directions. When matching multiple views, tracking the feature trajectories over time is used to compute their 3D position, as well as the camera motion. Two main alternatives exist for approaching this reconstruction step: incremental structure from motion, that solves the camera poses one by one and adds them into the reconstruction, and global structure from motion, that attempts to solve all of them at the same time. After the reconstruction a sparse 3D point cloud is obtained, which is refined by applying a bundle adjustment that optimizes the initial projection matrices of the views by the means of minimizing a cost function [44]. As a final step, with the 3D pose identified for the features in every view, the depth information can be computed for pixels in nearby regions of feature points. This data is used to produce a denser and colourized 3D point cloud as the output of the structure from motion algorithm. Figure 2.1 shows an overview of the data flow of the structure from motion technique.

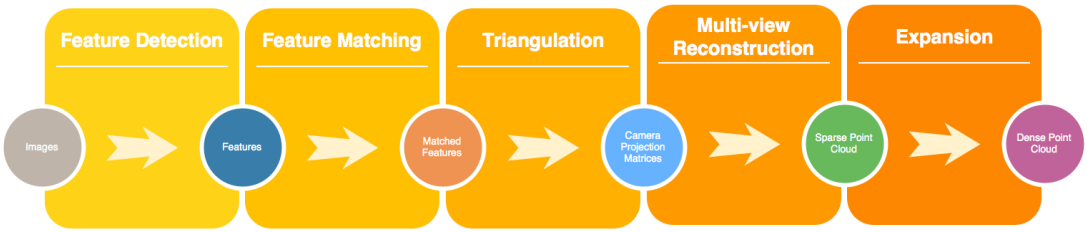


Figure 2.1: Overall process flow of an structure from motion algorithm.

2.1 Camera image projection

For being able to reconstruct 3D data from a 2D view, the inverse process in which the camera projects the information of a 3D world into a 2D image must be understood first. The simplest way of modelling this process is through the pinhole camera projection model, in which any point in 3D space is mapped to a point in the image projection plane by the means of a straight lines that connects both points but also goes through a fixed point in space: the centre of projection or camera centre C. This is called the central projection camera model (Figure 2.2)[21].

2.1.1 Camera projection matrix

Central projection simply becomes then a mapping from 3D space to a 2D plane that can be represented with a transformation matrix. Instead of using the real image plane where the image physically projects on the camera sensor, a virtual image plane will be used to derivate this matrix for simplicity, which lies in front of the camera centre at a distance f, the focal length of the camera [24]. If some other simplifications are taken, as considering the camera centre as the origin of the coordinate system and considering the plane $Z = f$ as the image plane, the mapping for any point in space $\mathbf{X} = (X, Y, Z)^T$ is projected into the point $\mathbf{x} = (fX/Z, fY/Z, f)^T$ in the image plane (Figure 2.3). Ignoring the third space coordinate which is a constant, a mapping from Euclidean 3-space $\mathbb{R}^3$ to Euclidean 2-space $\mathbb{R}^2$ is obtained. Using homogeneous coordinates for matrix notation, a camera projection matrix can be defined for the transformation in matrix multiplication form $\mathbf{x} = P\mathbf{X}$ where $\mathbf{x}$ is the resulting point in the image plane, $\mathbf{X}$ the original point in 3D space and P the projection matrix. From the definition above, the projection matrix can be inferred as follows

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.1}$$

$$P = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.2}$$

This assumes the origin of the image plane is at the principal point p (Figure 2.3), but this is not the case in practice. For that the mapping must be defined to take into account the offset of the principal point in reference to the bottom left corner of the sensor of the camera, which functions as the origin in the practical case (Figure 2.4). The mapping then becomes $(X, Y, Z)^T \mapsto (fX/Z + p_x, fY/Z + p_y)^T$ and can be defined as

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{2.3}$$

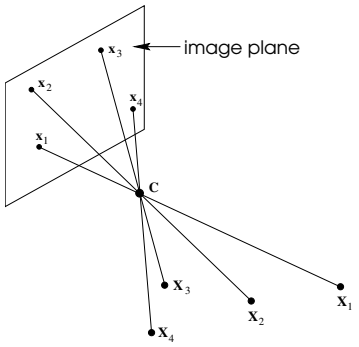


Figure 2.2: Central projection camera model [21, p. 8].

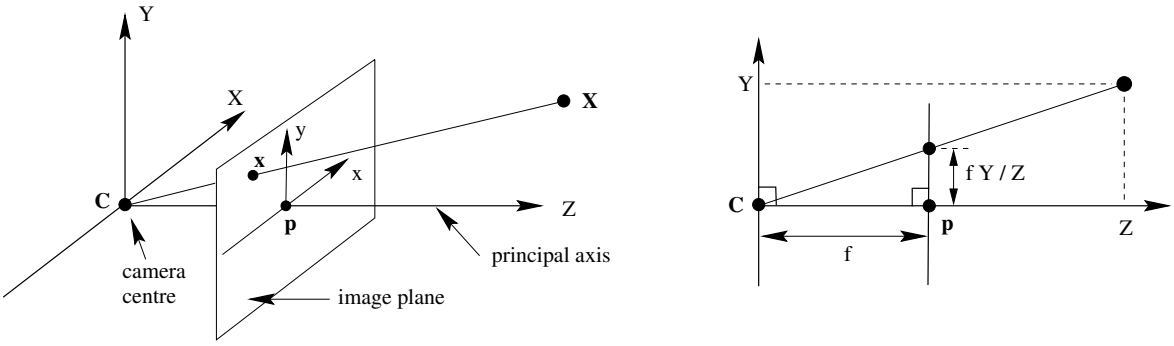


Figure 2.3: Camera projection geometry using a virtual image plane [21, p. 154].

$$K = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.4}$$

This matrix K is called the camera calibration matrix, and for the idealized camera model does not contemplate any kind of distortion for now. So far it has been assumed that the camera centre is located at the origin of the frame of reference, but the camera position and orientation in the world coordinate frame must be taken into account as well, as working with multiple cameras is the final goal. The two coordinate frames, camera and world coordinates, are related only by one translation and one rotation operation (Figure 2.5). The vector $\tilde{\mathbf{X}}$ represents the coordinates of a 3D point in this world coordinate system and the vector $\tilde{\mathbf{X}}_{cam}$ represents the same point but in the camera coordinate system, then their relationship can be expressed as $\tilde{\mathbf{X}}_{cam} = R(\tilde{\mathbf{X}} - \tilde{\mathbf{C}})$, where $\tilde{\mathbf{C}}$ denotes the camera position in the world coordinate frame and R is a rotation matrix representing the orientation of the camera coordinate frame. This can be rewritten in homogeneous coordinates as follows.

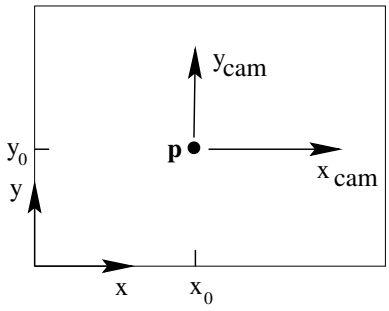


Figure 2.4: Coordinate systems of the image plane (x, y) and the camera (x_{cam}, y_{cam}) and their relation with the principal point p [21, p. 155].

$$\tilde{\mathbf{X}}_{cam} = \begin{bmatrix} R & -R\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \tilde{\mathbf{X}} \tag{2.5}$$

Finally, the complete transformation from a point in world coordinates to a 2D point in the image plane of the camera $\tilde{\mathbf{x}}$ can be obtained by pre-multiplying $\tilde{\mathbf{X}}_{cam}$ by K, which is given by the expression

$$\tilde{\mathbf{x}} = K \begin{bmatrix} R & -R\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \tilde{\mathbf{X}} \tag{2.6}$$

Of which the transformation matrix between $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{X}}$ can be grouped as

$$\mathbf{P} = K \begin{bmatrix} R & -R\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \tag{2.7}$$

$\mathbf{P}$ is known as the pinhole camera homogeneous projection matrix. It is worth noting that this final camera matrix has 9 degrees of freedom: 3 for K (f, p_x, p_y), 3 for R and 3 for $\tilde{\mathbf{C}}$. The parameters in K are called the internal camera parameters and the parameters in R and $\tilde{\mathbf{C}}$ (position and orientation) are called the external parameters [21, p. 156].

It has been assumed so far that the camera sensor consist of square pixels but this is not the case for CCD sensors in modern cameras. These sensors use different techniques, like a Bayer filter, to capture colour as three separate channels of light intensity [6]. This can make a pixel (the combination of the three channels) not square in shape in the physical sensor and thus K must be redefined to compensate for this change in form factor. The focal length of the camera can be defined then in terms of pixel dimensions, and it has to be defined for both the horizontal and vertical axis. Considering m_x as the number of pixels per unit of length in the x axis of the sensor, and m_y similarly for the y axis of the sensor, it can be defined α_x as the focal length in terms of pixel dimensions for the x axis as $\alpha_x = fm_x$ and α_y as the focal length in terms of pixel dimensions for the y axis as $\alpha_y = fm_y$. Then K can be defined as follows

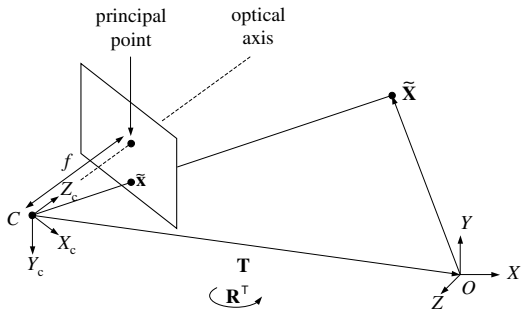


Figure 2.5: Illustration of the final camera projection matrix [44, p. 4].

$$K = \begin{bmatrix} \alpha_x & 0 & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.8}$$

The change in the displacement or the principal point p has to be commented as well, since now all measurements are given in pixel dimensions Therefore, x_0 is defined as the x coordinate of the principal point in the image plane in terms of pixel dimension as $x_0 = m_x p_x$ and similarly for $y_0 = m_y p_y$. This change overall increases the degrees of freedom to 10.

2.1.2 Radial distortion

Camera lenses are not perfect, so distortions have to be taken into account for a correct reconstruction of the 3D data encoded in the image. The most common type of distortion in a lens is the radial distortion (Figure 2.6). Under this type of distortion, points are displaced in a radial direction from or to the centre of distortion. One approach to simplify this problem is assuming that the centre of distortion is the principal point. Considering the point $\mathbf{x} = (x, y)^T$ as the point in the image plane in reference to the principal point (i.e. $\mathbf{x} = K^{-1}\tilde{\mathbf{x}}$), the radial distortion is corrected by

$$\hat{x} = x + L(r)x \tag{2.9}$$

$$\hat{y} = y + L(r)y \tag{2.10}$$

Where $\hat{x}$ and $\hat{y}$ are the corrected values and the distortion function $L(r)$ can be approximated by the expression $L(r) \approx k_1 r^2 + k_2 r^4$ and $r^2 = x^2 + y^2$. The values of k_1 and k_2 are also considered to be intrinsic camera values [44, p. 5]. This is one possibility to model radial distortion, as there exist more models that can be applied to describe this effect mathematically.

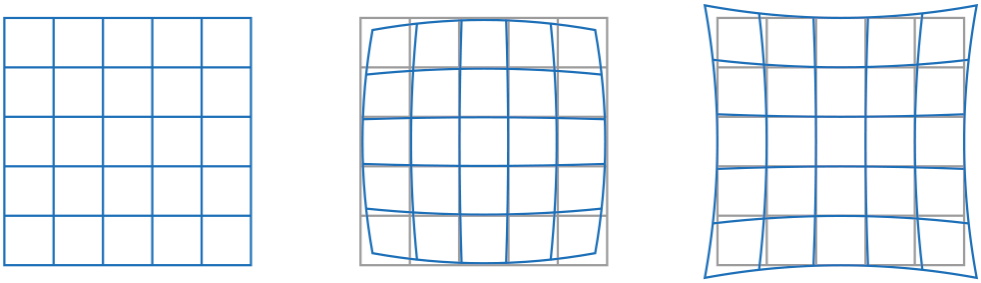


Figure 2.6: Radial distortion of a lens. Left: undistorted image. Centre: image with positive radial distortion or barrel distortion Right: image with negative radial distortion or pincushion distortion.

2.2 Feature detection

The first step in a structure from motion reconstruction pipeline is detecting features in a view that can be later matched across multiple images to infer the camera position and depth information of the views. For this purpose, a method is required to detect such features that is invariant to image scaling and rotation and also to changes in illumination and camera point of view; to maintain a consistent and stable feature tracking across multiple views. One of the most popular algorithms for this purpose utilized in multiple structure from motion packages is the scale-invariant feature transform (SIFT) [32]. It will be presented in this section to illustrate the process of a feature detection algorithm, but it is by no means the only alternative to solve this problem.

The SIFT algorithm starts with an extrema detection in the scale-space of the image [58]. The scale-space of an image is defined as a function $L(x, y, \sigma)$ that results from applying the convolution of a Gaussian filter $G(x, y, \sigma)$ to the input image $I(x, y)$ [33] as shown below

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.11}$$

Where $G(x, y, \sigma)$ is the 2D Gaussian filter defined as

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{2.12}$$

According to the author [33], stable keypoint locations are detected efficiently instead in the difference of Gaussian function convolved with the image which is computed from the difference of two nearby scales separated by a constant factor k as follows

$$D(x, y, \sigma) = [G(x, y, k\sigma) - G(x, y, \sigma)] * I(x, y) \tag{2.13}$$

Which can be distributed according to the properties of the convolution properties, resulting in the substraction of the image smoothed k times minus the image smoothed one time, as indicated below

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2.14}$$

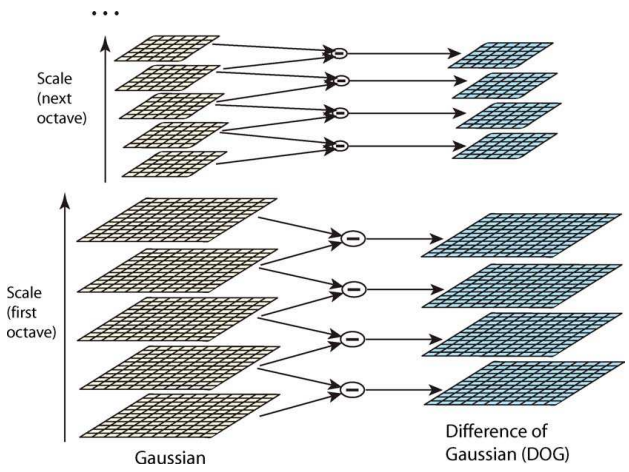


Figure 2.7: Scale-space and difference of Gaussian function generation [33, p. 95].

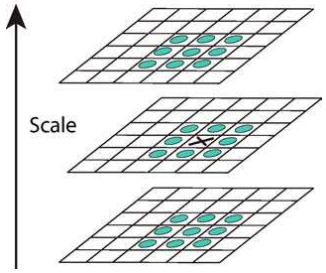


Figure 2.8: Local minima and maxima selection of a point compared to its neighbours in the current image, the image below and the image above. The current point is marked with an X and its neighbours are marked with green circles [33, p. 95].

In practice, the image is incrementally convolved with Gaussian filters to produce images separated in scale-space by a constant factor of k, and in between every application of the Gaussian filter, the original image is substracted from the resulting one to generate the difference of Gaussian function $D(x, y, \sigma)$ (Figure 2.7). Every time σ doubles its value, an octave is generated. After an octave has been processed, the image is resampled by taking every second pixel in each row and column, effectively reducing its size by half, then the process is repeated.

For detecting the local maxima and minima (extrema) of the function $D(x, y, \sigma)$ each point in the function is compared to its eight near neighbours in the current image and also to the nine neighbours in the image below and above. This point is selected as a keypoint only if it has the greatest or smallest value than all of its neighbours (Figure 2.8).

When the keypoints are selected, a filtering process is applied fitting the nearby data for location, scale and ratio of the principal curvatures. This process allows points to be rejected when they have low contrast or are poorly localized along an edge [33]. The approach consists of applying a Taylor expansion of the scale-space function $D(x, y, \sigma)$, shifted to make the sample point its origin [7] as follows

$$D(\mathbf{x}) = D + \frac{\delta D^T}{\delta \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\delta^2 D}{\delta \mathbf{x}^2}\mathbf{x} \tag{2.15}$$

Where D and its derivatives are evaluated at the sample point and $\mathbf{x} = (x, y, \sigma)^T$ is the offset from this point. The location of the extremum $\hat{\mathbf{x}}$ is found by setting the derivative of this function equal to zero, as indicated below

$$\hat{\mathbf{x}} = -\frac{\delta^2 D^{-1}}{\delta \mathbf{x}^2}\frac{\delta D}{\delta \mathbf{x}} \tag{2.16}$$

If the location of the extremum $\hat{\mathbf{x}}$ is greater than 0.5 in any dimension it means that the extremum lies closer to a different sample point and the sample point is changed instead and the process repeated. If is not the case, then the offset $\hat{\mathbf{x}}$ is added to the sample point to interpolate the estimation of the location of the extremum. Also, the value of the function at the extremum offset $D(\hat{\mathbf{x}})$ can be evaluated to reject unstable extrema with low contrast.

This algorithm will have a strong response along edges in the image. Therefore removing the keypoints along this edges is important to increase the stability of the algorithm. A poorly defined peak can be found when the principal curvature is much larger across an edge than the principal curvature along it, and thus the point must be removed. Finding these principal curvatures can be simplified by computing the second order Hessian matrix $\mathbf{H}$ since the eigenvalues of $\mathbf{H}$ are proportional to the principal curvatures of D. $\mathbf{H}$ is defined as

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{2.17}$$

The computation of the eigenvalues is not required, since only the ratio r between them is of interest, where $r = \alpha/\beta$ with α and β being the eigenvalues. The sum of the eigenvalues is given by the trace of $\mathbf{H}$ and the product is given by the determinant of $\mathbf{H}$. It can be shown that

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} = \frac{(r+1)^2}{r} \tag{2.18}$$

Which is minimum when the eigenvalues are equal (i.e. $r = 1$). Thus, the larger the absolute difference between both eigenvalues, and in consequence, the larger the absolute difference between both principal curvatures, the larger the ratio becomes. Let r_{th} be the threshold value for the desired maximum ratio between both principal curvatures r, deciding if the point is not a poorly defined peak along the edge is then just a matter of evaluating if

$$\frac{Tr(\mathbf{H})^2}{Det(\mathbf{H})} < \frac{(r_{th}+1)^2}{r_{th}} \tag{2.19}$$

All these stages of keypoint filtering (Figure 2.9) lead to a constant recognition of features across multiple views of the same object that can now be matched across multiple perspectives to reconstruct 3D information from the scene.

The SIFT algorithm also assigns a value of scale and orientation for each feature, which is based on the strength and direction of the gradient where they are located in the image scale-space (Figure 2.10). These values can be used in later stages to compute a local image

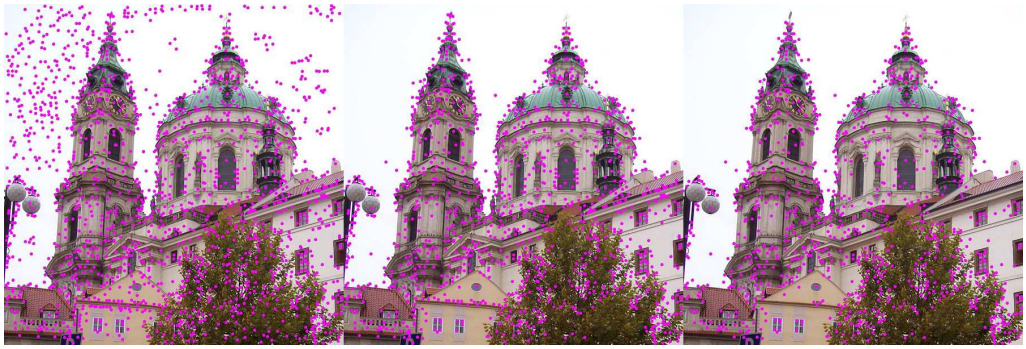


Figure 2.9: Sample SIFT feature detection on an image. Left: original detection of keypoints. Centre: Features after discarding low contrast keypoints. Right: Features after discarding keypoints along edges [34, Adapted].

Figure 2.10: Resulting visualization of SIFT features, with their corresponding magnitude and direction represented as arrows [33, Adapted p. 98].

descriptor, which is helpful for finding correspondences between the features of different images. For each feature in the scale-space function of the image $L(x, y)$, the gradient magnitude $m(x, y)$ and its orientation $\theta(x, y)$ are computed using pixel differences as indicated below

$$m(x,y) = \sqrt{\Big[L(x+1,y) - L(x-1,y)\Big]^2 + \Big[L(x,y+1) - L(x,y-1)\Big]^2} \tag{2.20}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}\right) \tag{2.21}$$

2.3 Feature matching

Feature matching, that is, finding the corresponding points in different images of the same object, has proven to be a particularly complicated problem to solve automatically [25]. This

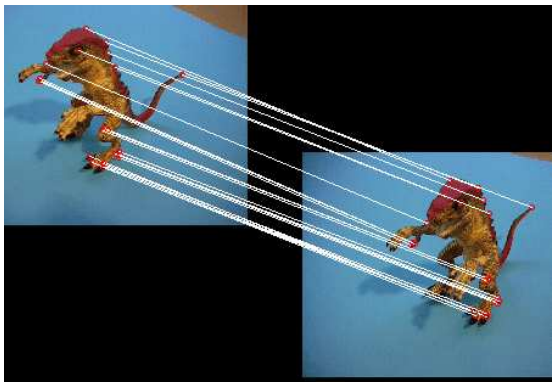


Figure 2.11: Visualization of the output of a feature matching algorithm between two images with different perspectives of the same object, showing the detected correspondences between the features of the two images [2, p. 7].

is due to the fact that two images may exhibit substantial change of scale of the photographed object, different angles or perspectives or even degrees of occlusion.

A possible approach for detecting correspondences in features between different images [2] is to assume that small local patches of surface in the object are subject to only affine transformations (i.e. translation, scale, rotation, skew). It is important to clarify once more that this is not the only solution to this problem, but the one used in this section to illustrate the process of feature correspondence.

The first step in this algorithm is to examine a region in the image centred in a keypoint and calculate a set of characterization values for that region. This involves calculating affine invariants for stretch, skew, photometric intensity changes and rotation effects, using shape-adapted texture descriptors [30]. The output of this algorithm is a feature vector containing 43 invariants describing an image patch for an RGB image.

For matching the patches across images, a Mahalanobis distance metric [35] is used since the features are sufficiently well characterised to use a simple matching scheme. Given two feature vectors $v^{(i)}$ and $w^{(j)}$ and a distance measure $d(v, w)$, the matching algorithm first calculates the distance matrix $m_{i,j} = d(v^{(i)}, w^{(j)})$ between the pairs of features across the two images. For identifying potential matches (i, j), a feature i in the first image is selected in such a way that is the closest one to feature j in the second image, and vice versa. These potential matches are scored using an ambiguity measure [9], that measures the relative distance between the two matching features and the next closest distance between one of the matched pair and any other feature. Finally, unambiguous matches are chosen or the best n matches according to the score (Figure 2.11).

2.4 Triangulation

Previous work has shown that a fundamental matrix that relates a pair of calibrated views can be estimated from at least eight correspondences between them, and how this matrix can

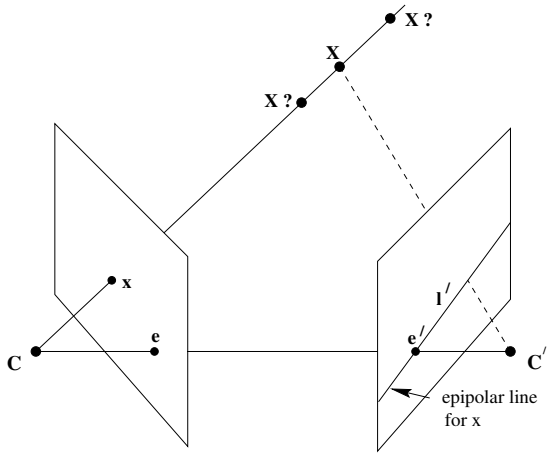


Figure 2.12: Epipolar geometry of two different views of the same point **X**. The camera centres are denoted by C and C' for the first and second image respectively. The epipolar plane is the plane intercepting the points C, C' and **X** [21, p. 240].

then be decomposed to estimate the relative camera orientation and position between the views [31]. This matrix can be derived by the means of epipolar geometry. With the relative camera position and orientation estimated, the projection matrices of the cameras can be computed by decomposing the fundamental matrix. Finally, the 3D points can be triangulated by computing the intersection of the back-projected rays of the image points between at least two images.

2.4.1 Epipolar geometry

The corresponding projections of a 3D point in two different images is constrained by the epipolar line l' (Figure 2.12). Given a camera with optical centre C', the epipolar line is defined as the projection of the ray going from the optical centre C of another image through the image point $\mathbf{x}$ on the image plane of C'. The plane that intercepts the optical centres C and C' of both images and the image point $\mathbf{x}$ on the first image plane is called the epipolar plane. The projection of the other optical centre in each image is called the epipole e and e' in the cameras C and C' respectively. The essential matrix $\mathbf{E}$ describes the epipolar constraint algebraically. Given a point $\mathbf{X}'$ in the coordinate system of camera C', the position $\mathbf{X}$ in the coordinate system of camera C is given by the expression

$$\mathbf{X} = \mathbf{R}\mathbf{X}' + \mathbf{T} \tag{2.22}$$

Where $\mathbf{R}$ is a 3×3 rotation matrix as $\mathbf{T}$ is a translation vector. If every side of the equation is pre-multiplied by $\mathbf{X}^T[\mathbf{T}]_\times$ it gives that

$$\mathbf{X}^T[\mathbf{T}]_\times\mathbf{R}\mathbf{X}' = \mathbf{X}^T\mathbf{E}\mathbf{X}' = 0 \tag{2.23}$$

Where $\mathbf{E}$ is the 3×3 matrix defined as $\mathbf{E} = [\mathbf{T}]_\times\mathbf{R}$ and $[\mathbf{T}]_\times$ is the skew-symmetric 3×3 matrix for the vector $\mathbf{T} = [t_x\ t_y\ t_z]^T$ defined as [21]

$$[\mathbf{T}]_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \tag{2.24}$$

Equation 2.23 is also valid for the transformation between points $\mathbf{x}$ and the projection of point $\mathbf{X}$ in the second image plane, $\mathbf{x}'$. This is called the epipolar constraint [44], as defined below for points in the image plane

$$\mathbf{x}^T \mathbf{E} \mathbf{x}' = 0 \tag{2.25}$$

2.4.2 Fundamental matrix

In equation 2.4, the camera calibration matrix K was defined, which relates points in 3D space $\mathbf{X}$ to a pixel position in the image plane $\mathbf{x}$. Its inverse K^{-1} may be used to describe the opposite relation, 3D point given pixel position:

$$\mathbf{X} \sim K^{-1}\mathbf{x} \tag{2.26}$$

Replacing into the epipolar constraint (see equation 2.23), it can be rewritten as

$$(K^{-1}\mathbf{x})^T \mathbf{E} (K'^{-1}\mathbf{x}') = 0$$

$$\mathbf{x}^T (K^{-1}\mathbf{E}K'^{-1})\mathbf{x}' = 0$$

$$\mathbf{x}^T \mathbf{F} \mathbf{x}' = 0 \tag{2.27}$$

Where $\mathbf{F} \sim K^{-1}\mathbf{E}K'^{-1}$ is called the fundamental matrix [11]. It is a 3×3 matrix with a rank of 2. Given 8 or more correspondences between the images, it can be estimated linearly. Let $\mathbf{x}_i$ and $\mathbf{x}_i{}'$ be two points in two different image planes, for which a correspondence has been detected. It is possible to decompose these points and the fundamental matrix into its elements as in the following

$$\begin{bmatrix} x_i' & y_i' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0 \tag{2.28}$$

For a set of n correspondence, these constraints can be arranged in a set of linear equations for the unknown 9 elements of the fundamental matrix

$$\begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0 \tag{2.29}$$

If at least $n \geq 8$ correspondences are given, a least squares solution can be found for the linear system of equations. A unique solution is obtained unless all the points lie on a plane [12]. When estimated, the fundamental matrix can be decomposed to recover the camera motion, and therefore, the projection matrices of both views. Nevertheless, the estimated projection matrices can only be recovered up to one parameter of ambiguity, that corresponds to a unknown scale for the camera translation matrix. Therefore, there exist no possibility of recovering the correct scale of a structure from motion generated model by the image data alone.

Backtracking the previous steps, the fundamental matrix can be used to estimate the essential matrix $\mathbf{E}$ when the camera calibration matrices K and K' are known, given that

$$\mathbf{E} \sim K'^T \mathbf{F} K \tag{2.30}$$

It can be proven that this matrix can be decomposed into a skew-symmetrical matrix corresponding to the translation and an orthonormal matrix corresponding to the rotation between the views by computing the singular value decomposition [20] as follows

$$\mathbf{E} \sim [\mathbf{T}]_\times \mathbf{R} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \tag{2.31}$$

The translation can be directly obtained with an unknown scale and arbitrary sign from

$$[\mathbf{T}]_\times = \mathbf{U} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{U}^T \tag{2.32}$$

In this case, it is common that $\mathbf{T}$ is scaled in such a way that $|\mathbf{T}| = 1$. The axis and angle of rotation can be directly obtained with an arbitrary sign from

$$\mathbf{R} = \mathbf{U} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{V}^T \tag{2.33}$$

The two projection matrices can now be completed, by aligning the reference coordinate system with the first camera

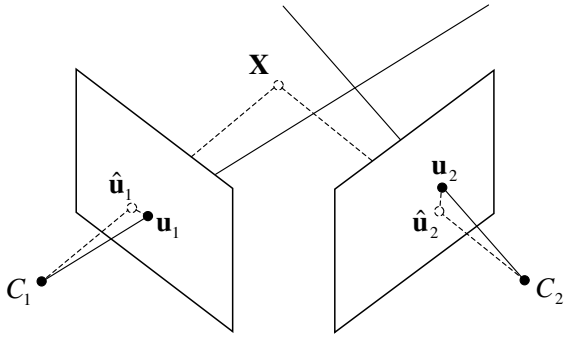


Figure 2.13: Representation of noise in the triangulation process. Given the projection matrices of the two cameras, the position of $\mathbf{X}$ can be computed, ideally in the intersection of the back-projected rays of the image points $\mathbf{u}_1$ and $\mathbf{u}_2$, but due to measurement errors, they will not intersect [44, p. 14].

$$\mathbf{P} = K[\mathbf{I} \mid 0] \tag{2.34}$$

$$\mathbf{P}' = K'[\mathbf{R} \mid \mathbf{T}] \tag{2.35}$$

There are still four possible solutions, given the arbitrary signs for translation and rotation. The final solution can be disambiguated by ensuring that the reconstructed points lie in front of both cameras, because the other three solutions will put the points behind at least of one of the cameras.

2.4.3 Triangulation and noise error

Now that the projection matrices are estimated, the 3D points can be computed from their measured image position in two or more views, using the intersection of the back-projected rays in the views. However the image capture process is not free of measurement noise, so these rays will not generally intersect (Figure 2.13). Therefore the sum of squared errors between the measured and predicted image positions of the 3D point in all views should be minimized as

$$\mathbf{X} = \underset{\mathbf{X}}{arg\,min} \sum_i ||\mathbf{u}_i - \hat{\mathbf{u}}_i(\mathbf{P}_i, \mathbf{X})||^2 \tag{2.36}$$

Where $\mathbf{u}_i$ and $\hat{\mathbf{u}}_i$ are the measured and predicted image positions respectively in view i (Figure 2.13). Assuming that the measurement noise is Gaussian distributed, this approach gives the maximum likelihood solution for $\mathbf{X}$ [44]. From equation 2.6 is known the relation between the theoretical image point on the image plane $\tilde{\mathbf{u}}$ and the 3D point $\mathbf{X}$ is given by the projection matrix

$$\tilde{\mathbf{u}}_i \sim \mathbf{P}_i\mathbf{X} \tag{2.37}$$

Since the homogeneous vectors $\mathbf{u}_i$ and $\mathbf{P}_i\mathbf{X}$ are parallel, it is possible to write

$$[\mathbf{u}_i]_\times \mathbf{P}_i \mathbf{X} = 0 \tag{2.38}$$

This expression has three rows but only provides two constraints on $\mathbf{X}$ since each row can be expressed as a linear combination of the other two (rank of 2). All constraints can be arranged in a matrix $\mathbf{A}$ as

$$\mathbf{A}\mathbf{X} = 0 \tag{2.39}$$

Where $\mathbf{A}$ is a $3n \times 4$ matrix, and n is the number of views on which the reconstructed point is visible. The required solution for the homogeneous point $\mathbf{X}$ minimizes the expression $||\mathbf{A}\mathbf{X}||$ subject to $||\mathbf{X}|| = 1$ and this is given by the eigenvector of $\mathbf{A}^T\mathbf{A}$, corresponding to the smallest eigenvalue. This can be found using the singular value decomposition of the matrix $\mathbf{A}^T\mathbf{A}$.

2.5 Multi-view reconstruction

Once that the fundamental matrices have been found between multiple pairs of images and the location of the 3D points in the relative frames of references from all cameras have been computed, the position of all the cameras in a common frame of reference can be calculated.

There are two main approaches to this problem. One is solving the positions in a sequential manner, where cameras are added one by one to the common frame of reference using the geometry encoded in the fundamental matrices that describe the relative movement between the camera to be added and any camera already on the common frame. The other tries to solve all camera positions simultaneously. The final step for both approaches is applying a bundle adjustment, which is used iteratively to refine the structure and motion parameters by the minimisation of a cost function [44].

2.5.1 Sequential structure from motion

As mentioned before, these algorithms work by incorporating successive views, one at a time (Figure 2.14). As each view is registered, a partial reconstruction is extended by computing the position of all 3D points that are visible in two or more views using the triangulation error minimization technique discussed in the previous section. The initialization pair on which all the successive views will be added later on is typically obtained by decomposing the fundamental matrix relating the first two views of the sequence. There are several strategies to accomplish the registration of the successive views.

Epipolar constraints. The two-view epipolar geometry that relates each view between each other is exploited. This can be employed when the camera intrinsic parameters are known and essential matrices can be used. Since the magnitude of the translation for each essential matrix is unknown, it can be fixed by selecting in the new view a single known 3D point that has already been reconstructed in previous views to maintain a constant scale.

Resection. This alternative relies on using the information of already computed 3D points to estimate the projection matrices of views where the intrinsic parameters cannot be

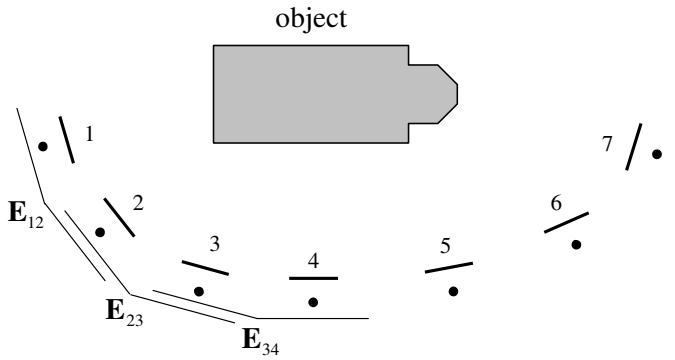


Figure 2.14: In sequential structure from motion, the views 1 to 7 are registered one at a time by estimating the essential matrices $\mathbf{E}_{12}$, $\mathbf{E}_{23}$, etc. relating the change of position and orientation of every view to its immediate predecessor. Then the matrices are decomposed and accumulated as they are added, to compute the 3D model of the object in a common frame of reference. [44, p. 15].

obtained. This is done using an Iterated Extended Kalman Filter, with a separate filter operating on each point to correctly align the projections of multiple camera views [4].

Merging partial reconstructions. It is also possible to merge partial reconstructions using corresponding 3D points in multiple views. When two or three view reconstructions are obtained from image pairs or triplets, they can be merged using their correspondences instead.

The main disadvantage of sequential registration schemes is that they require a high number of correspondences between matching views to accurately estimate the position of the views, and also these points should be visible in multiple views as well, thus a sufficient level of overlap should be provided between the views.

There are documented cases on where these standard algorithms will fail. These include camera rotation in the absence of translation, planar scenes and 3D points lying on a line passing through the optical centres of the cameras on where they are visible [53].

2.5.2 Global structure from motion

Also called factorization or batch methods, here the camera pose and scene geometry is computed using all image measurements simultaneously. The advantage of these methods is that the reconstruction errors can be distributed meaningfully across all measurements, therefore gross errors associated with loop closure can be avoided.

There has been methods developed using singular value decomposition factorization of image point measurements for simplified linear camera models [52]. The disadvantage of these solutions is that modern camera lenses are too wide-angle to be approximated as linear, and therefore cannot be used in real-world scenes.

Other methods rely on “factorization-like” techniques for perspective cameras as well. But these methods are iterative and there is no guarantee that they will converge to the optimal

solution [50].

Another limitation of global structure from motion algorithms is that there exists degenerate structure and motion configurations for which they will fail. They also have a low tolerance for missing data, since they expect that all points are visible in all views. Therefore, they are not applicable in sparse modelling problems.

2.5.3 Bundle adjustment

Bundle adjustment is a process that minimizes a cost function that is related to a weighted sum of squared re-projection errors of the 3D points given the projection matrices of the views [5]. The name refers to the bundles of light rays leaving each 3D feature and converging on each camera centre, which are adjusted optimally with respect to both feature and camera positions. The goal of this process is to determine an optimal estimate of a set of parameters $\boldsymbol{\theta}$ given a set of noisy measurements. These parameters must be able to be observed directly on the data, like the measured image coordinates of the imaged 3D points. Usually a Gauss-Newton iteration is used for rapid convergence [54].

Given a set of predictions on the parameters $\mathbf{z}(\boldsymbol{\theta})$ and a set of their corresponding observations $\bar{\mathbf{z}}$, the residual prediction error $\Delta\mathbf{z}$ is given by the expression

$$\Delta\mathbf{z} = \bar{\mathbf{z}} - \mathbf{z}(\boldsymbol{\theta}) \tag{2.40}$$

Bundle adjustment proceeds by minimizing the appropriate cost function, that should reflect the likelihood of the residual $\Delta\mathbf{z}$. Under the assumption of Gaussian distributed noise in the measurements, the appropriate cost function is defined as a sum of the squared errors for each independent measurement $\bar{\mathbf{z}}_i$ and its respective prediction $\mathbf{z}_i(\boldsymbol{\theta})$ as

$$f(\boldsymbol{\theta}) = \frac{1}{2}\sum_i \Delta\mathbf{z}_i(\boldsymbol{\theta})^T \mathbf{W}_i \Delta\mathbf{z}_i(\boldsymbol{\theta}) \tag{2.41}$$

Where $\Delta\mathbf{z}_i(\boldsymbol{\theta})$ is the feature prediction error and $\mathbf{W}_i$ is a symmetric positive definite weight matrix that is chosen to approximate the inverse covariance of the measurement noise associated with measurement $\bar{\mathbf{z}}_i$. This cost function is typically solved with the Levenberg–Marquardt algorithm [29] for solving non-linear least squares problems.

2.6 Expansion

After the camera poses and the projections of the 3D points are computed into a common frame of reference, the following step is to explore the area in the image plane of each view around the key feature points and spread the matches around these areas to produce a denser point cloud as a final result. If the image has colour information, then this expansion step also colourises the added regions in the final model, producing a colourised point cloud (Figure 2.15).

Given an image pair where their features have correspondences in each other, one possible approach to the expansion problem [27] takes one of the images from the pair and is divided in a grid of partially overlapping square shaped patches. An affine transformation is calculated for these image patches that are located close to correlated features in the second image, under the

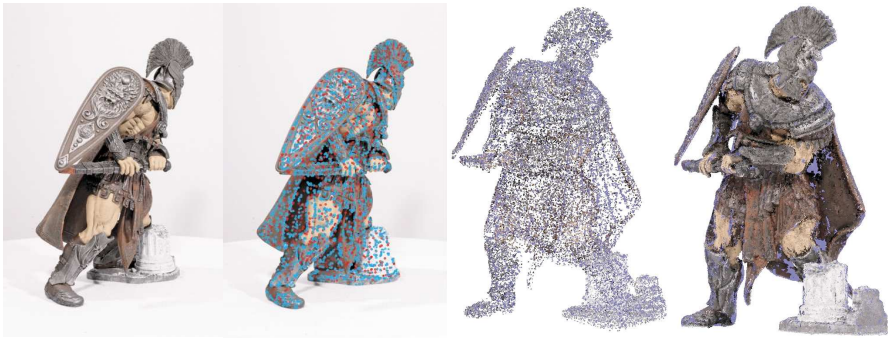


Figure 2.15: Reconstruction steps of the structure from motion algorithm. Left: Arbitrary image of the object to model. Middle-left: Features detected on that image. Middle-right: 3D point cloud of all detected features on all the views after triangulation and multi-view reconstruction. Right: Colourised and denser point cloud after expansion. [16, Adapted p. 2].

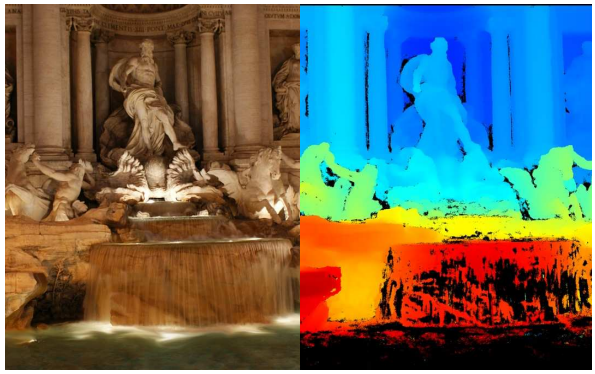


Figure 2.16: Usage of depth estimation in multi-view stereo with a sample image and its corresponding depth map, where a map of colours indicate the depth of every pixel in the image [47, Adapted p. 14].

assumption that nearby pixels also undergo the same or a very similar transformation as the feature itself. A refinement process is used to align the predicted patch correctly in the second image and if the correlation between the patch and its alignment in the second is high enough, it is accepted as a secondary match. This process usually covers most of the object surface densely with matches.

Another approach [47] uses a depth estimation algorithm to compute the image depth on a per pixel basis. This information can then be used to place all the pixels of the images in the final 3D model, yielding a high density point cloud (Figure 2.16).

Chapter 3

Algorithms to Benchmark

Although all available structure from motion packages follow the outline described in the previous chapter, the technical implementation can different since each package is tailored specifically against certain goals such as speed optimization, memory usage, ability to run in a graphics processing unit (GPU) instead of the central processing unit (CPU), mathematical simplification of the models despite accuracy, etc. Some packages use as well different algorithms for the stages of the structure from motion pipeline, product of improvements in recent research in the subjects of feature detection and matching and computer stereo vision.

Some packages only focus on the multi-view reconstruction problem, what is typically referenced in literature as the "structure from motion" stage, where only the camera pose is solved and the keypoints triangulated. These rely on other packages capable of solving multi-view stereo where the expansion stage takes place. Therefore even combination of pipelines are possible, where one step can be done through one software and later steps with different algorithms.

All this differences have impacts on the speed of the overall process as well as the accuracy of the result. For this reason, popular and well documented packages will be chosen to benchmark, specially in the cases where the source code is available to rationalize and interpret the results accordingly. Therefore it is important to point out that this is not an exhaustive list of all available solutions currently present. In this chapter, the software packages to be used are exposed in a detailed manner, highlighting what can they do and how they approach the same problem in their particular way or what they improve upon what was considered "state-of-the art" at the moment of their release.

3.1 Visual Structure from Motion[1]

Visual Structure from Motion is a software package developed by Wu Changchang from the University of Washington and was released in 2013 [59]. It is an integration of previous software utilities, most from the same developer, packed into a graphical user interface (GUI) application for structure from motion reconstruction.

It uses an implementation of the SIFT feature detection algorithm [33] that uses the GPU to process pixels and features parallely in the Gaussian pyramid construction, keypoint detec-

[1] `http://ccwu.me/vsfm/`

tion and descriptor generator. This implementation though was developed specifically for the CUDA application programming interface (API) developed by nvidia for parallel computing on a GPU. If no CUDA-compatible GPU is present, the package allows the use of third-party implementations of feature detection, provided that their output is a set of SIFT features and descriptors.

The feature matching step is done using an approximate nearest neighbour algorithm, and it can be accelerated by using a preemptive feature match. Using the descriptors of the SIFT image feature detection algorithm, this preemptive feature matcher works first by sorting the features in all images by decreasing scale order, then it matches the top h largest scale features in all possible image pairs and then only does a full feature match of a given pair if the image pair contains more than certain threshold t_h of large scale matching features. If it does not, it is discarded. The author claims that the yield of feature matching using only the h largest scale features in every pair is closely correlated to the yield obtained by matching all features on the pair, even for low values of h, such as only using $h = 100$ features [59, p. 2].

The incremental multi-view reconstruction is accelerated using a preconditioned conjugate gradient for the bundle adjustment step [60]. The bundle adjustment is implemented by the Levenberg-Marquardt method for minimizing the cost function for the reprojection error matrices [29]. In simpler terms, Visual Structure from Motion avoids explicitly and iteratively computing the inverse of certain matrices and instead estimates them implicitly, thus reducing the cubic $O(n^3)$ class of complexity of the bundle adjustment problem to linear $O(n)$ time [59, p. 3].

The bundle adjustment is only run for a small set consisting only of the latest cameras added to the model and their corresponding 3D points, and a full bundle adjustment on the whole model is only run every time it increases in a certain relative ratio r. The author establishes that the partial bundle adjustment will be run with no more than 20 cameras and the relative growth ratio for running the full bundle adjustment is set to $r = 5\%$.

Another simplification in the multi-view reconstruction is made when incrementally adding new cameras to the model: if the number of cameras in the model that is compared to the camera to add is capped to a maximum, and thus kept arbitrarily constant for every camera discarding any additional matches that this new camera may have with other cameras already in the model, the time complexity of the whole multi-view reconstruction can be kept in linear $O(n)$ time [59, p. 4].

As a closing step in the multi-view reconstruction, a re-triangulation is performed for fixing the accumulated drift generated by the incremental nature of the algorithm. This process is similar to loop closure, as is only able to fix the drift when a loop in the recorded data is present. This re-triangulation step is performed every time the model grows by $r = 25\%$, and it consist of looking into under-reconstructed image pairs, where the relative pose between the two cameras in the pair is a low ratio between their common points and their feature matches [59, p. 5].

For the final dense expansion stage, the Patch-based Multi-view Stereo (PMVS) software toolchain developed by Yasutaka Furukawa from the University of Illinois at Urbana-Champaign in 2007 [16] is used by default. This algorithm outputs a set of quasi dense rectangular patches covering the surfaces visible in the input images. It decomposes the input images into a set of image clusters of manageable size. Then each cluster is processed independently and in parallel.

3.2 Open Multiple View Geometry[2]

The open multiple view geometry (openMVG) software package was developed by Pierre Moulon, Pascal Monasse and Renaud Marlet of the Center for Visual Computing of the Université Paris-Est and released in 2012 [39]. This package provides the required tools for feature detection and matching and incremental structure from motion, but it does not provide a method for the expansion stage, thus a third party tool must be used if a denser model is desired.

The key feature point of this software package is the use of the "a contrario" methodology for improving the estimations on the fundamental and essential matrices, and thus improving the camera pose estimation. This methodology consist of using adaptative thresholds for the model estimation, instead of using globally-fixed values that are only optimized for a particular scenario. This adaptation frees the user from guessing threshold values and significantly improves the accuracy of the model estimation.

It uses its own SIFT implementation for feature detection and for feature matching it implements the random sample consensus (RANSAC) algorithm [13], which will subject to the "a contratio" methodology. The RANSAC algorithm repeatedly selects a random sample set of the data, whose minimal size is sufficient to estimate the parameters of the model. At each trial, inlier matches are defined with the data that fits the model within an acceptable threshold. After a given number of iterations, the model parameters that maximize the number of corresponding inlier matches are returned. Naturally, the dependency of the threshold parameter has an impact on the accuracy of the estimated model: if its too small, too little data is selected as inlier matches; or if its too big, false positives can become inlier matches. In either cases, an incorrect estimation of this threshold value can lead to inaccurate or wrong models.

The threshold-setting problem appears again when constructing the incremental model from the different views in the dataset, since the pose of points and cameras has to be estimated multiple times from a multitude of image pairs. In this case, the threshold is adapted to the corresponding data noise in the dataset. Nevertheless, using a global threshold can yield again problems, since it may be too low for some noisy data or too high for clear images which will produce false positives. Therefore the "a contrario" methodology finds a model that fits the data with a confidence threshold that adapts automatically to noise.

The "a contrario" methodology relies on the idea that a configuration that is unlikely to be explained by chance is conspicuous, and it has been applied to detection in images in previous work [10]. An approach to this method that improves on the RANSAC algorithm is known as AC-RANSAC [37], on which the statistical criterion is data-specific and avoids empirically setting thresholds for inlier and outlier matches discrimination. In a broad sense, it minimizes the so called "number of false alarms" for any model that can be produced by a probabilistic correspondence between the features of both images, instead of just maximizing the number of inlier matches in the usual RANSAC approach [38]. In this context, a false alarm is understood as a model that is actually due to chance. This approach yields an accurate camera projection matrix that can be used in the later stages of triangulation and incremental structure from motion.

[2] `http://imagine.enpc.fr/~moulonp/openMVG/`

3.3 Open Multiple View Stereovision[3]

The open multiple view stereovision (openMVS) software package was developed by Dan Cernea et al. in 2015. This software is not targeted for incremental multiple view reconstruction, rather it provides a set of algorithms to recover the full surface of the scene to be reconstructed after the reconstruction by structure from motion is finished. The input is a set of camera poses plus the sparse point-cloud of features and it can output either a denser cloud of points or it can continue to build a textured mesh if the user desires to do so.

The implementation of the expansion approach is based on the Patch-Match algorithm [1]. This algorithm takes advantage of the fact that the image elements (patches of image pixels) have been correlated already by an multi-view reconstruction algorithm before. This allows to focus the computational effort into regions that are more likely to have good matches. It is based on the idea that good patch matches can be found via random sampling of a nearest neighbour field and that natural coherence in the imagery allows to propagate these matches quicky to surrounding areas.

3.4 Multi-view Environment[4]

Multi-view environment (MVE) is a software package developed by Simon Fuhrmann, Fabian Langguth and Michael Goesele of the graphics, capture and massively parallel computing department of the Technische Universität Darmstadt and was released in 2014 [15]. It is an implementation of a complete pipeline for image-based geometry reconstruction. It includes solutions for incremental structure from motion, multi-view Stereo and it can also do surface reconstruction.

MVE uses its own implementation of the SIFT feature detection algorithm, and uses jointly as well an implementation of the speeded-up robust features (SURF) [3] for machine-recognizable feature detection. It proceeds then to a feature matching across views and a filtering of false correspondences by the means of enforcing epipolar geometry constraints. This pairwise matches are combined and expanded over several views, yielding feature tracks along all the matching views, where feature track corresponds only to one 3D point in the final reconstruction. For speeding up the matching problem, it uses a combination of reducing the number of features to match in each image as well as parallelizing this process. The reduction of features is done by applying a low resolution feature matching to identify image pairs that potentially will not match before performing a full resolution match. The developers claim that, although good image pairs can be rejected by this method, it does not affect the overall accuracy of the model but decreases significantly the processing time [15, p. 3].

When views have been matched, the incremental multi-view reconstruction first chooses an initialization pair where all the feature tracks in both images are triangulated. Then the rest of views is added one at a time until all reconstructable views are part of the scene. In this stage the lens distortion parameters are also estimated.

[3] `http://cdcseacave.github.io/openMVS/`

[4] `https://www.gcc.tu-darmstadt.de/home/proj/mve/`

After all camera parameters are estimated, the dense reconstruction stage takes place. Aiming to creating denser point clouds than other solutions, MVE uses the "Multi-view Stereo for Community Photos Collections" approach [18]. In this approach, the depth maps for every view are computed, where every pixel is associated with a depth value, representing its distance to the camera. Although this depth map approach creates lots of redundancy since may views overlap and see similar parts of the scene, it scales effortlessly to large datasets as only a small set of the views is actually required for the reconstruction, instead of the usual expansion approach where patches are matched across all views to assert their geometry.

For fusing the depth maps together and produce the denser cloud, the floating scale surface reconstruction approach is used, as it is able to adapt the interpolation and approximation behaviour depending on the sample scale and redundancy without relying on explicit parameter settings [14]. An advantage of this approach is that it doesn't interpolate regions with insufficient geometric data like other depth map fusion approaches, but instead leaves the region empty, useful for outdoors scenes where normally manual clean-up would be required.

3.5 COLMAP[5]

COLMAP is a software package for structure from motion and multi-view stereo reconstruction developed by Johannes Schönberger from the Technische Hochschule Zürich et al., and it was released in 2016. Its main goal is to provide a general-purpose solution for structure from motion reconstruction that can work with high reliability under a variety of conditions, and claims to be an improvement upon the current state of the art techniques used by software packages like visual structure from motion [46].

COLMAP implements a SIFT feature detection and matching algorithm capable to run in the GPU through the use of the CUDA API, but it is also possible to compile and run this stage on the CPU although for a trade-off on execution runtime.

The improvements COLMAP implements over the incremental structure from motion reconstruction includes a geometric verification strategy that makes a more robust triangulation of the scene, it uses a next best view selection for increasing the accuracy of the reconstruction process, a more robust triangulation method and an iterative bundle adjustment, re-triangulation and outlier filtering strategy to improve completeness and reduce drift.

The geometric verification strategy consist of identifying the relationship between two images and classifying it as a panoramic (pure rotation), planar or general transformation model. This is helpful for finding the initialization pair for the incremental structure from motion stage, as panoramic pairs can lead to an unstable scene reconstruction. Also triangulation from panoramic pairs is avoided on the later incremental reconstruction step, as it can lead to degenerate points in the scene as well.

The next best view selection refers to the problem of how to select the next camera to add to the current model in the incremental structure from motion reconstruction, given the already triangulated camera poses and 3D points. Usually this problem is solved by selecting the view that sees the most triangulated points, but this process can lead to a later failure in triangulation if the pose estimate for this view is inaccurate. The proposed solution is to

[5] `https://demuc.de/colmap/`

select the view that sees the most triangulated points as the default approach, except when the configuration of observations of this view is not well-conditioned [19]. Experimental observations have determined that the accuracy of the pose estimation of a camera depends on the number of features and their distribution on the image, therefore COLMAP develops an algorithm to compute a score on a view based on these criterions, and then decides if the view is suitable to be registered into the model when the view presents a high number of visible features already triangulated in an homogeneous distribution.

A robust triangulation method can be achieved by using image pairs with larger baselines for the triangulation. The usual approach looks for similar images to triangulate, and therefore this pair has a small baseline and can lead to an inaccurate computation of the camera pose. Larger baselines can be constructed by producing feature tracks along concatenations of two-view correspondences, and a multi-view triangulation method is performed.

Bundle adjustment applies a similar approach to the one used in visual structure from motion, where only a partial bundle adjustment is performed on the latest added views and a complete bundle adjustment is executed only when the model grows after certain percentage. Some views are also filtered out after the global bundle adjustment, such as views with large re-projection errors and those product of degenerate cameras, such as panoramas. Also re-triangulation is performed, minimizing drift errors and accounting for loop closure. These stage of bundle adjustment is also iterated over and again until the number of filtered views diminishes, since it has been observed that this can significantly increase the results due to the fact that bundle adjustment is severely affected by outliers. Therefore discarding them and repeating the process can increase accuracy.

The multi-view stereo dense reconstruction implemented by COLMAP runs only and entirely on the GPU through the CUDA API. It uses a depth and normal estimation algorithm based on pixelwise view selection using photometric and geometric priors, multi-view geometric consistency for refinement and image-based depth and normal fusion [47]. It is based on the PatchMatch algorithm [1] but applies several improvements to it. Among others, it includes the support of an adaptive window through bilateral photometric consistency for improved occlusion boundary behaviour and it introduces a multi-view geometric consistency term for simultaneous depth and normal estimation and image-based fusion.

The PatchMatch algorithm employs normalized cross-correlation to compute colour similarity between two patches. This process is statistically optimal for Gaussian noise but is specially vulnerable to producing blurred depth discontinuities. Therefore, a bilaterally weighted adaption of the normalized cross-correlation process is used by COLMAP, where every pixel is weighted with the likelihood of belonging to the same plane as the centre pixel in the other patch. This adaptation is the so-called photometric consistency and it has been empirically proven to produce more accurate results at occlusion boundaries.

Since multi-view stereo usually suffers from outliers due to noise, ambiguities and occlusion, the photometric consistency may be ambiguous as large depth variations induce only small cost changes in the weight function. Photometric ambiguities are usually unique to each individual view, so using the information from multiple views can help to find the right depth solution. If multi-view depth coherence is enforced through left-right consistency checks, these outliers can be filtered out in a post-processing step where these geometric consistencies are enforced.

3.6 PhotoScan[6]

PhotoScan is a closed-source, commercial software package developed by Agisoft and initially released in 2010. It is able to follow the reconstruction pipeline completely, including structure from motion and multi-view stereo reconstruction. Unfortunately there is little information about how this software works or what kind of algorithms it uses for feature detection, matching and reconstruction. Nevertheless, this package has been extensively studied in various research and used in diverse industry cases [43][51][55][49][26] where it has received a very positive reception. Therefore is worth analysing its results and compare them to other solutions as well.

What little is known of PhotoScan is that it uses a similar approach to the SIFT feature detection and matching techniques in the sense that their algorithm also detects features and assigns a descriptor to them that is used for matching. It uses triangulation and bundle adjustment algorithms for solving the camera poses and it implements a depth-map computation approach for point cloud expansion [48].

[6] http://www.agisoft.com/

Chapter 4

Methodology

The process on how the different algorithms are benchmarked is discussed in this chapter. First all the reconstruction pipelines using the different software packages mentioned in the previous chapter are be enumerated, as well as which software version is used. The datasets on which the algorithms are tested is also presented, and is also explained why are they chosen. The equipment used to record the data and to process it is also described, and finally it is discussed how the comparison process will take place and what measurable quantities and subjective estimations will be taken into account for analysing the results.

4.1 Structure from Motion pipelines

Exploiting the fact that it is possible to combine modules from different packages for the structure from motion step, that is feature detection, matching, triangulation and multi-view reconstruction; and the multi-view stereo step, different combinations between these packages is benchmarked.

Not all possible combinations can be tested though, since for some cases is not possible to export one project from one package to a understandable format by another. Therefore, the following combinations are to be tested (Figure 4.1).

A – Visual SFM Data processed with Visual Structure from Motion for the complete reconstruction pipeline, using the default PMVS multi-view stereo expansion included for the dense reconstruction.

B – openMVG + openMVS A sparse point cloud will be obtained with openMVG and it will be densified using openMVS. This is the approach recommended by the openMVG documentation.

C – Visual SFM + openMVS The sparse model obtained using Visual Structure from Motion can also be densified with openMVS, which natively understands Visual SFM projects.

D – MVE As this software package can execute the complete structure from motion and multi-view stereo pipeline, here it will be tested.

E – Visual SFM + MVE The sparse point cloud generated by Visual SFM will be densified by MVE. An importer for Visual SFM projects to the MVE format is provided.

F – openMVG + MVE Here as well the sparse point cloud generated by openMVG will be densified by MVE. The respective importer is provided too.

G – COLMAP with Photometric Consistency The COLMAP pipeline will be executed for the dataset, applying a photometric consistency check in the multi-view stereo expansion stage. This is the default pipeline suggested by the COLMAP documentation.

H – COLMAP with Geometric Consistency After finishing the COLMAP dense model with the photometric consistency check, a geometric consistency check can also be applied. This is the recommended pipeline by the COLMAP documentation as it can improve accuracy, although it also mentions that it takes a substantial amount of time to compute.

I – PhotoScan The default PhotoScan pipeline.

All pipelines are run with the default values they have defined in the provided configuration files or documentation, and the automatic camera calibration routines they all provide is to be used as well to compute the camera intrinsic parameters. In some packages, such as openMVG and MVE that rely on a sensor database where the sensor dimensions are stored for computing the actual focal length of the camera, the camera information has been added if it was not already provided. The parameters are then estimated from the EXIF data stored in each image, or from the image size itself.

Some packages are still under active development, even some receive updates at a weekly basis such as COLMAP. Therefore the most stable version at the moment of writing will be used for the benchmark. Table 4.1 presents which version of each package is used and when it was released.

Table 4.1: Software versions used for the benchmark.

Software Package	Version	Date released
Visual SFM	v0.5.26	26-Apr-2015
PMVS	v2	13-Jul-2010
openMVG	v1.1 r721	04-Jan-2017
openMVS	v0.7-18	26-May-2017
MVE	–	05-Jun-2017
COLMAP	v3.1-34	30-Jun-2017
PhotoScan	v1.3.3-4827	16-Aug-2017

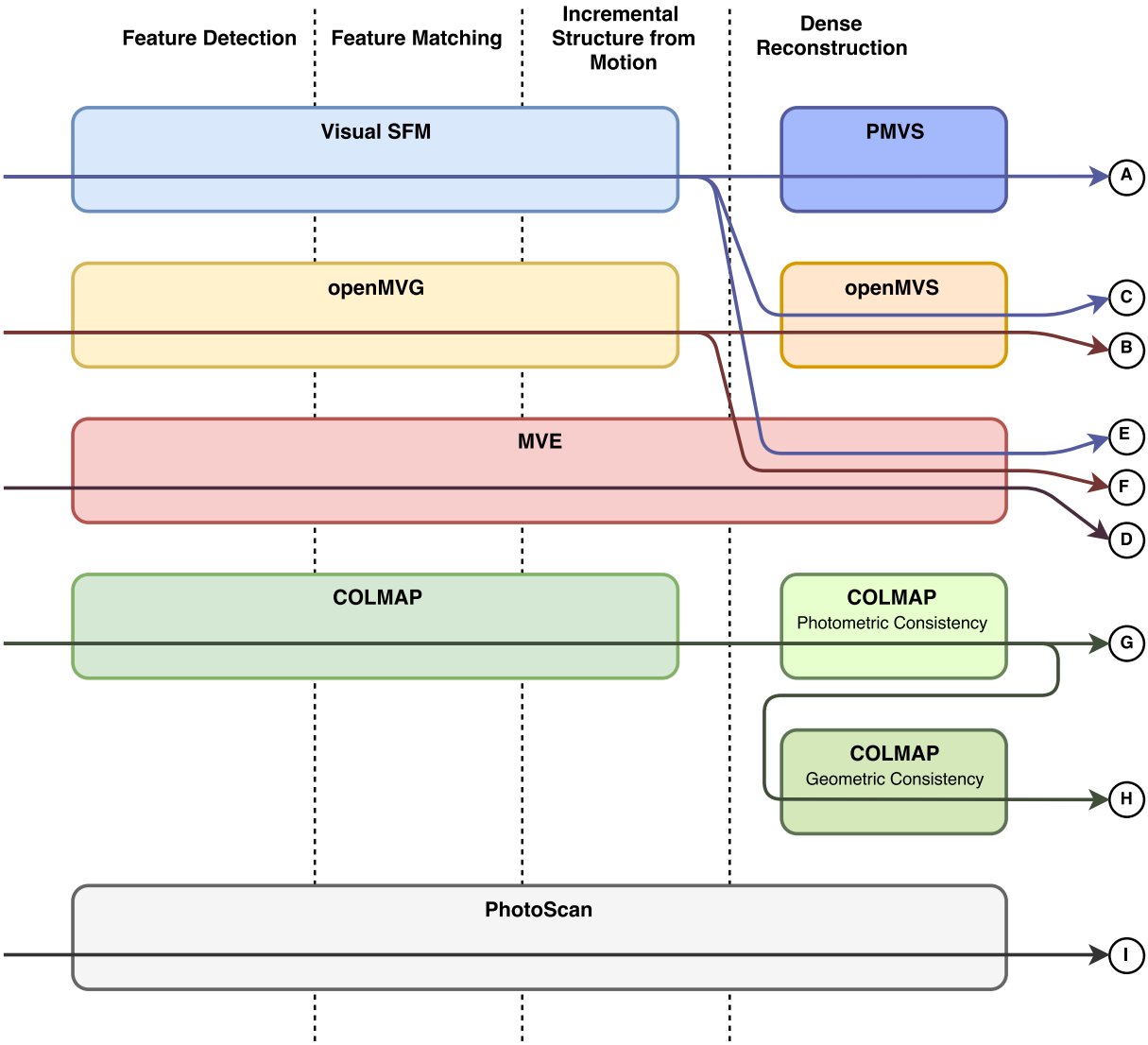


Figure 4.1: Structure from Motion Pipelines used to process the input data that will be benchmarked.

4.2 Datasets

Structure from motion algorithms have known limitations when reconstructing structures with texture-less surfaces, such as buildings with plain colour painted walls [45], as well with reflective surfaces and foliage. The selection of datasets has considered these issues, as to evaluate how these algorithms perform under these situations and to check which one of them produces the most complete and accurate result overcoming these limitations.

Figure 4.2: Aerial view of the robotics hall of the University of Würzburg.

4.2.1 Robotics hall

The first dataset to be recorded is the robotics hall, belonging to the Chair VII of the Institut für Informatik of the Julius-Maximilians Universität Würzburg (Figure 4.2). It is located on the campus Hubland Süd of the university, in the outskirts of Würzburg (49°46′53″N 09°58′29″E) and the dimensions at its base are of 17.5 m × 29.5 m. This building is primarily used for the development and improvement of autonomous mobile robots by the university, and therefore there is an open space behind the building where the tests take place.

The main reason to choose this building is due to the openness of the space around it. It is not close to any other major structure and there are very few trees around it, which makes it an ideal open location lo fly the drone and collect data without interferences. Therefore complete data from this building from both the laser scanner and the camera on the drone can be recorded, including the roof of the building for the laser scanner data, taken from the roof of a taller nearby building.

The geometry of the building is very simple consisting only of two rectangular volumes, it also features a small number of windows that can cause reflections but issues may arise because it has almost texture-less walls.

4.2.2 Chapel at Randersacker

The next dataset to be recorded is the Maria-Schmerz chapel on the vineyard hills around the village of Randersacker near Würzburg (49°45′50″N 09°58′53″E, Figure 4.3) and the dimensions at its base are of 7.5 m × 15.5 m. The chapel is in the middle of a cluster of trees and on a hill with a moderate degree of steepness, therefore the setting of the location makes it particularly difficult to manoeuvre the drone around it, specially on the front side of the chapel that is very

Figure 4.3: Aerial view of the chapel on the hills near Randersacker.

close to the trees, so image data can be poor in this region due to the oblique point of view, result of flying higher above the trees in this area. Laser data can be acquired from all around the structure, but no data for the totality of the roof can be recorded, specially the side facing downhills, as the laser scanner has to be placed at a lower height than the structure.

The fact that the chapel itself is made of stone makes it perfect for a structure from motion reconstruction, since the scene is rich in features to extract. Therefore no gaps in the models produced by the algorithms are expected.

4.2.3 Practice hall at the Bavarian Firefighter School

The biggest dataset to be acquired is the practice hall of the Bavarian Firefighter School, located in the Zellerau district of Würzburg (49°47′53″N 09°54′12″E, Figure 4.4) and the dimensions at its base are of 76.8 m × 48.5 m. Since this building is used for search and rescue training, it features a large glass surface on the back wall of the structure to let for natural light and to allow visibility of the inside from the outside. Also a substructure in the front and top has also a glass surface. This creates a lot of reflections of the surrounding structures around the building, so the objective is to check how the algorithms react under such geometric disturbances. The building also features two large white walls to the sides, without any prominent feature, so it is expected that the expansion stage of the algorithms will have problems in this area.

Since the building is particularly big, the drone has to be flown at a higher altitude from further away, so a loss of precision on the generated model may be possible. Laser measurements can be taken around and above the building, except for the ceiling of the secondary structure in the middle of the building.

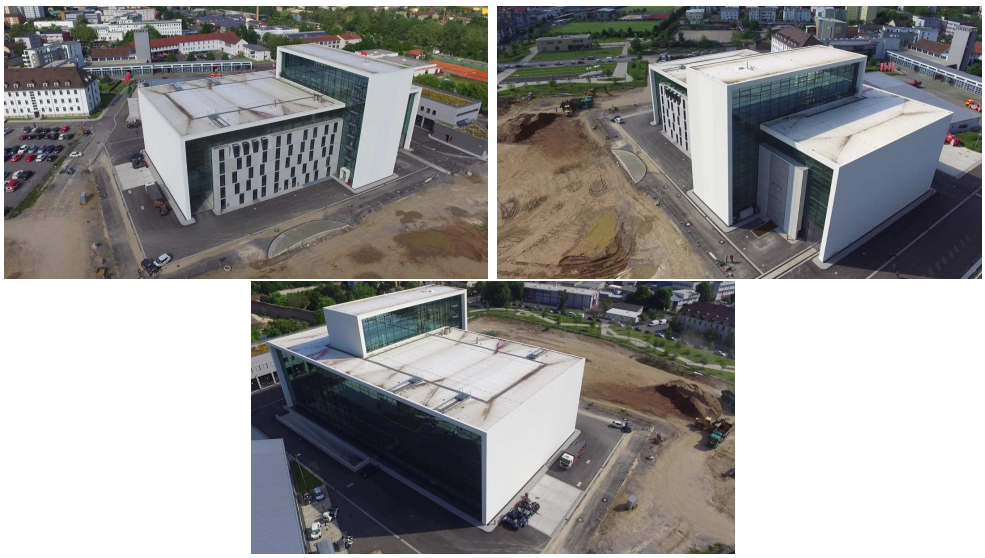


Figure 4.4: Aerial view of the practice hall at the Bavarian Firefighter School in Würzburg.

4.2.4 Burning house at the Bavarian Firefighter School

The final dataset to be recorded is also located in the Bavarian Firefighter School in Würzburg. This structure is the burning house (49°47′54″N 09°54′09″E, Figure 4.5) and the dimensions at its base are of 12.6 m × 12.5 m. In this building, the firefighter students can practice search and rescue techniques under a realistic environment, as the building can generate smoke and high temperatures inside. The reason to choose this building is mainly to evaluate how the structure from motion algorithms deal with a dynamic structure, such as smoke, when creating the 3D model of the object. In the development of this work it was found that certain algorithms had problems with depth perception in the expansion stage with clouds in the horizon, with lead to noisy structures as a result around the main object. Therefore the question on how this algorithms would perform with noise from smoke arose, since it can occlude large sections of the object of interest from multiple perspectives and fail to triangulate. Since part of the scope of this work is to assert the performance and adequacy of the algorithms for emergency situations, this evaluation is of interest.

There is nothing special about the structure itself. The façade of the building resembles a typical modern german house, the walls are of plain colours, except where the smoke has tainted them, and the geometry is simple. During the recording of the image data there were people moving around the structure, so the effect of partial occlusions by moving objects can also be seen.

4.3 Equipment

For this work, first a reference frame to which compare the results is needed, this is acquired with a commercially-available ground-surveying 3D laser scanner. The input data for the algorithms

Figure 4.5: Aerial view of the burning house at the Bavarian Firefighter School in Würzburg.

is captured with a popular commercial drone with a built-in camera. No modifications will be done to this equipment; they will be used as default. Finally, the data has to be processed in a computer with sufficient resources for the task; a home computer is used for the processing of the data and the comparison technique. In this section the characteristics of the equipment are presented with more detail.

4.3.1 Laser scanner

For taking a reference frame of the structures to model, the commercially available ground-surveying Riegl VZ-400 scanner is to be used (Figure 4.6). This is a 3D laser scanner capable of achieving up to 350 m of range with an accuracy of 5 mm and a precision of 3 mm. It uses a rotating multi-facet mirror that allows the scanner to get a complete 360° horizontal field of view and a vertical field of view of 100°, and it can produce scans of an angular resolution of at least 1.8 arcsec. It is also able to record the GPS coordinates of the location where the scan takes place.

The laser scanner data is acquired by scanning the structure from multiple locations around it. Each scan will contain a 360° model of the structure from multiple perspectives, attempting to cover the structure completely, where possible. Considering a substantial amount of overlap between the scans, they will be later combined by using the 3D Toolkit[1] (3DTK). This is a software package for processing 3D point clouds that implements a 6 degrees of freedom (DoF) simultaneous localization and mapping (SLAM) algorithm. It uses the iterative closest point algorithm for scan matching and heuristics for closed loop detection and a global consistency method to produce a highly precise mapping system [41]. In this way the reference for comparison against the results of the structure from motion algorithms will be created.

[1] `http://slam6d.sourceforge.net/`

Figure 4.6: Riegl VZ-400 ground-surveying 3D laser scanner.

4.3.2 Drone

The video footage is to be taken from a DJI Phantom 3 standard edition amateur drone. It features a built-in camera on a 3 DoF gimbal for image stabilization. The camera has a focal length of 20 mm and a sensor form factor of $1/2.3''$ with a diagonal dimension of 7.7 mm, yielding a field of view of 94°. It is set to work at an aperture of $f/2.8$ and focused on infinity.

The camera can take still pictures up to a resolution of 4000×3000, and video footage at the 2.7K video mode (2704×1520) at 30 frames per second. For the image acquisition the 2.7K video mode is to be used and still images are obtained from it at the processing stage, due to the fact that the fastest still-image capture interval the drone allows is 5 seconds in the "Time-lapse mode". Because of the wide field of view of the camera lens , it features a noticeable degree of radial distortion, but no corrections will be applied to the image data taken by the camera.

The drone can fly up to 120 m above its launch point, and is able to do so for around 25 minutes per battery charge. The data will be acquired with the "Point of Interest" flight mode, which automatically controls the drone to fly around a given point in a circle of adjustable radius and height; and also controlling automatically the camera orientation through the gimbal to point towards the centre of the circle as it cycles around the object. During flight the path may be altered in radius or in heigh to avoid obstacles or change the perspective of the object, but the centre of the circle will be kept constant.

4.3.3 Computer for data processing

The data processing is to be done on a personal portable computer running Ubuntu 16.04 LTS x86-64 as operating system. The computer has a total workspace memory (RAM) of 16 GB, an Intel Core i7-4700MQ as CPU, having four cores operating at a frequency of 2.4 GHz and with Hyper-Threading enabled (allowing for eight parallel threads), and a nvidia GeForce GTX 765M as GPU with 2 GB of video memory and 768 CUDA cores operating at a frequency of 850 MHz.

By default, all the dependencies that any of the software packages require are obtained from the Ubuntu repository, except on the cases where they are not published there or incom-

Figure 4.7: DJI Phantom 3 Standard Edition amateur drone.

patibilities arise with that version. For most of the software packages, the installation is well explained in their respective documentations and no issues where encountered for openMVG, MVE, COLMAP and PhotoScan.

Visual SFM requires three dependencies that are not published in the default repositories of Ubuntu in order to be built (SiftGPU[2], Multicore bundle adjustment[3] and Graclus[4]). PMVS[5], used for Visual SFM for the dense reconstruction stage, also requires one dependency not available in the repositories (CMVS[6]) and is also not designed by default to be executed on a x86-64 architecture, therefore some adjustments have to be made to the source code and Makefiles. Fortunately installation guides are available[7] on the Visual SFM website on how to compile each dependency and alter it in order to successfully build it in a x86-64 architecture. In this work, using these guides leaded to a successful installation of Visual SFM, but some technical knowledge is required in order to make the correct changes.

In the case of openMVS, the only problem that arose during the installation was with the default version provided by the Ubuntu 16.04 repository of the Ceres solver by Google, which provides a beta version (v3.3.beta) that fails to compile with openMVS. Therefore, a non-beta version (v3.3) had to be downloaded from the Ceres project website and built specifically for openMVS. This beta version is no longer used in versions of Ubuntu superior to 16.04. No other problems were found on this installation.

4.4 Comparison process

As stated before, the data acquisition consists of surveying the structure in two stages. First, measurements with the laser scanner are made from around the structure and above it, where possible. Second, the drone is flown around the structure in the "Point of Interest" flight mode, so the structure is always the central point of the video taken with the drone.

[2] `http://cs.unc.edu/~ccwu/siftgpu/`
[3] `https://grail.cs.washington.edu/projects/mcba/`
[4] `http://www.cs.utexas.edu/users/dml/Software/graclus.html`
[5] `http://www.di.ens.fr/pmvs/`
[6] `http://www.di.ens.fr/cmvs/`
[7] `http://ccwu.me/vsfm/install.html#linux`

4.4.1 Data preparation

The laser data is processed with 3DTK aligning the taken scans with the iterative closest point (ICP) algorithm using the recorded GPS data as a starting position estimate, and then with a global SLAM 6D algorithm for global consistency and loop closure [41]. Table ... shows how many 3D points each laser scanner model has.

From the recorded video footage of the drone, images will be extracted with ffmpeg at a rate of 1.2 frames per second, so for each 5 seconds there will be 6 frames extracted. Since the drone is flown at a relatively slow velocity, no more than a metre per second, this provides sufficient overlap in a frame to frame basis without generating too much redundant input data for the algorithms.

4.4.2 Data processing, usability of the software and issues

After the images have been extracted, the execution of the determined pipelines follows. Visual SFM offers a graphical user interface (GUI) for processing the complete pipeline with just the press of a few buttons, and the user is able to see the model as it is built. The other package that offers a GUI is MVE, but it was not used in this work as the functionality it offers is not well documented, so the command-line interface is used instead. The rest of the packages only offer a command-line interface, but all of them have a well documented procedure on how to use each component they provide, and how to import or export data across multiple formats for compatibility with other packages, so no special remarks are done in this work regarding the usability of the packages.

As the feature match stage in the typical structure from motion pipeline takes exponential time to be executed in function of the number of input images, a list indicating the pairs to be matched is provided to the algorithms. This simplification is done based on some assumptions, such as comparing one image only with the previous and next n images is appropriate, since they are taken from a continuous video and little overlap will occur on completely different segments of it. Also, segments of video that have the same or similar features of the structure in view are also matched together, and therefore loop closure is ensured. This is also true across multiple video files, as its images are also matched together across the different segments so they can be incremented to the same structure in the reconstruction stage. The same list with the matching pairs is provided for all the algorithms, and other forms of heuristics for finding compatible matches are disabled. Table 4.2 shows how many images are produced for each model, how many pairwise matches have to be computed theoretically and how many are indicated by the list and actually computed.

One issue that was encountered in this work that is worth mentioning though is regarding the dense reconstruction expansion stage of COLMAP, for both photometric and geometric constraints for consistency. Since this step runs entirely on the GPU for depth map generation and is therefore resource intensive, the operating system of the processing machine restarts the GPU driver because it thinks that it entered an infinite loop, as the expansion process does not responds to the operating system until it is finished. This has the consequence that the process is killed, and therefore cannot be executed for more than 10 seconds. This issue is explained is explained in the COLMAP documentation, although is presented as a possibility that can

Table 4.2: Number of images extracted per model, showing how many pairwise matches can be theoretically verified but indicating how many are actually performed.

Model	Extracted Images	Theoretical matches	Computed matches
Robotics hall	1,075	577,275	70,908
Randersacker chapel	995	494,515	74,332
Firefighter's hall	2,719	3,695,121	178,458
Burning house	342	58,311	13,260

happen on certain computers or operating systems, and is solved by stopping the graphics server and running the operating system in command-line interface mode only.

4.4.3 Registration and comparison

Since the structure from motion reconstruction cannot determine the scale of the model due to the ambiguity on the scale of the translation on the triangulation stage, the structure from motion models must be registered first against the laser scanner models with an adjustment for scale, in order for being able to compute the cloud to cloud distance. For this step 3DTK is used, choosing the unit quaternion minimization method for the ICP algorithm with scale adjustment as a registration algorithm [22], which uses the root-mean-square of the ratio between the deviations of the centroids of each point cloud to estimate the scale.

A visualization tool is needed for presenting the generated models, as well to gather data for an objective comparison. This is done by using CloudCompare[8]. This is a software for 3D point cloud visualization and gathering statistical information such as fitting error distribution models to the data. It allows for computing cloud to cloud distances when the models are registered, and can export a histogram of the distance distribution for all points in the point cloud.

4.4.4 Metrics

After all structure from motion models have been registered against its corresponding laser scanner model, metrics can be gathered for objectively comparing the models one to another. The first metric to be evaluated is the number of points that the generated point cloud has. It can be taken as an indicator of the density of the generated model, as well as to how detailed it might be.

The second metric to be used is the execution time of each pipeline. The execution time will be measured from start to finish of each program of the software package and will be grouped in four stages: feature detection, feature matching, sparse reconstruction and dense reconstruction. Feature detection and feature matching do not require any additional introduction, given the definitions established in previous chapters. Sparse point reconstruction includes the process of incremental structure from motion, triangulation and bundle adjustment; since all algorithms group these steps as a whole: it is an iterative process that involves the three of them, one after

[8] `http://www.cloudcompare.org/`

another in every iteration. Dense point cloud reconstruction includes the expansion stage or the depth estimation stage, depending on which strategy the pipeline utilizes to generate the dense point cloud.

The third metric to be evaluated is the cloud to cloud distance that is gathered after the clouds are registered against the laser scanner. This distance is interpreted as the error in the estimation of the structure from motion pipeline compared to the laser scanner data, as every point in the generated point cloud has associated to itself the distance to the closest point in the laser scanner point cloud. This distance is computed by CloudCompare as the nearest neighbour distance.

This error distance can be modelled as a Weibull probability distribution for comparison across the different pipelines [56], which is computed by CloudCompare. This distribution is used to describe probabilities of events that decay exponentially with time. Since the cloud to cloud distance error is not signed, it cannot be treated as a Gaussian distribution as it is not symmetrical. But the amount of points for a given distance error decays exponentially as the distance error increase, therefore is a good candidate for the Weibull distribution. The Weibull distribution is characterized by two parameters: a shape parameter k that defines if the shape of the curve indicates an exponentially rising or decreasing probability over time; and a scale parameter λ that defines how spread the distribution is over time. Therefore, a lower value of λ indicates that the error is distributed closer to zero for a similar value of k. The Weibull probability function is defined as follows.

$$f(x;\lambda,k) = \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^{k}} \tag{4.1}$$

For comparing the distribution of error across different models, the expected value $E(x)$ and the variance σ^2 will be calculated from each Weibull distribution modelling each resulting point cloud. These are calculated as follows.

$$E(x) = \lambda\,\Gamma\left(1+\frac{1}{k}\right) \tag{4.2}$$

$$\sigma^2 = \lambda^2\left[\Gamma\left(1+\frac{2}{k}\right) - \left(\Gamma\left(1+\frac{1}{k}\right)\right)^2\right] \tag{4.3}$$

Where $\Gamma(z)$ is the gamma function. As a final metric, it will also be indicated the minimum error distances that includes the 50%, 75%, 90%, 95% and 99% of all the points in the generated point cloud from all pipelines. Therefore the smaller these minimum error distances are, the closer are most points to zero error.

Chapter 5

Results

In this chapter the generated models by all the proposed pipelines will be analysed. First the outlook of the models will be presented subjectively to the reader: how complete do they look, how noisy they are, if they show artefacts or distortions, what problems and limitations are present in general, etc.

The next step is to review the time performance of the pipelines. It will be analysed by grouping the process of reconstruction in four steps: feature detection, feature matching, sparse point cloud reconstruction and dense point cloud reconstruction. The first two are self descriptive, given the definitions established in previous chapters. Sparse point reconstruction includes the process of incremental structure from motion, triangulation and bundle adjustment; since all algorithms tend to group these steps as a whole as it is an iterative process that involves the three of them. Dense point cloud reconstruction includes the expansion stage or the depth estimation stage, depending on which strategy the pipeline utilizes to generate the dense point cloud.

The final step is to present the comparison of the generated models against the reference frame given by the laser scanner measurements. As presented in the previous chapter, the cloud to cloud distance is computed for the generated models against the reference frame, and in this chapter it will be presented how big those distances are for the majority of the points and the distribution of this error distance, as to give an objective model on how to compare the pipelines with each other.

5.1 Generated models

5.1.1 Robotics hall

Nine models of the robotics hall in the grounds of the University of Würzburg where generated, one corresponding to each pipeline, with 1,075 images as input. Figure 5.1 shows the resulting size of each point cloud. The biggest point cloud was generated by openMVG with MVE for dense reconstruction, generating a total of 194,524,990 points. The smallest cloud was generated by PhotoScan with only 906,129 points.

Figures A.1, A.2 and A.3 show three different perspectives of the complete model for each pipeline that was run. Although the pipelines where MVE was used as a dense reconstructor,

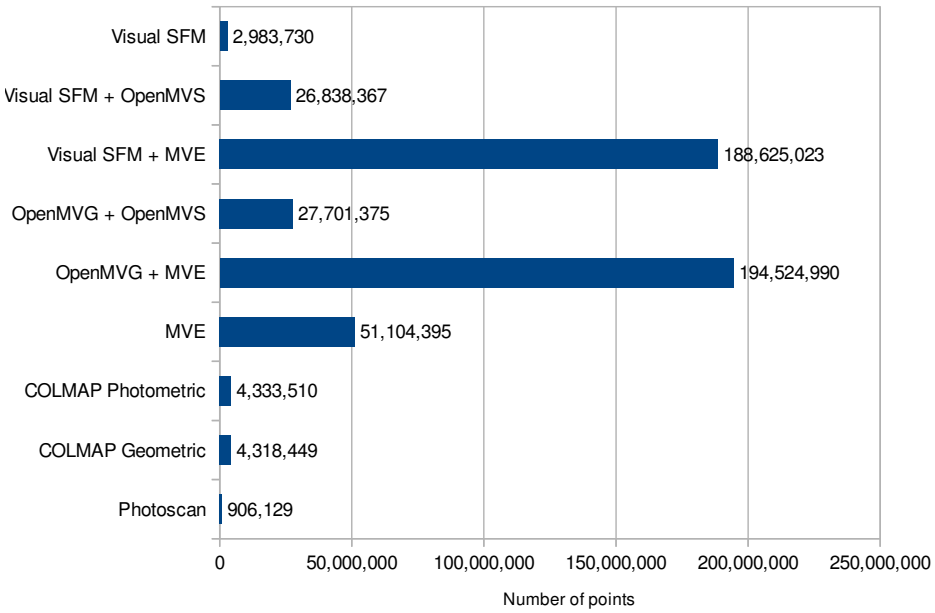


Figure 5.1: Point count per pipeline in the resulting point clouds for the robotics hall model.

that are Visual SFM with MVE (Figure A.1c) and openMVG with MVE (Figure A.1e), seem the most noisy for this model in particular, specially near the roof of the building. The model generated entirely with the MVE pipeline has a prominent artefact to the sides of the roof of the building, which is visible in the view on figure A.1f but not so much in the other two. This may indicate that the algorithm was not able to correctly triangulate points from certain views in the data, which leads to some projection aberrations.

Another interesting artefact in the produced models is colour bleeding from the grass in the background. This is evidently in the model generated by openMVG, specially the view on figure A.2d, where the edges of the building and its roof are green. This model is not the only one to present this artefact, but is the one that shows up more predominantly from its different views. The models generated by the pipelines of Visual SFM, COLMAP with photometric constraints and COLMAP with geometric constrains are the ones where the structure is most distinguishable: they contain some degree of noise but the structure looks sharp. The model generated by Visual SFM contains many holes in its structure, though, specially visible in figures A.1a and A.3a. The model generated by PhotoScan is evidently less dense than the others, but the structure is also distinguishable and complete from the different views presented.

5.1.2 Chapel at Randersacker

For the nine pipelines, the openMVG with MVE generated again the largest cloud with a total of 178,317,336 points, and the smallest cloud was again generated by PhotoScan with 1,186,148 points (Figure 5.2). The rest of the generated clouds range between 4,000,000 and 50,000,000

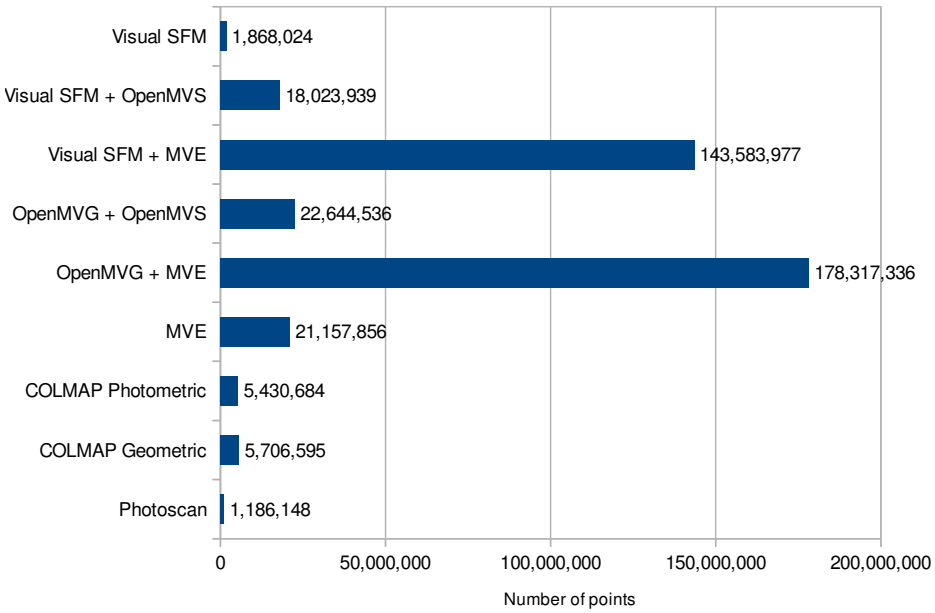


Figure 5.2: Point count per pipeline in the resulting point clouds for the chapel at Randersacker model.

points. As the number of input images is approximately the same as the one used for the robotics hall model (995 images), the generated point clouds are as well in similar sizes. In figures A.4, A.5 and A.6 the generated models are presented from three different perspectives.

This dataset was recorded when there were some clouds in the backgroud, and it proved to be problematic for some algorithms, due to the lack of features, to correctly determine that these clouds do not belong to the foreground of the model. This is most noticeable on the model generated with MVE (Figure A.4f) and the models where the expansion stage was made with openMVS (Figures A.4b and A.4d), which have large amounts of noise around the chapel's bell tower, and in the model produced by MVE the bell tower is barely visible.

As expected, all pipelines where able to reconstruct the model without any noticeable gaps in the structure, due to the feature rich nature of the structure itself. Even on areas where the images where taken with a very steep angle, such as the front of the chapel, the structure looks complete. Colour bleeding is not a concern in this model, as there is not much of it. Only some colour bleeding is visible on the top edge of the roof on the models generated by COLMAP, specially in figures A.5g and A.5h.

5.1.3 Practice hall at Firefighter School

The practice hall was generated with almost three times as many pictures than the other two models (2,719 images). The largest point cloud for the practice hall pipeline was generated with a total of 312,227,459 points, again by openMVG with MVE as a dense reconstruction step. The smallest point cloud was generated as well by PhotoScan with a total of 273,260 points (Figure

5.3). The rest of the clouds range between 5,000,000 and 20,000,000 points. In figures A.7, A.8, A.9 and A.10 four views of the generated model are seen.

The generated point clouds for this model contain overall a lot of noise and artefacts. For example the model generated with the Visual SFM + openMVS pipeline (Figure A.7b) contains large amounts of noise in the roof of the building, even making the secondary structure of the building indistinguishable. The model generated with MVE (Figure A.7f) also features many straight artefacts in multiple directions, which makes the borders of the structure hard to delimit. None of the pipelines was able to correctly generate the lateral walls of the structure, this is most evident in figure A.9. The algorithms where Visual SFM was used for the sparse reconstruction (Figures A.8a, A.8b and A.8c) had problems correctly triangulating a nearby structure, making a large visible artefact on the lower right corner of these figures, instead of an open driveway.

The models generated by COLMAP have many artefacts due to perspective projection errors, which are predominantly visible in figures A.10g and A.10h. In figure A.10 it can be seen that most pipelines where able to reconstruct some part of the large reflecting surface of the building, but none of them was able to do so successfully. The model generated by PhotoScan completely failed to triangulate the large reflective surface of the building, as that part of the building is completely missing from figure A.10i. In this same figure it can be seen that the model generated by PhotoScan tends to round the edges of the building as well, as the rightmost part of the roof seems to fall into a slope instead of forming a 90° angle with the wall. An hypothesis on why this dataset has so much noise and artefacts is because of the large number of reflective surfaces, which makes triangulation very difficult; and also due to the repetitive patterns the building possess when taking pictures from close to mid range.

Bleeding is also predominant in the model generated by MVE in figure A.9f, where a large green blur can be seen just below of the edges of the roof. There is no clear preference of generated point cloud for this model as all of them present, in one way or another, some degree of noise and distortion.

5.1.4 Burning house at Firefighter School

The burning house model was generated with a third of the number of images used for generating the robotics hall and chapel models (342 images), so as expected the models are roundabout one third of the size as for the robotics hall and chapel models. The largest model was generated yet again by openMVG with MVE for a total of 57,853,718 points and the smallest was generated by PhotoScan with just 141,392 points (Figure 5.4). The rest of the models have between 2,000,000 and 12,000,000 points. In figures A.11, A.12 and A.13, the model is shown from three different perspectives for the 9 pipelines.

Besides the regions where there was smoke in the recordings, the models look overall clean, with very little noise. Large amount of noise are found on the smoke regions in the algorithms where the dense reconstruction was performed with MVE, such as in figures A.12c, A.12e and A.12f. The rest of the models present very little or no smoke at all. This corroborates the findings in the chapel model, where MVE was found to be the model most susceptible to noise by clouds in the horizon.

There is no visible amount of bleeding in the generated models, as well as no artefacts in sight. About completeness, COLMAP seems to do a good job reconstructing surfaces without

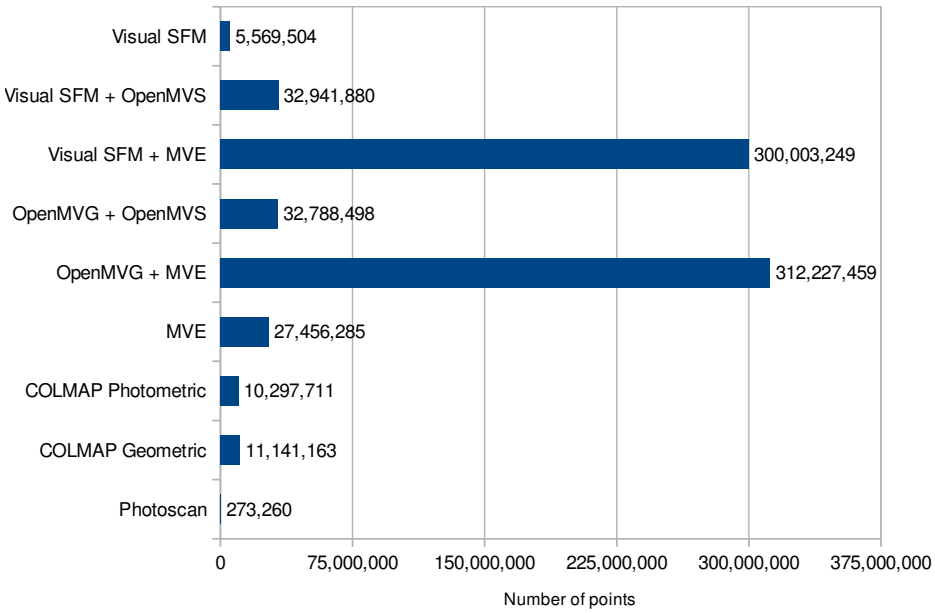


Figure 5.3: Point count per pipeline in the resulting point clouds for the practice hall at the firefighter school model.

texture, as found in the back wall of the building in figures A.13g and A.13h; specially compared to the model generated by Visual SFM in figure A.13a, where the back of the building is virtually missing.

Overall, it seems that openMVS is consistent with the size of the produced point cloud, without mattering if the sparse cloud was made with Visual SFM or with openMVG. But MVE does present a large difference on how it expands the dense cloud, because when the sparse cloud is generated with MVE it is significantly smaller than when imported from Visual SFM or openMVG.

5.2 Time performance

In this section the time that each stage took to process the data for each model will be discussed. It is expected that the execution time varies along the models depending on the amount of input images and if they contain a rich or poor scene in features.

Since three pipelines share Visual SFM for feature detection, feature matching and sparse reconstruction, two pipelines share openMVG for the same stages, and two other pipelines share COLMAP for the same stages again; only five different pipelines exist for the first three stages, and the tables and figures have been adjusted to reflect this.

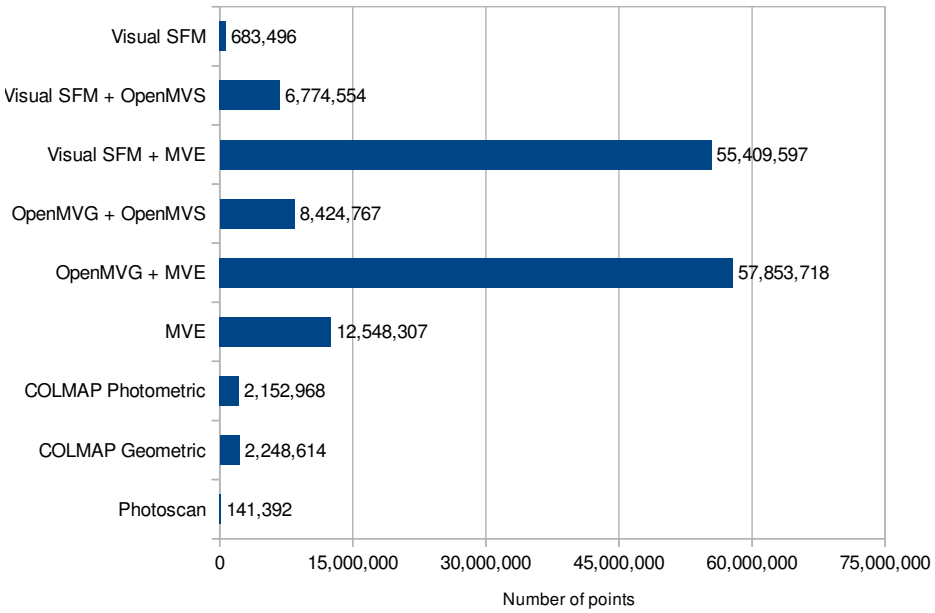


Figure 5.4: Point count per pipeline in the resulting point clouds for the burning house at the firefighter school model.

5.2.1 Feature detection

The first step is naturally the feature detection stage. Overall this stage is finished very quickly by all pipelines, in contrast to later stages. The execution time is shown in Table 5.1 and is given in seconds, and it is as well shown in figure 5.5 and given in hours.

The execution time seems to be linearly dependent on the amount of images each dataset posses, specially with Visual SFM and COLMAP, where the time it took for the robotics hall dataset is similar to the one to the chapel model, it is almost three times as large for the practice hall and it is approximately a third for the burning house. Comparing the times between the robotics hall and the chapel datasets with the openMVG and MVE algorithms, they seem to be dependent as well on how many features are found in the input images; as given approximately the same amount of images, the times for the chapel model, which is expected to be more rich in features than the robotics hall dataset, have a higher execution time.

Another remark is the time taken by MVE to detect features, that is significantly higher than the rest of the pipelines, given the same dataset. This might be because MVE performs SIFT and SURF feature detection, in contrast with the rest of algorithms that only perform SIFT detection.

5.2.2 Feature matching

The feature matching step is shown in Table 5.1 in seconds, and it is as well shown in figure 5.5 and given in hours. The trend in this stage is that Visual SFM and COLMAP are the fastest

Table 5.1: Execution times for the feature detection stage for all pipelines, given in seconds.

Pipeline	Robotics Hall	Chapel	Practice Hall	Burning House
Visual SFM	249	240	598	60
OpenMVG	481	792	1,224	164
MVE	1,280	1,733	3,190	437
COLMAP	198	212	373	65
Photoscan	396	481	373	66

Table 5.2: Execution times for the feature matching stage for all pipelines, given in seconds.

Pipeline	Robotics Hall	Chapel	Practice Hall	Burning House
Visual SFM	6,163	8,778	8,689	866
OpenMVG	236	187	629	60
MVE	25,734	42,694	39,394	603
COLMAP	5,836	11,430	7,150	726
Photoscan	26,609	40,618	89,715	9,986

algorithms for matching the features across all image pairs, taking approximately equal time to finish the task; and MVE and PhotoScan take the longest, specially PhotoScan for the practice hall and burning house datasets. A special observation is later made about openMVG, since the times are extremely low when compared against other algorithms.

A possible explanation on why MVE has such a large difference in execution time is that it has to match both SIFT and SURF descriptors between images, and that it constructs feature tracks for the same feature across multiple images, whereas the other algorithms match SIFT features in only a pair of images. Unfortunately the procedure implemented by PhotoScan is not documented. The documentation of COLMAP indicates that it runs this stage in the GPU, but there seems to be no real improvement over Visual SFM, which runs this step in the CPU; although than can be dependent on the used hardware.

There seems to be a correlation between runtime and to the amount of features each dataset possesses. For example with the robotics hall and chapel datasets, with similar number of images but with a different expectation of the number of features. It is clear that the matching stage takes significantly longer for the chapel dataset, and it is almost similar to the time taken for the practice hall, that has almost three times as many input images; although this reasoning does not apply for the PhotoScan pipeline.

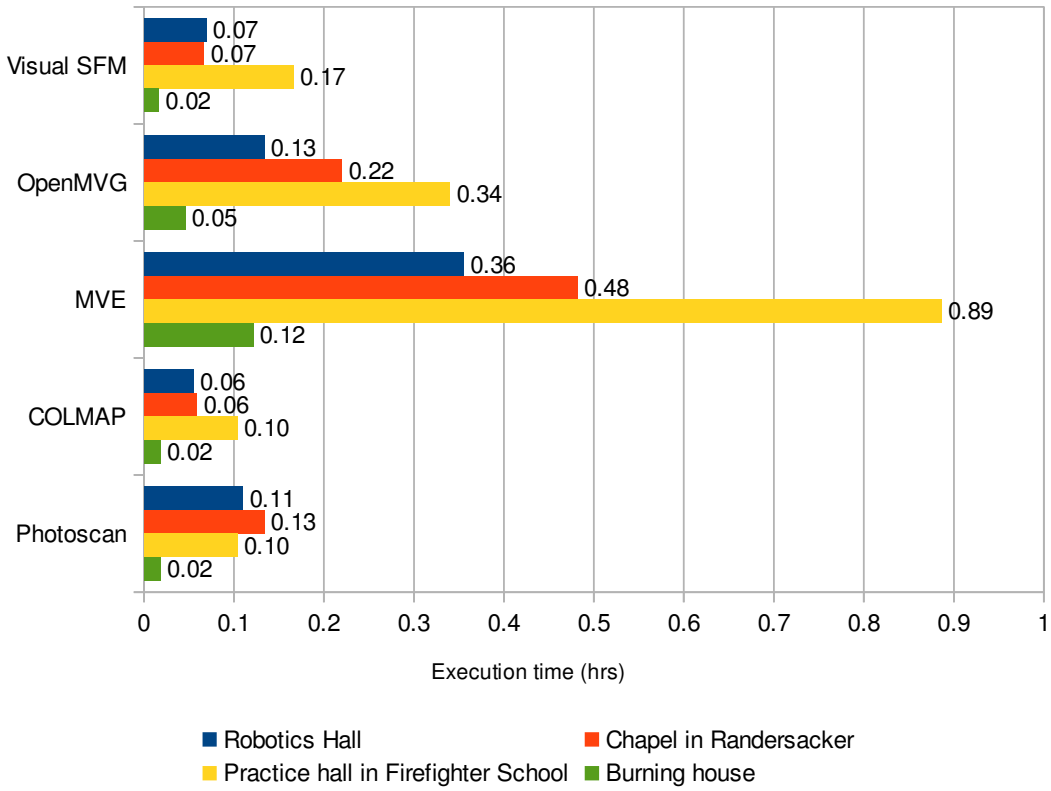


Figure 5.5: Total run time for all pipelines at the feature detection stage in hours.

5.2.3 Sparse reconstruction

On the sparse reconstruction step, presented in Table 5.3 and in figure 5.7, is clear that the simplifications made on how Visual SFM handles triangulation and bundle adjustment give a significant advantage over the other algorithms. The openMVG and MVE pipelines seem have a correlation between execution time and the number of generated features, as the robotics hall model is finished significantly faster than the chapel model for both pipelines. COLMAP and Visual SFM seem to be not sensitive to the number of generated features, as the robotics hall and chapel models finish in similar amounts of time.

OpenMVG presents an abnormal behaviour, because it was found during this work that it does not really separate the feature match and sparse reconstruction steps as the other algorithms do. Instead, openMVG does a preemptive (low-resolution) feature match in what the developers call the feature matching stage. But since a list of image pairs for the matching step is provided, this steps finishes after a couple of minutes and in the sparse reconstruction stage, the main loop includes a pair match step before the triangulation, incremental structure from motion and bundle adjustment steps, which are the stages modified to follow the “a contrario” methodology. Therefore is impossible to compare individually the feature match and sparse reconstruction

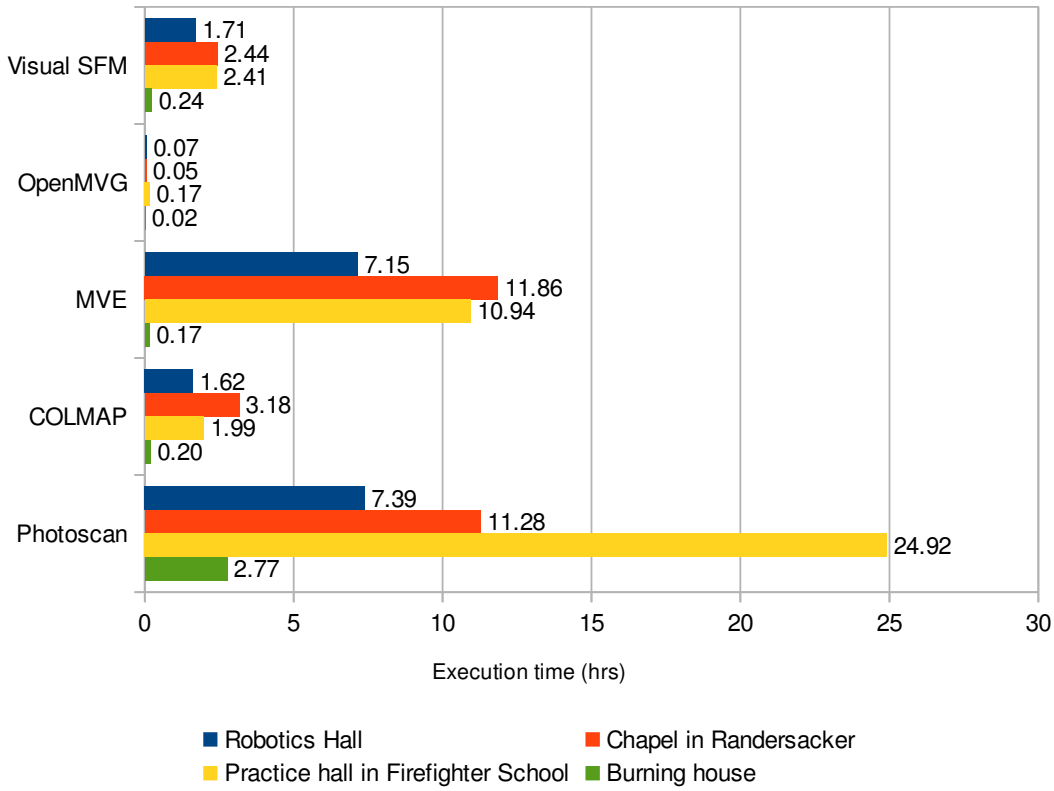


Figure 5.6: Total run time for all pipelines at the feature matching stage in hours.

stages of openMVG. Table 5.4 presents the execution time summarized in hours for both feature match and sparse reconstruction stages, to allow for comparing openMVG. From there it can be seen that the execution times of openMVG lie close to the times of COLMAP, except for the practice hall dataset, where it is significantly slower.

5.2.4 Dense reconstruction

Finally the dense reconstruction stage is presented in Table 5.5 and in figure 5.8. Interestingly it was found that MVE finishes relatively fast when expanding a sparse point cloud generated by MVE, and is actually one of the fastest; but when working from a project previously generated by Visual SFM or openMVG, it can take up to twenty five times as long to finish. It is not clear in the execution logs that extra steps are made during the dense reconstruction if the sparse point cloud was not generated by MVE, but it seems to be related to the fact that it also generates much larger point clouds when importing sparse clouds from Visual SFM or openMVG than when expanding a MVE sparse cloud. In this aspect openMVS is more consistent, taking similar amounts of times to expand a sparse point cloud created by Visual SFM or openMVG; which also is consistent with the generated point cloud sizes.

Table 5.3: Execution times summarized for the sparse reconstruction stage for all pipelines, given in seconds.

Pipeline	Robotics Hall	Chapel	Practice Hall	Burning House
Visual SFM	958	999	4,238	231
OpenMVG	9,351	24,261	65,334	906
MVE	2,275	5,871	4,383	1,212
COLMAP	6,588	7,927	21,915	1,602
Photoscan	3,711	2,028	8,365	508

Table 5.4: Execution times for the feature detection and sparse reconstruction stage for all pipelines, given in hours.

Pipeline	Robotics Hall	Chapel	Practice Hall	Burning House
Visual SFM	0.30	2.72	3.59	1.98
OpenMVG	0.27	6.79	18.32	2.66
MVE	0.50	13.49	12.16	7.78
COLMAP	0.65	5.38	8.07	3.45
Photoscan	2.92	11.85	27.24	8.42

Another remark is that COLMAP enforcing geometric constraints is a two-pass process, the first one with photometric constraints and the second one that enforces the geometric constraints, and each pass lasts approximately the same amount of time. This makes COLMAP with geometric constraints one of the longest running dense reconstruction process. This durations are almost comparable to the one from PhotoScan for the robotics hall and chapel datasets, but PhotoScan generates much less points.

5.2.5 Total execution time per dataset

In figures 5.9, 5.10, 5.11 and 5.12 is aggregated the total execution time for the robotics hall, chapel, practice hall and burning house datasets respectively, for all executed pipelines.

For the robotics hall dataset it takes approximately the same amount of time (about 4 hours) to execute the Visual SFM, Visual SFM + openMVS and openMVG + openMVS pipelines, followed by MVE with almost twice the execution time. The pipeline that took the longest to execute was PhotoScan with almost 33 hours but followed very closely by COLMAP with geometric constraints with almost 30 hours of execution time.

The chapel dataset models are also generated the fastest by Visual SFM and Visual SFM + openMVS with about 4 hours. In this case, however, the long time it takes for openMVG to

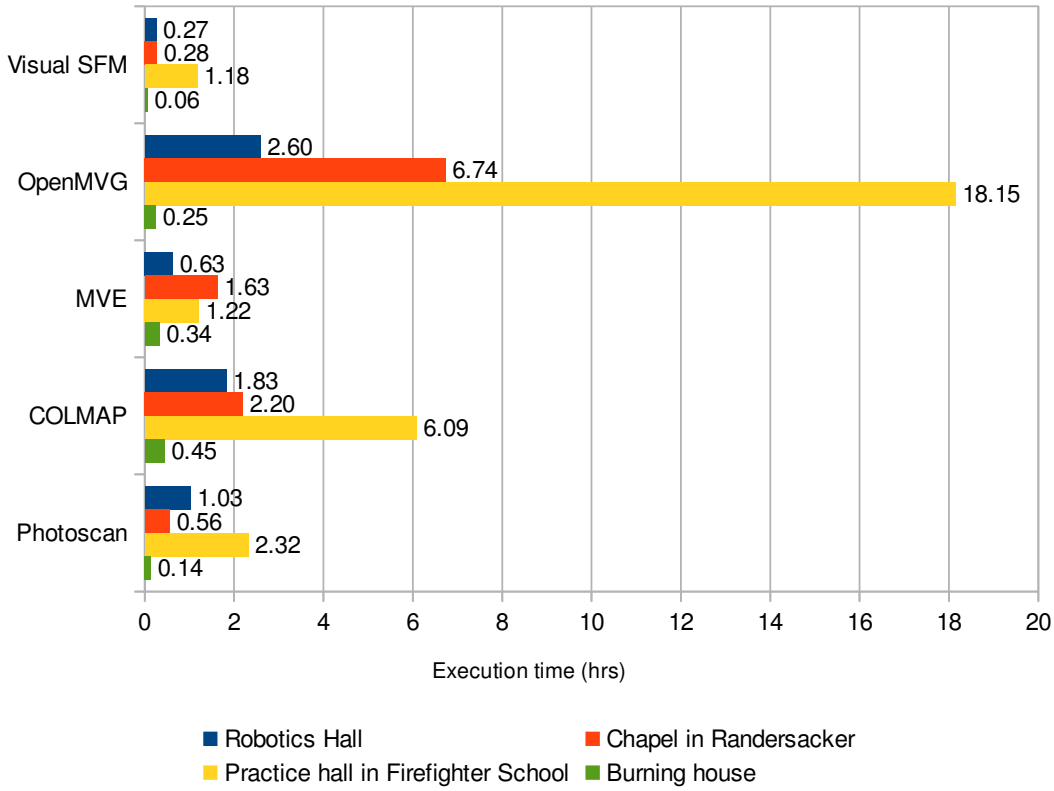


Figure 5.7: Total run time for all pipelines at the sparse reconstruction stage in hours.

Table 5.5: Execution times for the dense reconstruction stage for all pipelines, given in seconds.

Pipeline	Robotics Hall	Chapel	Practice Hall	Burning House
Visual SFM	4,553	2,958	18,004	1,108
Visual SFM + OpenMVS	5,159	5,166	10,191	1,451
Visual SFM + MVE	34,717	27,420	51,420	7,796
OpenMVG + OpenMVS	4,998	4,990	8,709	1,419
OpenMVG + MVE	51,038	36,770	85,689	7,930
MVE	1,761	5,020	6,110	1,853
COLMAP Photometric	47,820	42,530	105,871	14,829
COLMAP Geometric	94,800	87,050	209,407	30,069
Photoscan	87,525	92,871	135,817	6,020

generate the sparse point cloud due to the amount of features it has, makes it run twice as long than the fastest models. In this case, PhotoScan is again the slowest pipeline to generate the model.

In the case of the practice hall dataset, a similar tendency is seen as with the chapel, where Visual SFM, Visual SFM + openMVS are among the fastest models with about 8 hours of execution time. In this case, COLMAP with geometric constraints was the slowest generating pipeline with about 66 hours, closely followed by PhotoScan with 65 hours.

The burning house dataset presents a similar situation as with the robotics hall model. The fastest pipelines are Visual SFM, Visual SFM + openMVS and openMVG + openMVS with about 0.70 hours (40 minutes) of execution. The slowest algorithm was COLMAP with geometric constrains, totalling 9 hours of execution.

5.3 Cloud to cloud error analysis

For analysing the cloud to cloud error distance of the generated models against the reference laser scanner model, the error distance distribution histogram has been extracted from CloudCompare, as well as fitting the error distribution to a Weibull probability distribution; where the expected value and variance can be calculated for comparison purposes. Also it has been measured from the generated point cloud at what minimum error distance lie 50%, 75%, 90%, 95% and 99% of the points in the models, as to give an idea to the reader about how spread is the error distribution. The generated models are shown again in Appendix B, but where every point is coloured with a colour map according to the error distance associated to each point.

5.3.1 Robotics hall

The generated models of the robotics hall dataset are shown again in figures B.1, B.2 and B.3. This are the same views as the ones presented in Appendix A, but with a colour map representing the distance error between the generated model for each pipeline and the frame of reference generated by the laser scanner. The colour map codifies error distances between 0 and 2 m. Error distances greater than 2 m are truncated to 2 m, since at that scale the error is just too large for the actual value to be worth of analysis. The first image in these figures is the same view but of the laser scanner model, coloured with the reflectance data recorded by the scanner, as to present to the reader against what where the models compared to.

An observation worth mentioning from these figures is where it seems to be the most error in the generated models. For models that are very noisy, like the one generated by the Visual SFM + MVE pipeline in figure B.1d, the largest error seems to be mainly locaded on the roof plane, emanating in every direction from the roof. This holds true for the models generated by MVE in the dense reconstruction stage (figure B.1f and B.1g). Also the models generated by COLMAP (B.1h and B.1i) seem to have the largest error distributed in the same way.

In Table 5.6 the parameters from the estimated Weibull distribution for all the error histograms of all pipelines are shown, as well as the expected error value and standard deviation, according to the estimated distribution. In figure 5.13 the error histograms and the estimated Weibull distribution curve are plotted for all generated models. In figure 5.14 it is shown at what distances are located given percentages of amount of points on the model.

Table 5.6: Weibull distribution shape parameters (k), scale parameters (λ), expected value ($E(x)$) and standard deviation (σ) for the tested pipelines, corresponding to the error distribution of the robotics hall model.

Pipeline	Shape (k)	Scale (λ)	Expected value	Std. Deviation (σ)
Visual SFM	0.77	0.09	0.105	0.138
Visual SFM + openMVS	1.02	0.11	0.109	0.107
Visual SFM + MVE	0.97	0.06	0.061	0.063
openMVG + openMVS	0.97	0.04	0.041	0.042
openMVG + MVE	0.91	0.03	0.031	0.035
MVE	0.99	0.09	0.090	0.091
COLMAP Photometric	0.89	0.09	0.095	0.107
COLMAP Geometric	0.87	0.10	0.107	0.124
PhotoScan	0.86	0.13	0.140	0.164

From this data is possible to infer that the models with the less amount of error are generated with openMVG, which corresponds to the openMVG + MVE and the openMVG + openMVS pipelines, with an expected value of error of 3.1 cm and 4.1 cm respectively; and they posses the lowest spread of error, since the standard deviation is of 3.5 cm and 4.2 cm respectively. Although in figure B.1f the model generated by openMVG + MVE looks very noisy, the estimated distribution indicates that most points lie under 6.6 cm of error distance (that is, expected value plus standard deviation) and the measurements on the point clouds indicate that 90% of all the points have distance errors no larger than 9 cm. On the other hand, the point cloud generated by openMVG + openMVS looks cleaner in figure B.1e and the estimated distribution indicates that most points lie under 8.3 cm of distance error and the measurements indicate as well that 90% of all points also have errors no larger than 9 cm.

Considering the produced cloud size and time performance for both these pipelines, the openMVG + MVE pipeline generated the most points between the two, totalling 188 million points against 27 million generated by the other pipeline. But on execution time, the openMVG + openMVS performed better, generating the results in 4 hours and 11 minutes, whereas openMVG + MVE took almost 17 hours to finish. Therefore the openMVG + openMVS pipeline would be the preferred alternative to similar structures like the robotics hall dataset.

Although the Visual SFM pipeline was the fastest to finish, it has more than the double of expected error value than the most accurate pipelines, with 10.5 cm and a standard deviation of 13.8 cm. The pipeline that has the most spread error distribution is PhotoScan, where the expected value is of 14 cm with a standard deviation of 16.4 cm. This is accord to the measured values, where 90% of the points are found up to 33 cm away from the reference point cloud.

5.3.2 Chapel at Randersacker

The generated models of the chapel dataset with the error distance colour map are shown in figures B.4, B.5 and B.6. The most prominent location where error can be found is around the chapel's bell tower, specially for the MVE pipeline in figure B.5g. This model in particular features some error in the foliage as well.

Another point of interest is the error due to missing sections on the laser scanner model, like the roof of the chapel. In figure B.6 this shows as an inverted L shape in the roof of every model, since this data was not able to be recorded from the ground with the scanner.

The parameters of the estimated Weibull distributions for the error histograms are shown in Table 5.7, and the curves are plotted in figure 5.15. The minimum distances that contains certain percentages of points are shown on figure 5.16.

From this data it can be seen that the model with the smallest expected error value corresponds to the one generated by the Visual SFM + MVE pipeline, followed closely by the openMVG pipelines: openMVG + MVE and openMVG + openMVS, with an expected value of 8.4 cm, 10 cm and 10.3 cm respectively. These pipelines have a respective standard deviation of 11.6 cm, 12.3 cm and 15.3 cm. This seems to indicate that, due to the complexity of the dataset, the accuracy of the generated models is decreased and the error is spread wider. The measure data corroborates these estimations, as 90% of the points of the model generated with the Visual SFM + MVE pipeline have an error distance of no more than 20 cm, and the same for the model generated with the openMVG + MVE pipeline. For the model generated with openMVG + openMVG, this error distance increases to 28 cm.

If point count is taken into account, both the Visual SFM and openMVG + MVE pipelines generate the most points overall for the chapel dataset, with 143 million and 178 million points respectively. The openMVG + openMVS generates only 22 million points. When execution time is evaluated, the openMVG + openMVS is the fastest of the three, generating the model in 8 hours and 24 minutes, compared to 10 hours and 24 minutes for Visual SFM + MVE and 17 hours and 14 minutes for openMVG + MVE. Therefore, if an increase of 24% of execution time is affordable for the generation of a model similar to the chapel dataset, the Visual SFM + MVE dataset would be recommended for a gain in accuracy.

The highest expected value, and therefore spreader error distribution, was produced by PhotoScan, with an expected error value of 26.8 cm and a standard deviation of 35.2 cm; which is also the model that takes the longest to generate.

5.3.3 Practice hall at Firefighter School

The practice hall dataset seemed to be the one that produced the most noise, due to the amount of reflective or featureless surfaces, or the repetitive patterns of the building. In figures figures B.7, B.8, B.9 and B.10, the four views of the generated model are presented, with the error distance colour map for each point. From this images is worth making the clarification that the laser scanner frame of reference lacks the roof of the secondary structure of the building. The largest amount of error seems to be located in front of the reflective surfaces, both of the secondary structure (visible in figure B.8) and the big reflective surface on the back wall of the building (figure B.10). Also the algorithms generated points inside the structure, which are

Table 5.7: Weibull distribution shape parameters (k), scale parameters (λ), expected value ($E(x)$) and standard deviation (σ) for the tested pipelines, corresponding to the error distribution of the chapel at Randersacker model.

Pipeline	Shape (k)	Scale (λ)	Expected value	Std. Deviation (σ)
Visual SFM	0.71	0.09	0.112	0.162
Visual SFM + openMVS	0.75	0.14	0.167	0.226
Visual SFM + MVE	0.74	0.07	0.084	0.116
openMVG + openMVS	0.69	0.08	0.103	0.153
openMVG + MVE	0.82	0.09	0.100	0.123
MVE	0.80	0.12	0.136	0.171
COLMAP Photometric	0.78	0.12	0.139	0.179
COLMAP Geometric	0.78	0.12	0.139	0.179
PhotoScan	0.77	0.23	0.268	0.352

visible as large errors on figure B.9.

The parameters of the estimated Weibull distributions for the error histograms are shown in Table 5.8, and the curves for the histograms and error distributions are plotted in figure 5.17. The minimum distances that contains certain percentages of points are shown on figure 5.18.

The expected error values are larger than for the chapel dataset, the lowest being scored by the openMVG + MVE pipeline, followed closely by the openMVG + openMVS pipeline, with expected error of 25.5 cm and 26 cm. The standard deviation for these two pipelines are of 40 cm and 39.8 cm, which are substantially large than the mean value, therefore describing a very spread error distribution. Indeed the measurements indicate that 90% of the points in both models generated by these pipelines are well over a metre, with 1.73 m for the openMVG + MVE pipeline and 1.14 m for the openMVG + openMVS pipeline. This indicate that structure from motion algorithms are not suitable for structures with the characteristics exhibited by the practice hall, specially reflective surfaces and repetitive patterns.

The largest expected error value for this dataset was produced by the MVE pipeline, with an expected error value of 61.4 cm and a standard deviation of 57.4 cm. In the measurements it was found that 90% of the points for this model are up to 1.7 m away from the laser scanner frame of reference.

5.3.4 Burning house at Firefighter School

The burning house dataset is shown in figures B.11, B.12 and B.13 with error distance colour codification. From this figures it seems that the largest amount of error is due to the noise because of the smoke in the pipelines that were susceptible to this effect, specially the MVE pipeline (Figure B.13g).

There is also smoke error in the back of the building, in the Visual SFM + openMVS and openMVG + openMVS pipelines (Figures B.13c and B.13e), which is caused by the interaction

Table 5.8: Weibull distribution shape parameters (k), scale parameters (λ), expected value ($E(x)$) and standard deviation (σ) for the tested pipelines, corresponding to the error distribution of the practice hall at the firefighter school model.

Pipeline	Shape (k)	Scale (λ)	Expected value	Std. Deviation (σ)
Visual SFM	0.80	0.37	0.419	0.528
Visual SFM + openMVS	0.91	0.40	0.418	0.460
Visual SFM + MVE	0.83	0.29	0.320	0.388
openMVG + openMVS	0.68	0.20	0.260	0.398
openMVG + MVE	0.66	0.19	0.255	0.400
MVE	1.07	0.63	0.614	0.574
COLMAP Photometric	0.84	0.45	0.493	0.590
COLMAP Geometric	0.83	0.46	0.508	0.616
PhotoScan	0.88	0.56	0.597	0.680

of the smoke that rises under the shadow of the building and is suddenly visible due to the sun.

The parameters of the estimated Weibull distributions are shown in Table 5.9, and the curves for the histograms and error distributions are plotted in figure 5.19. The minimum distances that contains given percentages of points are shown on figure 5.20.

The estimated Weibull parameters for this dataset generate expected error values a bit higher than those found on the chapel dataset. The lowest expected error value was achieved by the openMVG + openMVS pipeline and the openMVG + MVE, which only differs in a couple of millimetres. The expected error values are 11.6 cm and 11.8 cm respectively, and the standard deviations are of 12.9 cm and 13.6 cm, which indicates that the error has a significant degree of spreading compared to the robotics hall or the chapel model, as a large standard deviation means that there is a significant amount of points spread to the right of the mean value in comparison to the points to the left, since the error cannot be negative. The measurement made to the point cloud indicate that 90% of the points in both the openMVG + openMVS and openMVG + MVE pipelines lie no further away than 23 cm and 25 cm respectively from the laser scanner frame of reference.

Taking into account the generated point size between the openMVG + openMVS and openMVG + MVE pipelines, the later generates a much larger point cloud with more than 55 million points, in contrast with the 8.4 million points generated by openMVG + openMVS. But in time performance, openMVG + openMVS is much faster, taking only 0.71 hours (42 minutes) to completely generate the model, compared to the almost 3 hours that it takes for openMVG + MVE to finish. Therefore the openMVG + openMVS would be strongly recommended in this case.

The highest expected error distance was produced by MVE, since is the model with the most noise due to the smoke. The expected error value is of 29.1 cm with a standard deviation of 29.4 cm.

Table 5.9: Weibull distribution shape parameters (k), scale parameters (λ), expected value ($E(x)$) and standard deviation (σ) for the tested pipelines, corresponding to the error distribution of the burning house at the firefighter school model.

Pipeline	Shape (k)	Scale (λ)	Expected value	Std. Deviation (σ)
Visual SFM	0.81	0.21	0.236	0.293
Visual SFM + openMVS	0.93	0.22	0.228	0.245
Visual SFM + MVE	0.99	0.14	0.141	0.142
openMVG + openMVS	0.90	0.11	0.116	0.129
openMVG + MVE	0.87	0.11	0.118	0.136
MVE	0.99	0.29	0.291	0.294
COLMAP Photometric	1.00	0.20	0.200	0.200
COLMAP Geometric	0.99	0.17	0.171	0.172
PhotoScan	0.87	0.21	0.225	0.260

5.4 Summary

5.4.1 Overall quality of the generated models

It was shown that the Visual SFM + MVE and openMVG + MVE pipelines consistently generate the largest point clouds for all the datasets. This is explained by the nature of the dense reconstruction step of MVE, that instead of triangulating image patches close to feature points, a full depth map is calculated for the input images. This virtually allows to place every pixel in the image into the produced model, but it has to be accompanied by a robust incremental structure from motion algorithm: since when running the MVE pipeline it was found to generate the noisiest models for all datasets as it failed to compute correctly the depth map, perhaps due to a not accurate enough camera pose and keypoints estimation; as well the models generated with MVE seem to be the most susceptible to artefacts produced by clouds or smoke.

The adaptative nature of the openMVG algorithm for feature match and triangulation paid off on the benchmarks, as it constantly scored among the most accurate pipelines for all datasets. The mathematical simplifications made by Visual SFM also seems to not impact severely on the quality of the generated model, as it also stood close to openMVG or even better in certain cases with the aid of MVE for the dense reconstruction.

COLMAP did not score among the most accurate pipelines, but it managed to generate correcly most of the texture-less surfaces in the robotics hall and burning house models with very little visible noise, as it uses an improved version of the PatchMatch algorithm that MVE implements, that allows for enforcing photometric constraints that manages to detect depth information subjectively better than MVE. No significant improvement could be measured or subjectively detected when examining the output model of COLMAP when enforcing geometric constraints. Therefore is not recommended to use this approach, as it consumes a significant amount of runtime in comparison to enforcing only photometric constraints.

PhotoScan did not manage to stand up to the other pipelines in this benchmark, either in point count, execution time or error evaluation. Although the produced point clouds are very clean, they are too sparse in comparison to the rest of algorithms, and the error analysis indicates that they contain large deformations that keeps the expected error distance higher than almost any other pipeline.

The encountered distance error also seems to be dependent on the complexity of the scanned structure as well. The expected error distances are lower for the robotics hall, which is the simplest model, and increase for the chapel dataset, where a lot of features and foliage exists, and also for the burning house, that contains curved roofs and a changing structure by the produced smoke that can generate some degree of occlusion.

All pipelines failed to some degree to generate correctly and accurately the practice hall model, as the generated errors are very high and the models are largely incomplete, specially on the white walls on the sides of the building. This is attributed to the high amount of reflective surfaces this structure contains, the repetitive patterns when flying in mid to close range, and the texture-less surfaces to the sides. All these characteristics make the task of triangulation difficult to solve, and therefore plenty of noise and artefacts are generated. Inclusive PhotoScan failed to integrate the back of the building into the generated model, that corresponds to the largest reflective surface.

5.4.2 Real time applications

Part of this work looked for evaluating the usability of a structure from motion pipeline for search and rescue applications on natural catastrophes or man-caused accidents, since the data collection is fairly simple and it can be achieved with low cost equipment, in contrast to ground surveying for example, where an expensive laser scanner is needed.

The execution times behaved at best following a linear time scale in accordance with the size of the input and the expected amount of detectable features in the dataset. At worst, it behaved non linearly for most pipelines, yielding execution times of up to 70 hours for some of them. In this regard, Visual SFM behaved consistently as the fastest algorithm for all pipelines and datasets. But its execution time is still too large for real time applications when working with medium and large datasets.

If a fast model is required, it is recommended to keep the number of input images to a minimum, and also reduce the resolution of the images for reducing the amount of features to be found and matched. It is also suggested to use some kind of preemptive matching algorithm, or to provide a custom list with the desired images pairs to match.

Another point to remark is the generated noise due to clouds or smoke that can affect the amount of noise presented in the generated models. In this work it was found that the expansion stage of MVE is the most susceptible algorithm to this kind of noise; therefore its usage in this scope of applications is not recommended. The rest of pipelines where expansion algorithms other than MVE where used, did not shown to be sensible to a moderate amount of smoke.

5.4.3 Installation and usability

In the previous chapter some issues that arose during the installation of the software packages, as well as during the usage for generating the resulting point clouds where documented. Therefore in this section it is described and rated in a subjective fashion how easy to install are the software packages, as well as what problems or inconveniences where found.

Visual SFM Requires manual installation of multiple dependencies of software not available in the Ubuntu repositories. Although guides exists for this purpose, they involve modifying source code and installing library files, and might be overwhelming for the inexperienced user. The model is generated with the click of a few buttons.

openMVG All dependencies are installed from the repositories, and the software downloads and compiles effortlessly from gitHub. Usage is only available by the command-line interface and although it is documented, it involves one command for every step, which may become confusing. The camera sensor size has to be added to an internal camera database.

openMVS Also installs effortlessly from gitHub. Only one command line is involved, and reads Visual SFM projects natively.

MVE Installs as well without problems. It is executed with just two commands, one for the feature detection, matching and sparse reconstruction stages and the other one for the dense reconstruction. However, it does not save progress for the first command until it is completed, which is a major disadvantage: if the process is stopped, it has to start all over again. It provides importers for both Visual SFM and openMVG projects.

COLMAP It required a special version of the Ceres library not available in the repositories for installation. The software is not difficult to use, but the desktop manager has to be shut down while the dense reconstruction stage is running, as it is too GPU intensive.

PhotoScan No installation required, and depends on software already available on most computers, like the Qt runtime library. Usage is also very simple, just a few buttons need to be pressed. But a major disadvantage is that it does not allow to save progress until the processing stage has finished. So if the software is stopped, it has to start all over again. Also a manual save is required, as by default it does not save automatically when it finishes, therefore the user has to be very careful to ensure to save the project after every step.

In Table 5.10 it is subjectively rated how easy or difficult is to install and to use the software packages employed in this work. The used key is described as follows: (++) very easy, (+) minor adjustments need to be made, (-) needs adjustments that require technical knowledge or presents inconveniences in respect to usability, and (--) major modifications are needed or usage documentation is not provided or completely broken.

Table 5.10: Qualitative rating of the used structure from motion algorithms.

Software package	Installation	Usability	GUI Provided
Visual SFM	-	++	Yes
openMVG	++	+	No
openMVS	++	++	No
MVE	++	-	Yes
COLMAP	+	+	Yes
PhotoScan	++	-	Yes

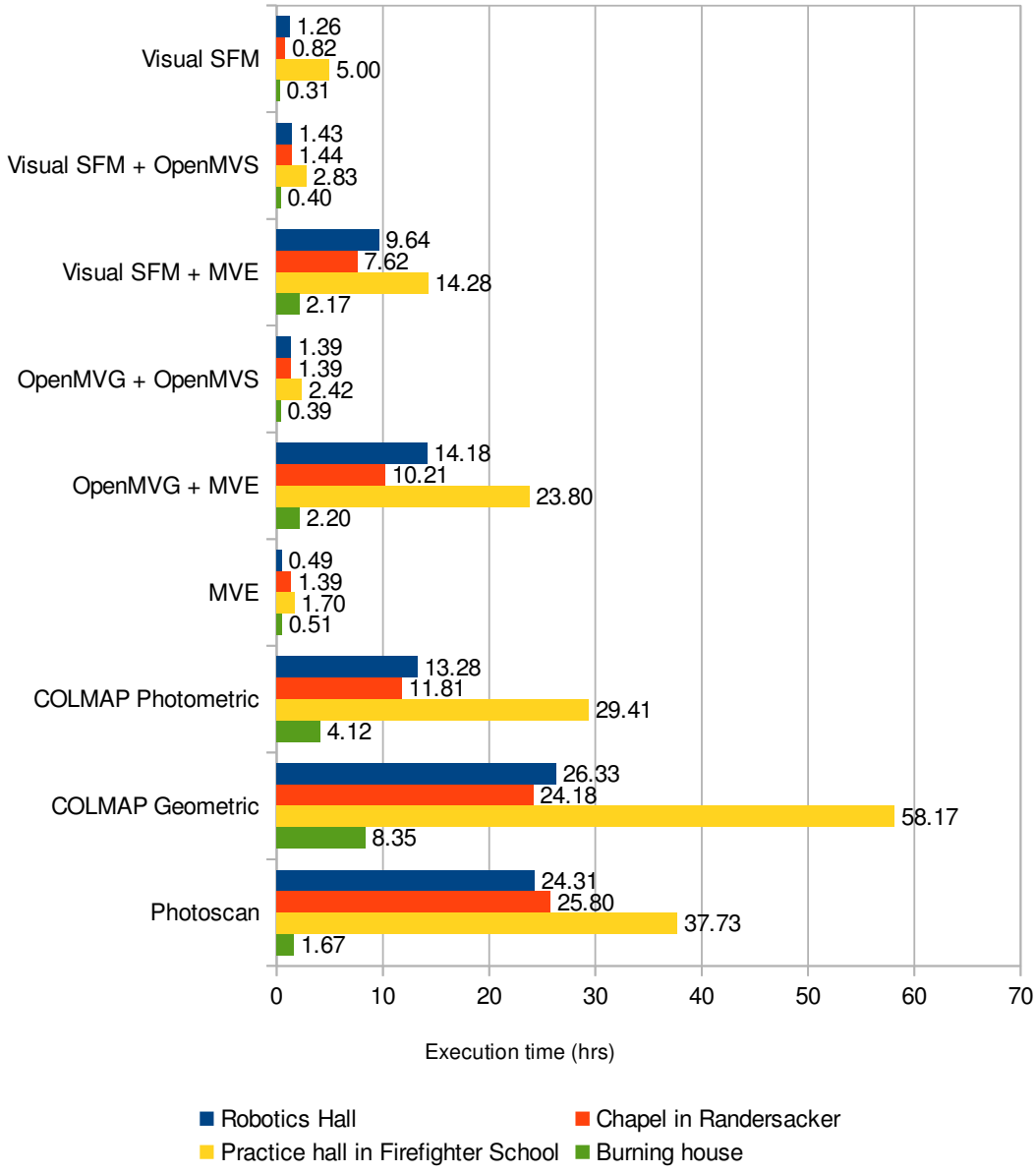


Figure 5.8: Total run time for all pipelines at the dense reconstruction stage in hours.

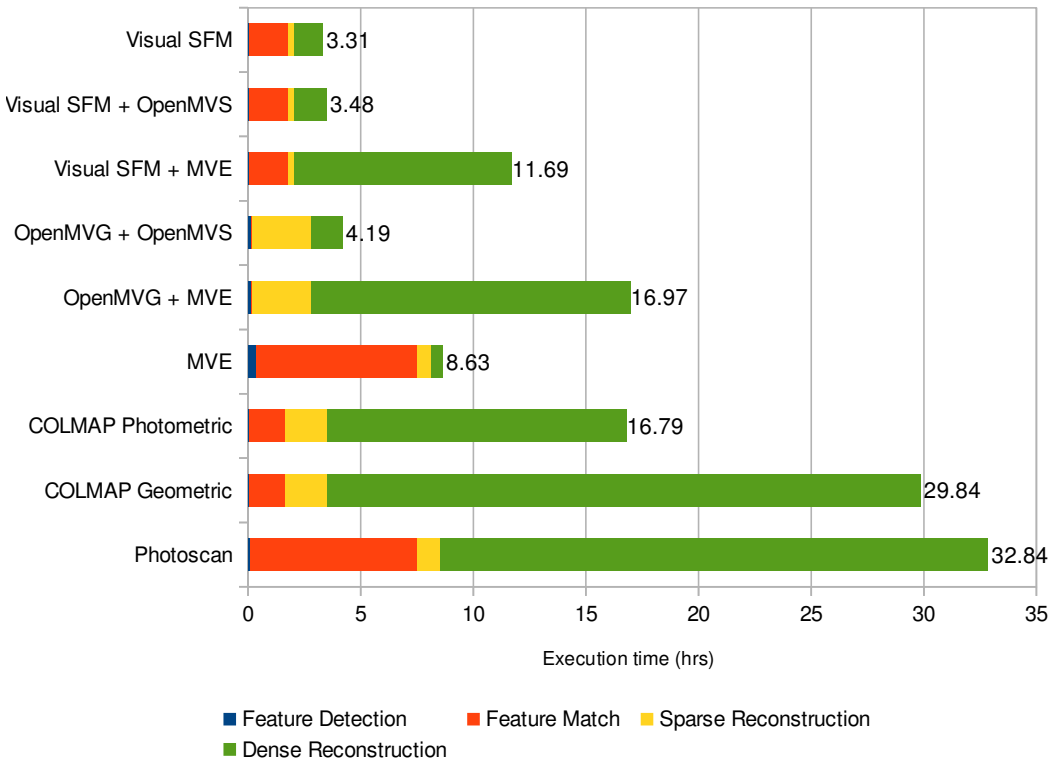


Figure 5.9: Total run time for the robotics hall pipelines.

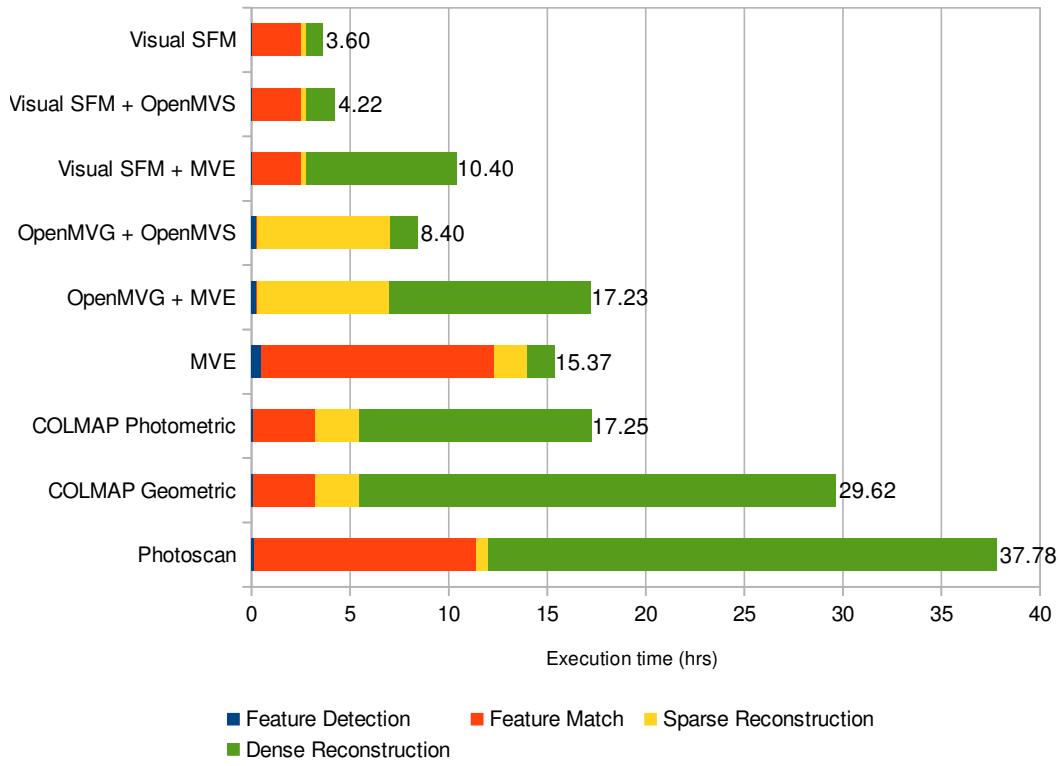


Figure 5.10: Total run time for the chapel at Randersacker pipelines.

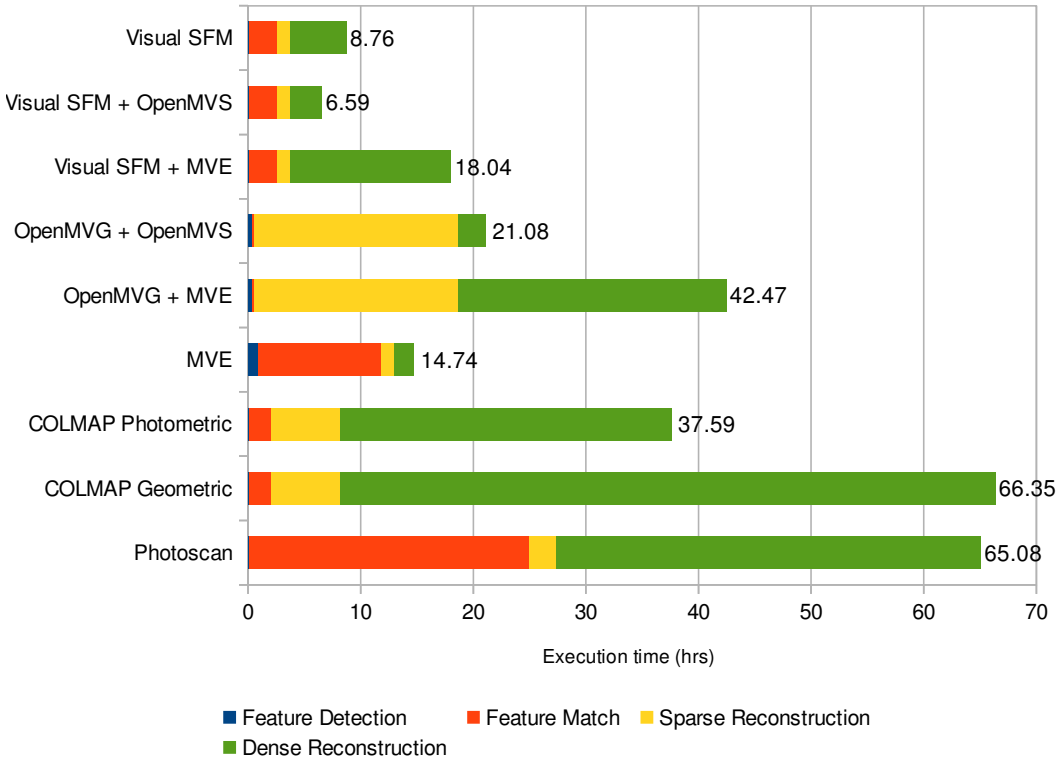


Figure 5.11: Total run time for the practice hall at the firefighter school pipelines.

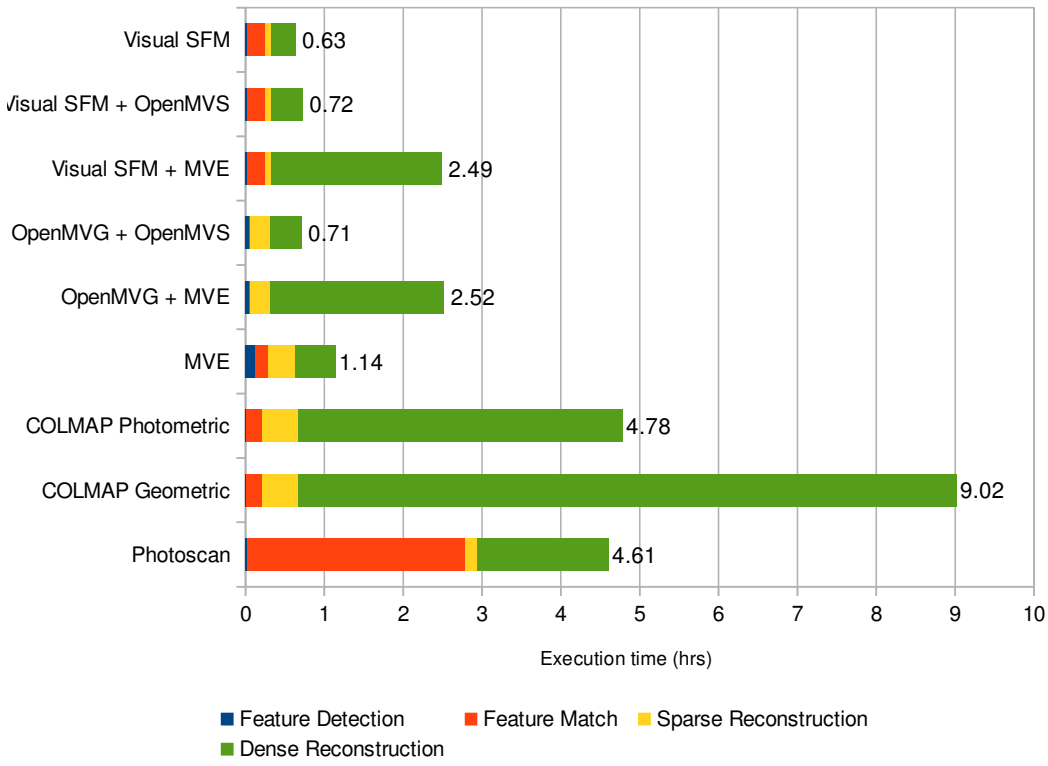
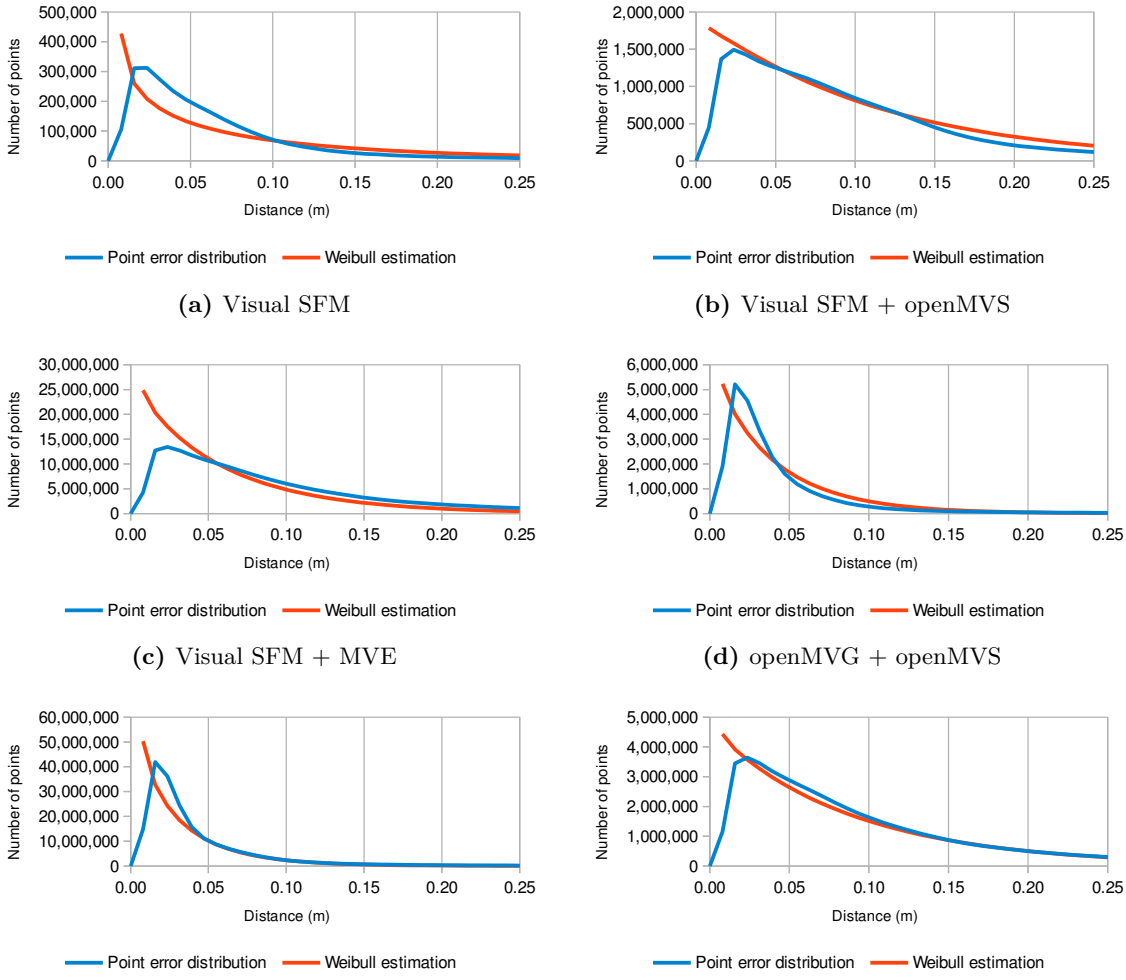


Figure 5.12: Total run time for the burning house at the firefighter school pipelines.



(a) Visual SFM

(b) Visual SFM + openMVS

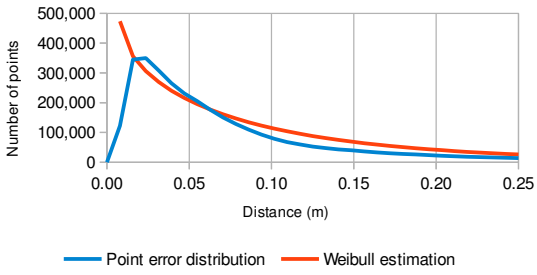
(c) Visual SFM + MVE

(d) openMVG + openMVS

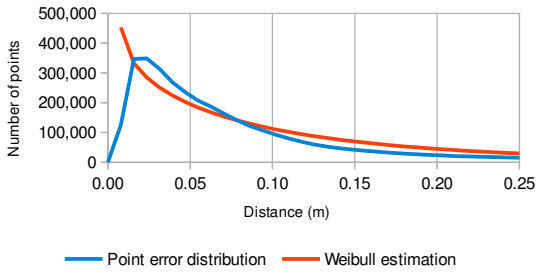
(e) openMVG + MVE

(f) MVE

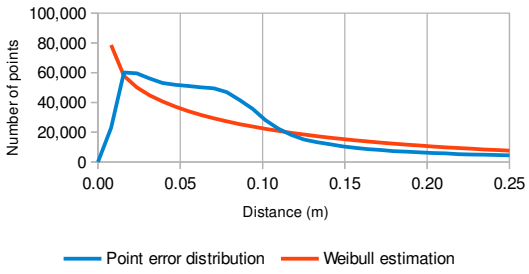
Figure 5.13: Error distribution in metres between the pipelines of the robotics hall model and the laser scanner reference frame. The Weibull distribution estimated for the error distribution is also shown in the graph. The horizontal axis scale is the same for all the graphs, and it has been cut up to a reasonable value as to make visualization of the distribution easier.



(g) COLMAP Photometric



(h) COLMAP Geometric



(i) PhotoScan

Figure 5.13: Error distribution in metres between the pipelines of the robotics hall model and the laser scanner reference frame. The Weibull distribution estimated for the error distribution is also shown in the graph. The horizontal axis scale is the same for all the graphs, and it has been cut up to a reasonable value as to make visualization of the distribution easier.

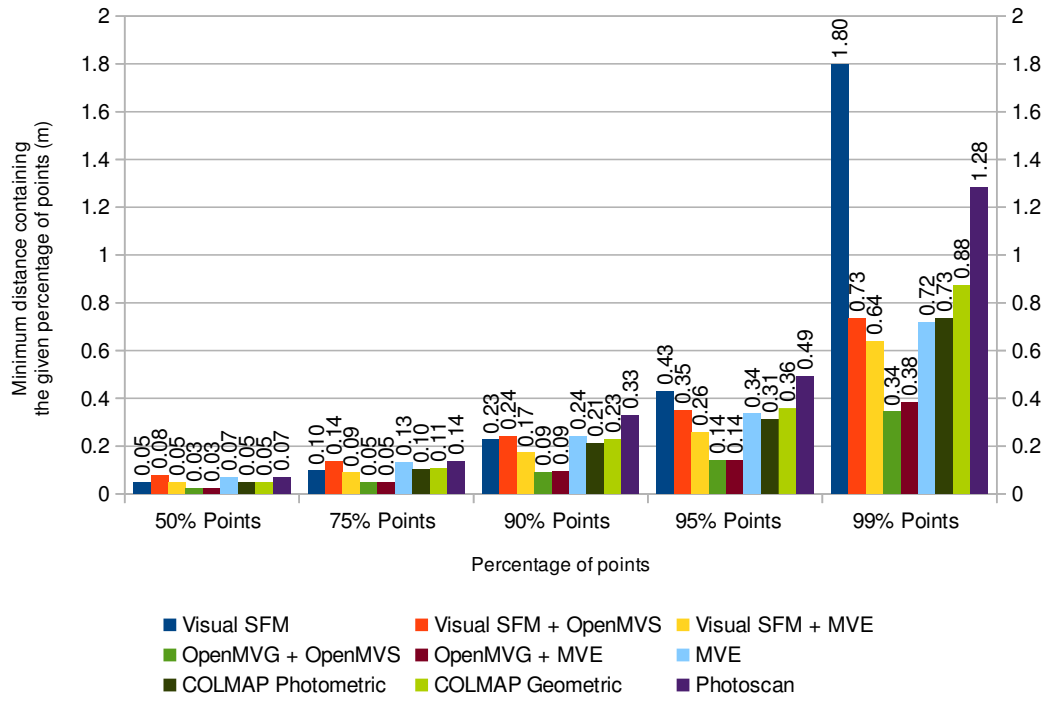
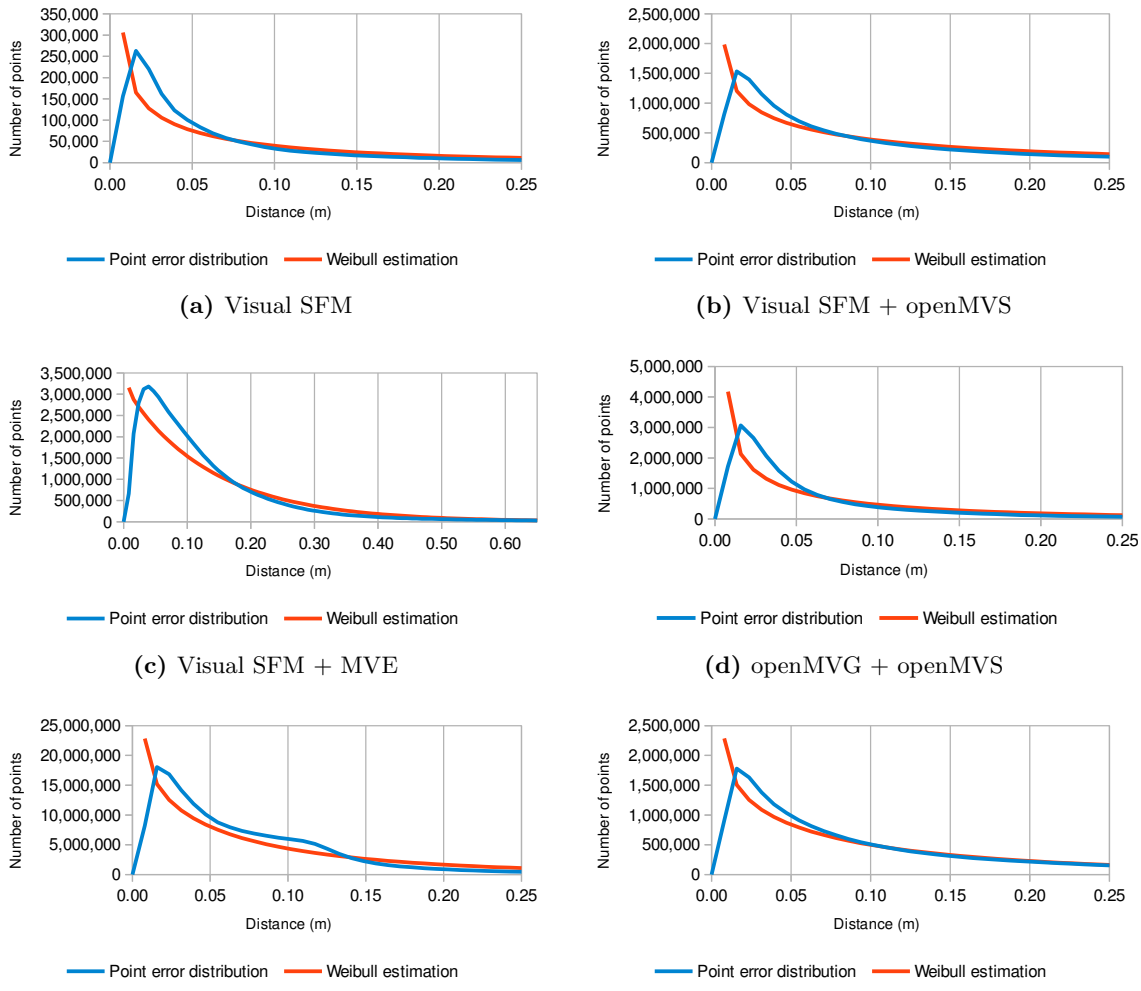


Figure 5.14: Minimum error distances containing at least 50%, 75%, 90%, 95% and 99% of all points in the generated robotics hall model per pipeline, when compared against the laser scanner frame of reference.



(a) Visual SFM

(b) Visual SFM + openMVS

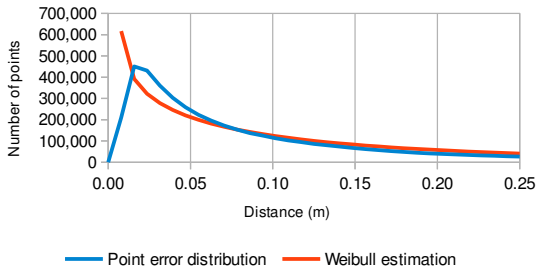
(c) Visual SFM + MVE

(d) openMVG + openMVS

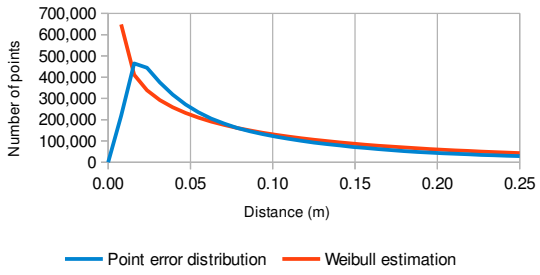
(e) openMVG + MVE

(f) MVE

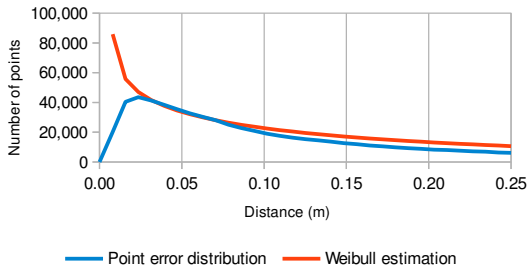
Figure 5.15: Error distribution in metres between the pipelines of the chapel model and the laser scanner reference frame. The Weibull distribution estimated for the error distribution is also shown in the graph. The horizontal axis scale is the same for all the graphs, and it has been cut up to a reasonable value as to make visualization of the distribution easier.



(g) COLMAP Photometric



(h) COLMAP Geometric



(i) PhotoScan

Figure 5.15: Error distribution in metres between the pipelines of the chapel model and the laser scanner reference frame. The Weibull distribution estimated for the error distribution is also shown in the graph. The horizontal axis scale is the same for all the graphs, and it has been cut up to a reasonable value as to make visualization of the distribution easier.

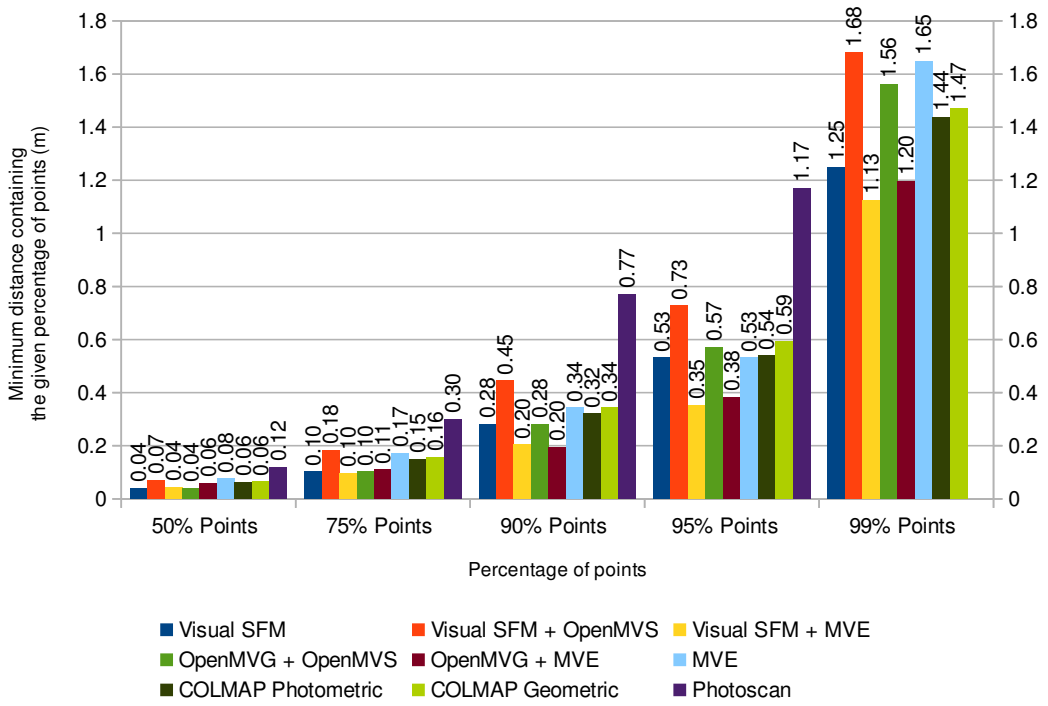
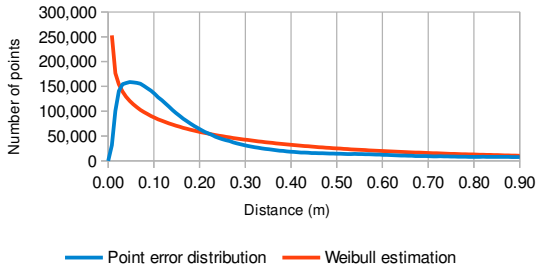
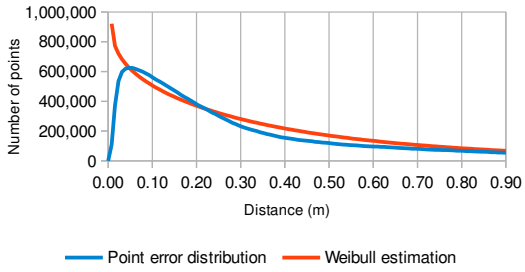


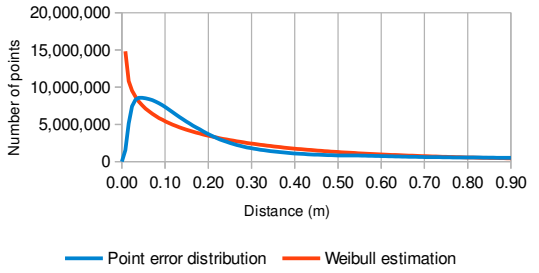
Figure 5.16: Minimum error distances containing at least 50%, 75%, 90%, 95% and 99% of all points in the generated chapel at Randersacker model per pipeline, when compared against the laser scanner frame of reference. Missing data indicates distances greater than 2 m.



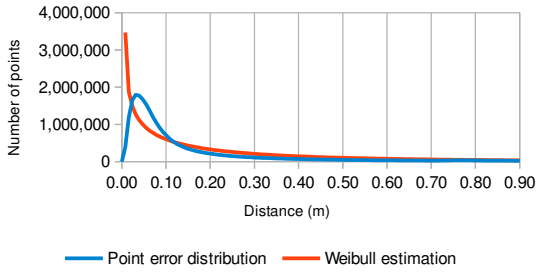
(a) Visual SFM



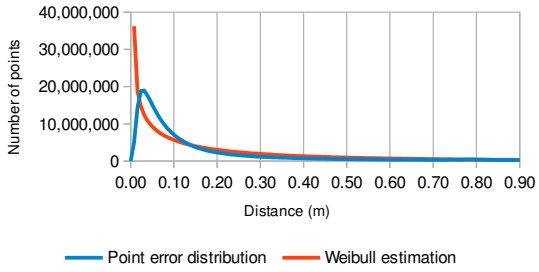
(b) Visual SFM + openMVS



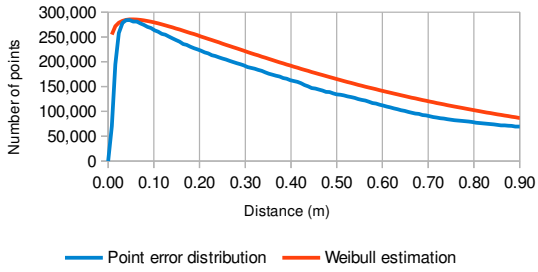
(c) Visual SFM + MVE



(d) openMVG + openMVS

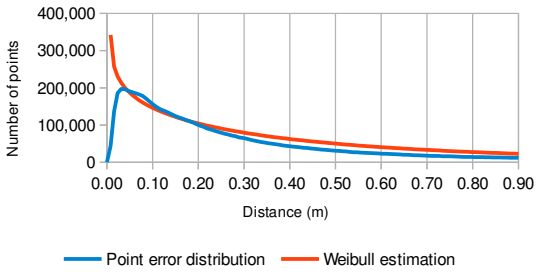


(e) openMVG + MVE

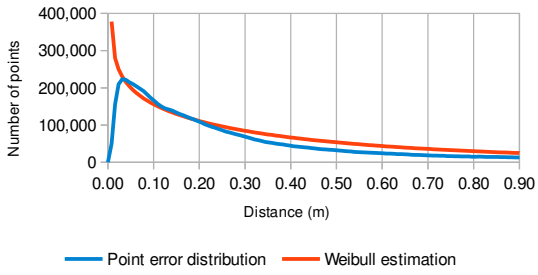


(f) MVE

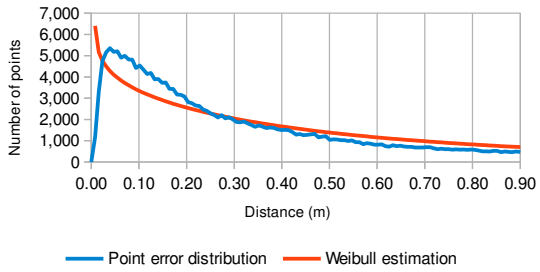
Figure 5.17: Error distribution in metres between the pipelines of the practice hall at the firefighter school model and the laser scanner reference frame. The Weibull distribution estimated for the error distribution is also shown in the graph. The horizontal axis scale is the same for all the graphs, and it has been cut up to a reasonable value as to make visualization of the distribution easier.



(g) COLMAP Photometric



(h) COLMAP Geometric



(i) PhotoScan

Figure 5.17: Error distribution in metres between the pipelines of the practice hall at the firefighter school model and the laser scanner reference frame. The Weibull distribution estimated for the error distribution is also shown in the graph. The horizontal axis scale is the same for all the graphs, and it has been cut up to a reasonable value as to make visualization of the distribution easier.

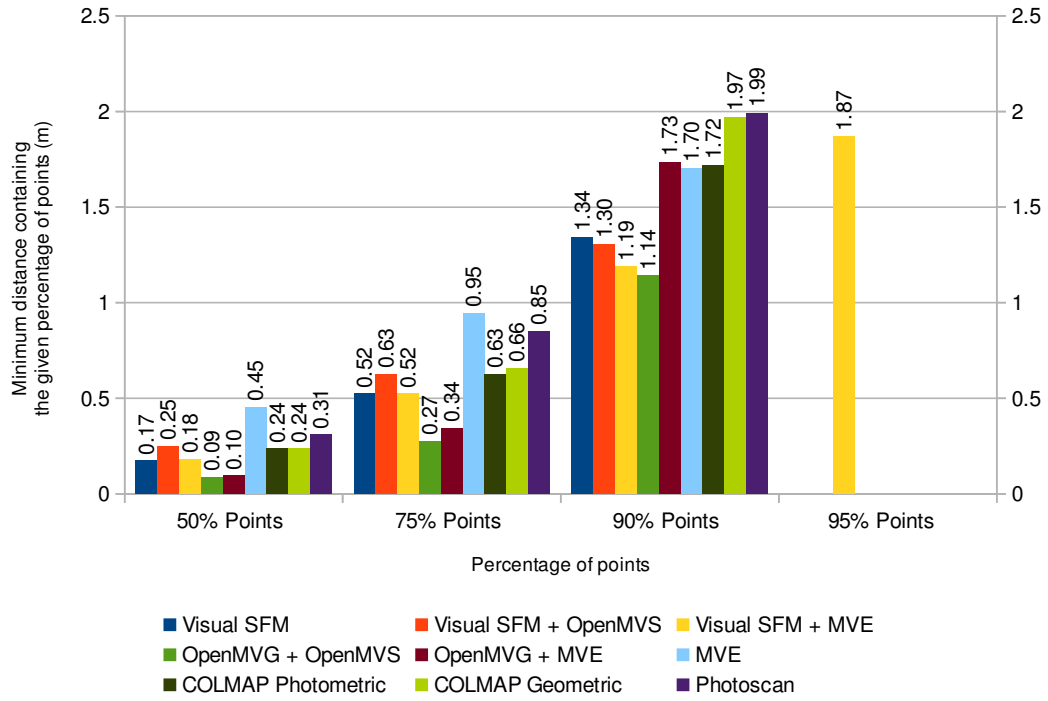
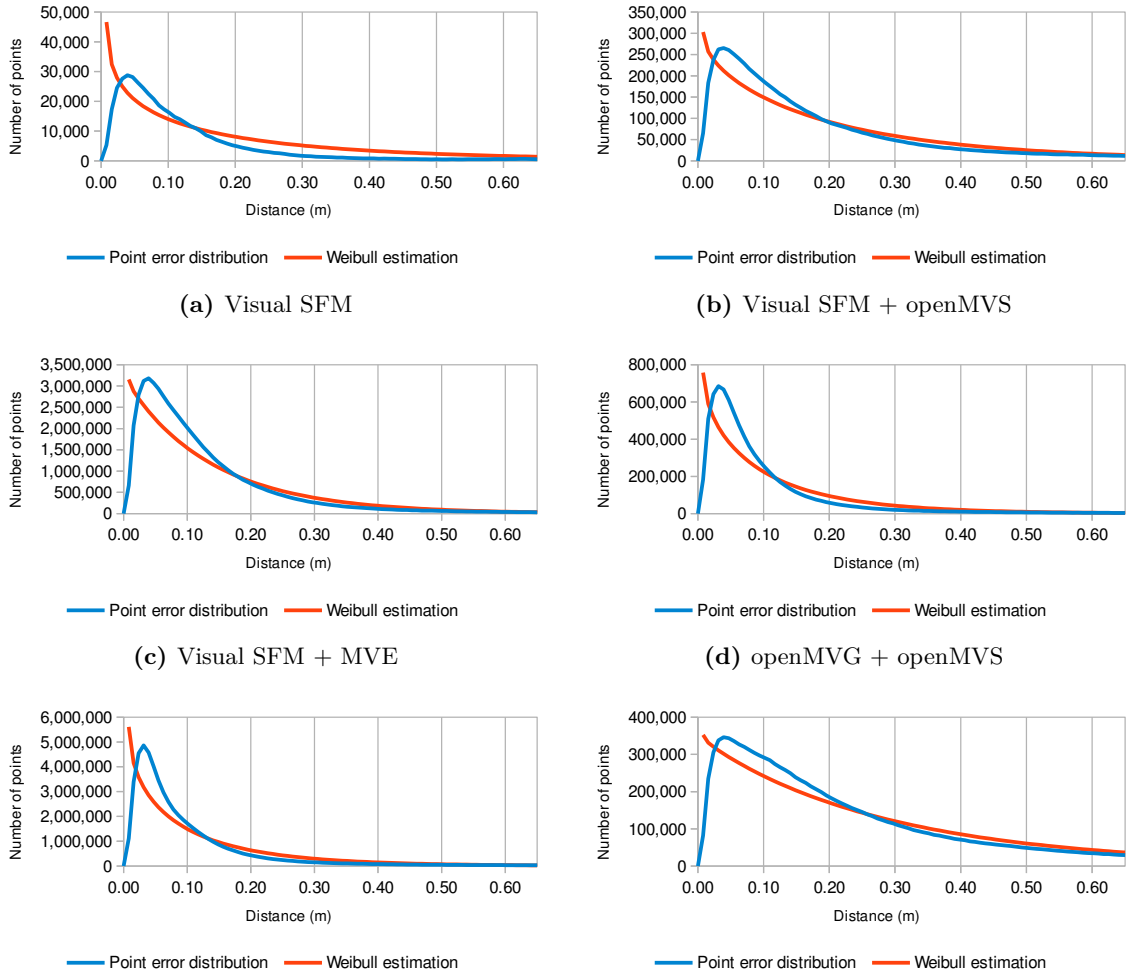


Figure 5.18: Minimum error distances containing at least 50%, 75%, 90% and 95% of all points in the generated practice hall at the firefighter school model per pipeline, when compared against the laser scanner frame of reference. Missing data indicates distances greater than 2 m.



(a) Visual SFM

(b) Visual SFM + openMVS

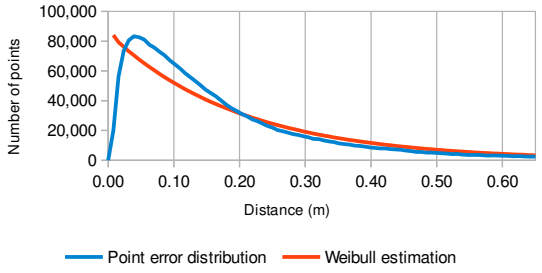
(c) Visual SFM + MVE

(d) openMVG + openMVS

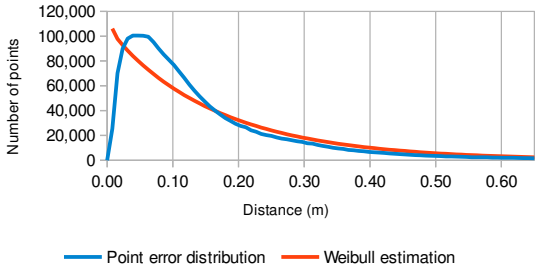
(e) openMVG + MVE

(f) MVE

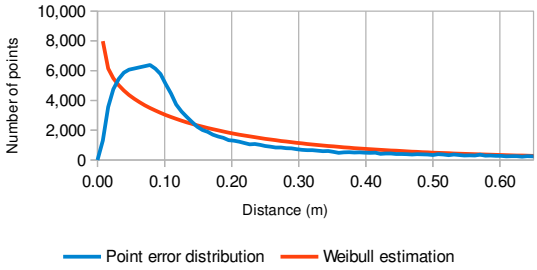
Figure 5.19: Error distribution in metres between the pipelines of the burning house at the firefighter school model and the laser scanner reference frame. The Weibull distribution estimated for the error distribution is also shown in the graph. The horizontal axis scale is the same for all the graphs, and it has been cut up to a reasonable value as to make visualization of the distribution easier.



(g) COLMAP Photometric



(h) COLMAP Geometric



(i) PhotoScan

Figure 5.19: Error distribution in metres between the pipelines of the burning house at the firefighter school model and the laser scanner reference frame. The Weibull distribution estimated for the error distribution is also shown in the graph. The horizontal axis scale is the same for all the graphs, and it has been cut up to a reasonable value as to make visualization of the distribution easier.

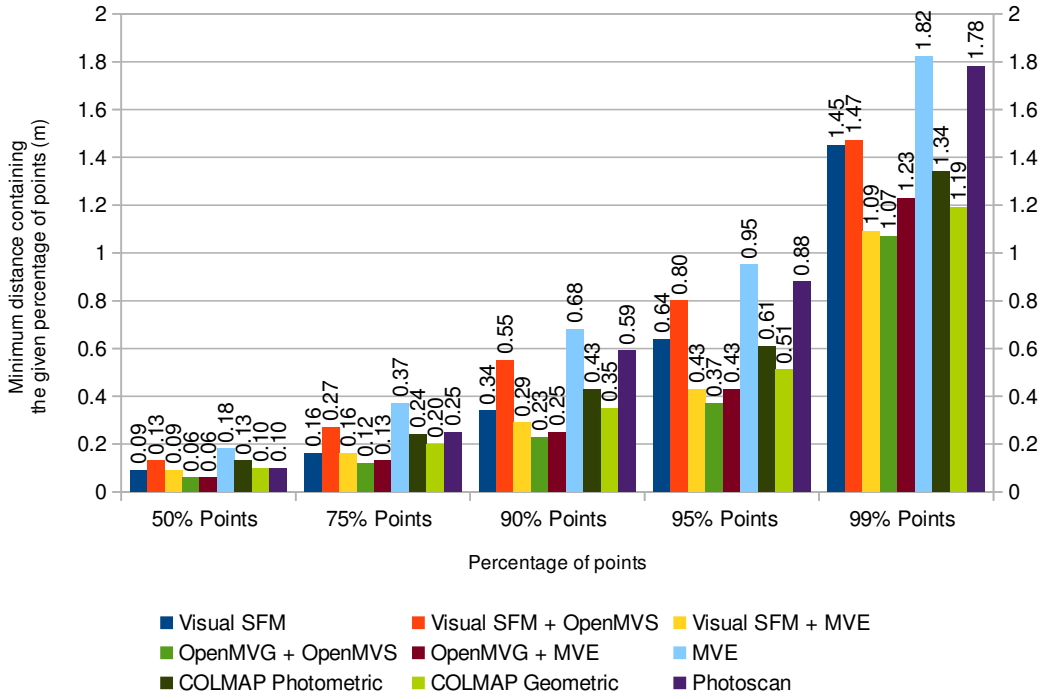


Figure 5.20: Minimum error distances containing at least 50%, 75%, 90%, 95% and 99% of all points in the generated burning house at the firefighter school model per pipeline, when compared against the laser scanner frame of reference. Missing data indicates distances greater than 2 m.

Chapter 6

Conclusions

In this work four datasets where recorded with both a terrestrial laser scanner and a drone with an built-in camera. The laser scanner data provides a 3D model of the digitalized datasets, and the video taken from the drone is used as input for nine different structure from motion pipelines.

The input images for all datasets where processed by the nine pipelines, and the execution times were recorded. The generated models where then registered against the frame of reference using transformations of translation, rotation and scale, preserving the relative scales between the x, y and z axis.

The pipelines where MVE was used as a dense reconstructor consistently produced the largest point clouds, and PhotoScan produced the most sparse point clouds. A downside of using MVE as a dense reconstructor is that it was found to be the most susceptible algorithm to noise due to clouds in the horizon or smoke.

The "a contrario" methodology of the openMVG algorithm for feature match and triangulation proved to increase the accuracy of the generated models, as the pipelines that were generated by openMVG constantly got the lowest expected distance error in comparison to the others, the less spread error distribution, and a execution time relatively fast when compared to other pipelines, although it was not the fastest.

Visual SFM also performs remarkably in accuracy, even considering the mathematical simplifications is subject to. The generated models by Visual SFM where computed faster than other pipelines, and when expanded with MVE, produced consistently one of the densest and accurate point clouds among all other pipelines.

The accuracy of the generated models depends on the complexity of the scene: it was found that simpler structures, like the robotics hall dataset, contain the smallest expected error distance achieving an accuracy of up to 3 cm. When the complexity of the scene is larger, such as with the irregular surfaces and foliage of the chapel dataset or the smoke artefacts on the burning house dataset, the expected error distance also increased. In the later cases, the accuracy decreased to 8.4 cm for the chapel dataset and 11.6 cm for the burning house.

The improved PatchMatch algorithm implemented by COLMAP for the dense reconstruction step allowed for a more complete generation of texture-less surfaces, thus providing less gaps in its generated models, but at the expense of execution time.

Structure from motion algorithms still fail to generate reflective surfaces and large surfaces without texture. No pipeline was able to generate a complete model for the practice hall at the Bavarian firefighter school. All structure from motion algorithms failed to generate the side walls of the buildings. The large reflective surfaces proved to be the source of many artefacts in the output models.

The large amount of execution time needed to compute the models does not allow for a real time application of the structure from motion yet, such as a disaster response data recollection method. It can be used as an auxiliary data recollection method though, as it can deliver results in less than one hour, if image count and size is kept low. It also allows for safely exploring a hard to reach area by ground, without risking human lives. Most pipelines where found to not being sensitive to moderate amounts of smoke in the scanned area.

6.1 Future work

Due to time constraints, is not possible to test all existing structure from motion software packages, and the distinct combinations of pipelines that mixes steps from different packages. Software like Bundler, MicMacs, 3DF Zephyr, Graphos and others where left out in order to not over-expand this work due to the time they might require to execute, and to keep the complexity of the benchmarks as low as possible. Therefore is recommended that future work includes different software packages that the ones used in this benchmark.

Among the tested software packages there exists more possible combinations of stages with different algorithms: for example, COLMAP allows to export a Visual SFM or MVE project after the sparse reconstruction step, thus allowing to make the dense reconstruction with either utility. Therefore it can be interesting to test these other alternatives.

Since there is a large variety of mesh generation and refinement algorithms available at the moment of writing, in this work it was excluded this step as to not increase the number of pipelines in the benchmarks, since the dense point cloud can be meshed by any meshing software. Naturally are implemented differently, thus potentially affecting the generated result. This step can drastically reduce error specially in the models with large point count, as the sparse error will be filtered out in favour of denser zones where the error has been shown to be low, like in the models generated by openMVG + MVE. Therefore is also advised to test what impact these steps may have on generated dense point clouds.

Appendix A

Views of the generated models

A.1 Robotics Hall

(a) Visual SFM

(b) Visual SFM + openMVS

(c) Visual SFM + MVE

(d) openMVG + openMVS

(e) openMVG + MVE

(f) MVE

Figure A.1: First view of the produced point clouds by all pipelines of the robotics hall model with the extracted colours. Models tend to accumulate most of the noise in the plane of the roof of the building. Artefacts generated by MVE are marked in red.

(g) COLMAP Photometric

(h) COLMAP Geometric

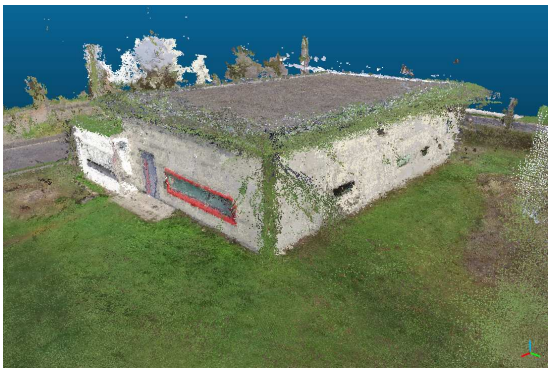
(i) PhotoScan

Figure A.1: First view of the produced point clouds by all pipelines of the robotics hall model with the extracted colours. Models tend to accumulate most of the noise in the plane of the roof of the building. Artefacts generated by MVE are marked in red.

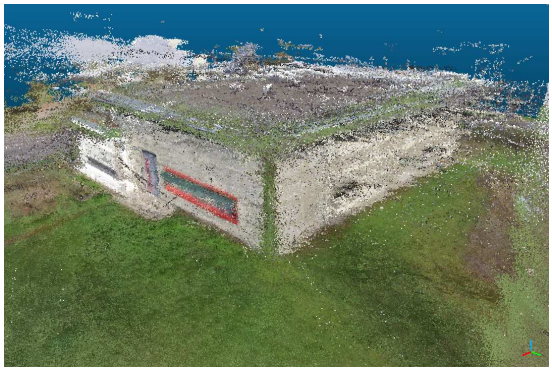
(a) Visual SFM

(b) Visual SFM + openMVS

(c) Visual SFM + MVE



(d) openMVG + openMVS



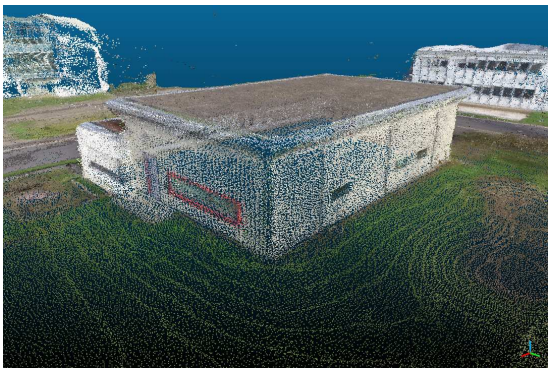
(e) openMVG + MVE

(f) MVE

Figure A.2: Second view of the produced point clouds by all pipelines of the robotics hall model with the extracted colours. Models show a tendency to colour bleeding due to the predominant green background of the grass.

(g) COLMAP Photometric

(h) COLMAP Geometric



(i) PhotoScan

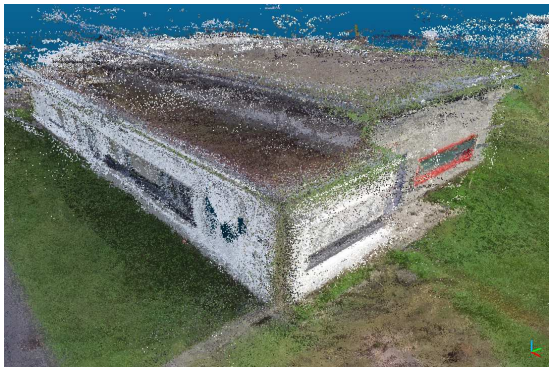
Figure A.2: Second view of the produced point clouds by all pipelines of the robotics hall model with the extracted colours. Models show a tendency to colour bleeding due to the predominant green background of the grass.

(a) Visual SFM

(b) Visual SFM + openMVS

(c) Visual SFM + MVE

(d) openMVG + openMVS



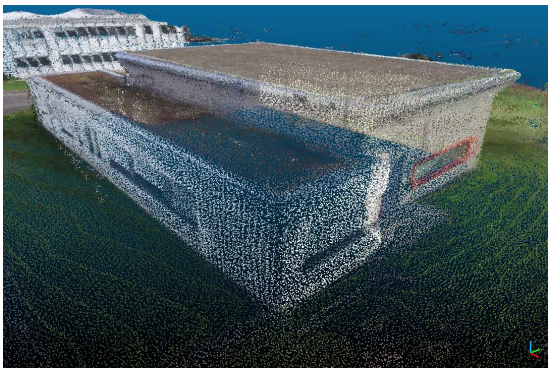
(e) openMVG + MVE

(f) MVE

Figure A.3: Third view of the produced point clouds by all pipelines of the robotics hall model with the extracted colours. Not all models where able to completely generate the featureless walls, like the Visual SFM pipeline.

(g) COLMAP Photometric

(h) COLMAP Geometric



(i) PhotoScan

Figure A.3: Third view of the produced point clouds by all pipelines of the robotics hall model with the extracted colours. Not all models where able to completely generate the featureless walls, like the Visual SFM pipeline.

A.2 Chapel at Randersacker

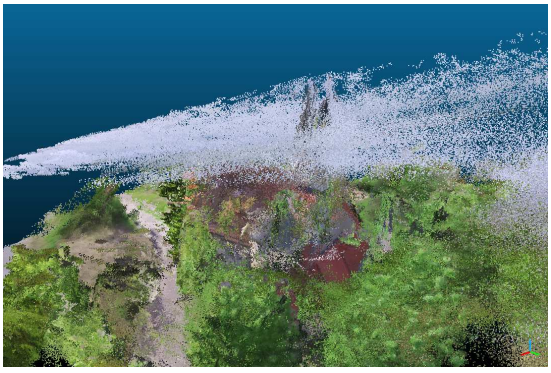
(a) Visual SFM

(b) Visual SFM + openMVS

(c) Visual SFM + MVE

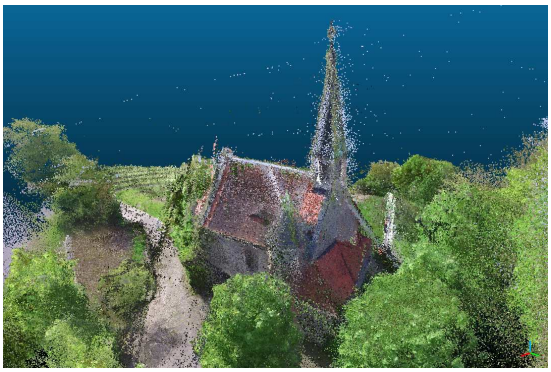
(d) openMVG + openMVS

(e) openMVG + MVE

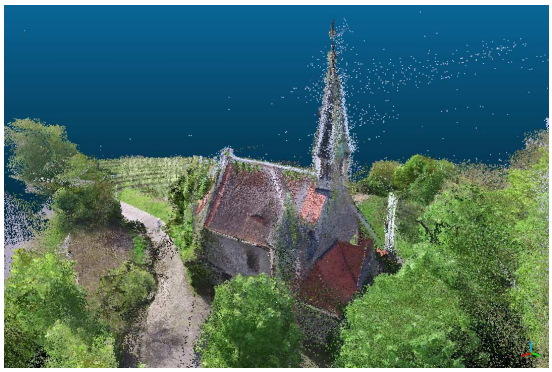


(f) MVE

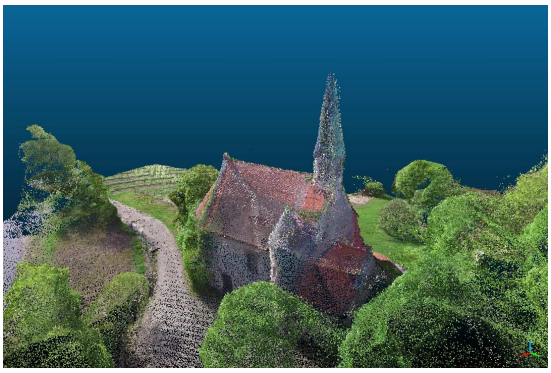
Figure A.4: First view of the produced point clouds by all pipelines of the chapel at Randersacker model with the extracted colours. Noise due to clouds in the horizon has been generated by MVE and openMVS.



(g) COLMAP Photometric



(h) COLMAP Geometric



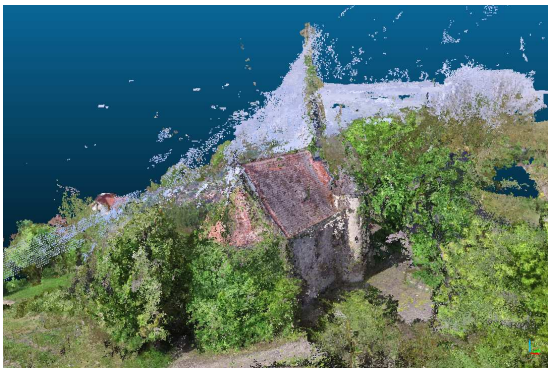
(i) PhotoScan

Figure A.4: First view of the produced point clouds by all pipelines of the chapel at Randersacker model with the extracted colours. Noise due to clouds in the horizon has been generated by MVE and openMVS.

(a) Visual SFM

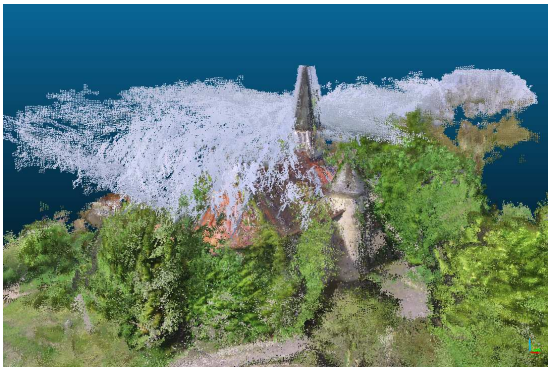
(b) Visual SFM + openMVS

(c) Visual SFM + MVE



(d) openMVG + openMVS

(e) openMVG + MVE



(f) MVE

Figure A.5: Second view of the produced point clouds by all pipelines of the chapel at Randersacker model with the extracted colours. Noise due to clouds is again visible from this view. Colour bleeding in the COLMAP generated models is marked in red.

(g) COLMAP Photometric

(h) COLMAP Geometric

(i) PhotoScan

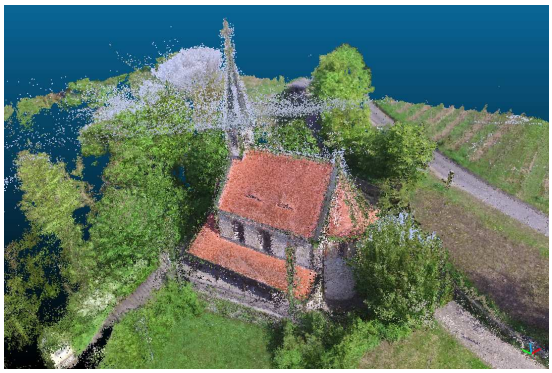
Figure A.5: Second view of the produced point clouds by all pipelines of the chapel at Randersacker model with the extracted colours. Noise due to clouds is again visible from this view. Colour bleeding in the COLMAP generated models is marked in red.

(a) Visual SFM

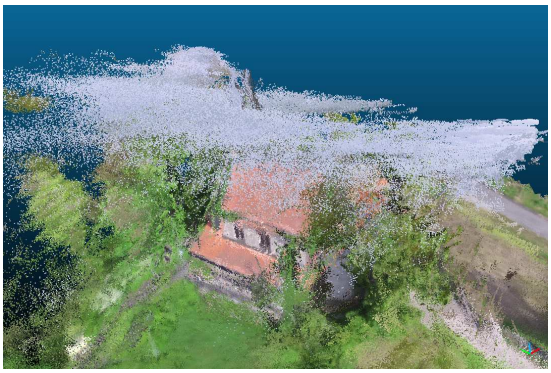
(b) Visual SFM + openMVS

(c) Visual SFM + MVE

(d) openMVG + openMVS



(e) openMVG + MVE



(f) MVE

Figure A.6: Third view of the produced point clouds by all pipelines of the chapel at Randersacker model with the extracted colours.

(g) COLMAP Photometric

(h) COLMAP Geometric

(i) PhotoScan

Figure A.6: Third view of the produced point clouds by all pipelines of the chapel at Randersacker model with the extracted colours.

A.3 Practice hall at the firefighter school

(a) Visual SFM

(b) Visual SFM + openMVS

(c) Visual SFM + MVE

(d) openMVG + openMVS

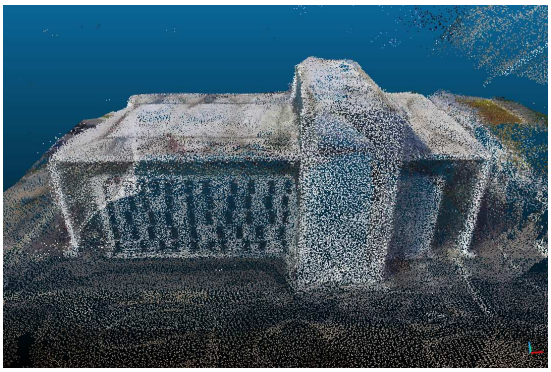
(e) openMVG + MVE

(f) MVE

Figure A.7: First view of the produced point clouds by all pipelines of the practice hall at the firefighter school model with the extracted colours. The generated models contain a large amount of noise. The Visual SFM + openMVS model generated almost indistinguishably the secondary structure of the building, marked in red.

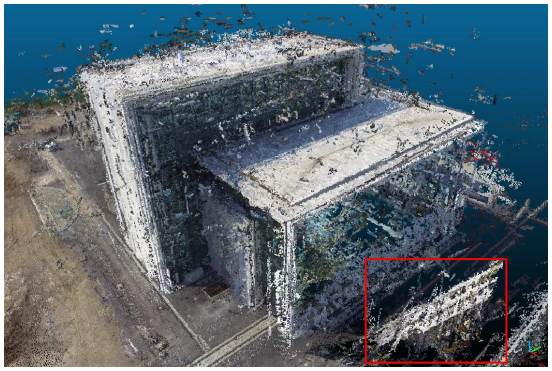
(g) COLMAP Photometric

(h) COLMAP Geometric



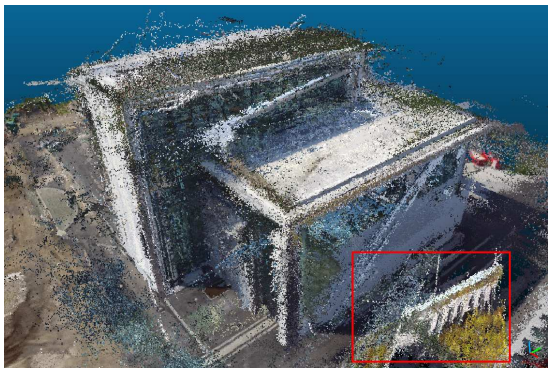
(i) PhotoScan

Figure A.7: First view of the produced point clouds by all pipelines of the practice hall at the firefighter school model with the extracted colours. The generated models contain a large amount of noise. The Visual SFM + openMVS model generated almost indistinguishably the secondary structure of the building, marked in red.



(a) Visual SFM

(b) Visual SFM + openMVS



(c) Visual SFM + MVE

(d) openMVG + openMVS

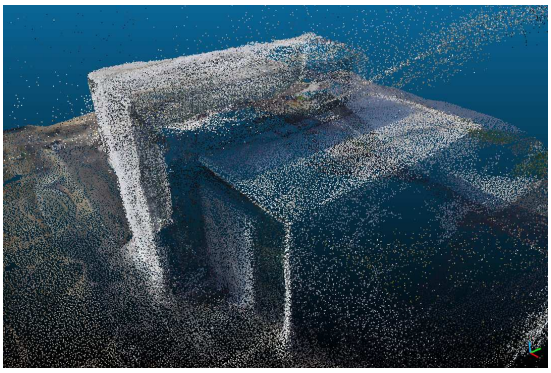
(e) openMVG + MVE

(f) MVE

Figure A.8: Second view of the produced point clouds by all pipelines of the practice hall at the firefighter school model with the extracted colours. Some secondary structures belonging to a nearby building where generated by the Visual SFM pipelines. This artefact has been marked in red.

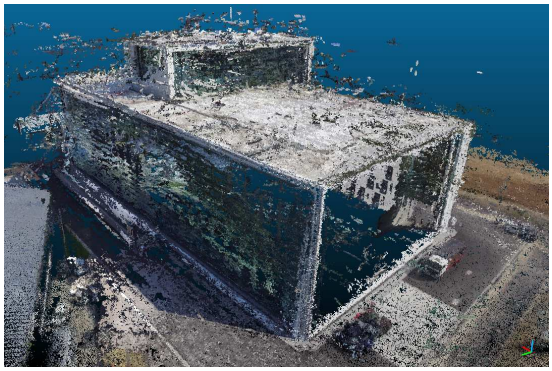
(g) COLMAP Photometric

(h) COLMAP Geometric



(i) PhotoScan

Figure A.8: Second view of the produced point clouds by all pipelines of the practice hall at the firefighter school model with the extracted colours. Some secondary structures belonging to a nearby building where generated by the Visual SFM pipelines. This artefact has been marked in red.



(a) Visual SFM

(b) Visual SFM + openMVS

(c) Visual SFM + MVE

(d) openMVG + openMVS

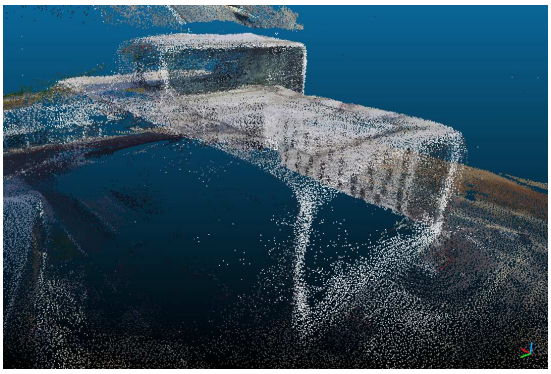
(e) openMVG + MVE

(f) MVE

Figure A.9: Third view of the produced point clouds by all pipelines of the practice hall at the firefighter school model with the extracted colours. No algorithm was able to generate the side walls of the building.

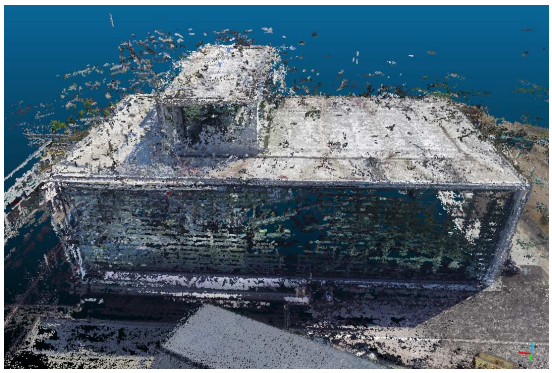
(g) COLMAP Photometric

(h) COLMAP Geometric



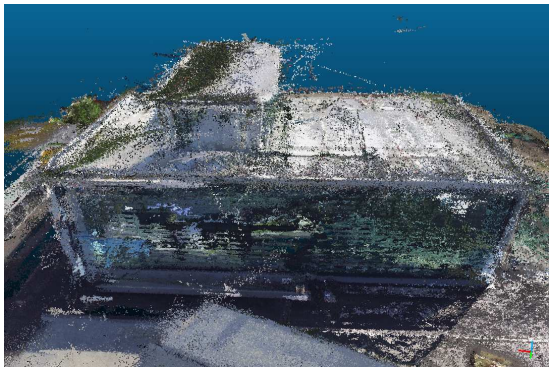
(i) PhotoScan

Figure A.9: Third view of the produced point clouds by all pipelines of the practice hall at the firefighter school model with the extracted colours. No algorithm was able to generate the side walls of the building.



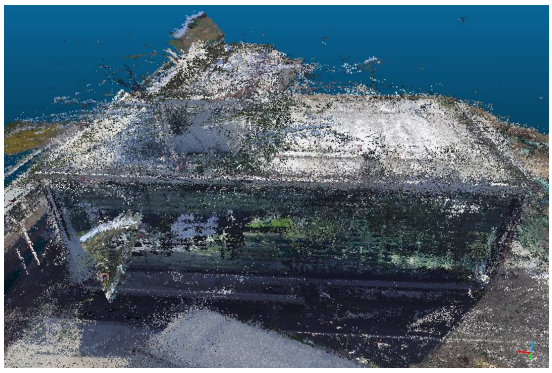
(a) Visual SFM

(b) Visual SFM + openMVS



(c) Visual SFM + MVE

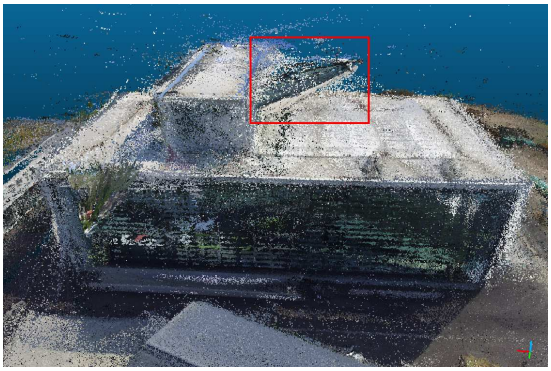
(d) openMVG + openMVS



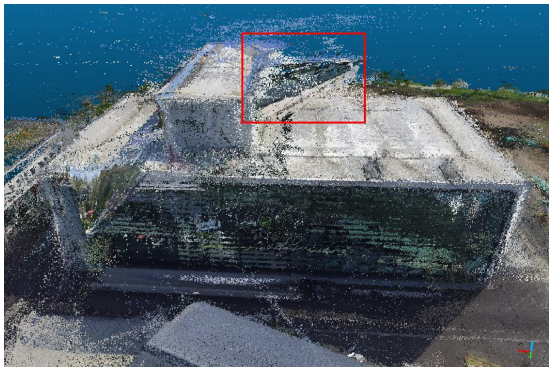
(e) openMVG + MVE

(f) MVE

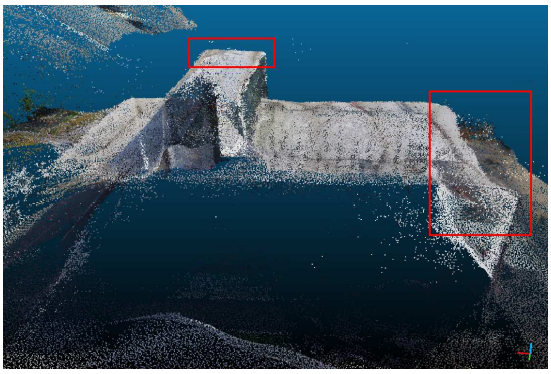
Figure A.10: Fourth view of the produced point clouds by all pipelines of the practice hall at the firefighter school model with the extracted colours. The back part of the building, composed entirely of windows, was not correctly generated by the algorithms. PhotoScan failed to produce any structure in this region. The perspective artefacts produced by the COLMAP pipelines are marked in red, as well as the corners that have been softened by PhotoScan.



(g) COLMAP Photometric



(h) COLMAP Geometric



(i) PhotoScan

Figure A.10: Fourth view of the produced point clouds by all pipelines of the practice hall at the firefighter school model with the extracted colours. The back part of the building, composed entirely of windows, was not correctly generated by the algorithms. PhotoScan failed to produce any structure in this region. The perspective artefacts produced by the COLMAP pipelines are marked in red, as well as the corners that have been softened by PhotoScan.

A.4 Burning house at the firefighter school

(a) Visual SFM

(b) Visual SFM + openMVS

(c) Visual SFM + MVE

(d) openMVG + openMVS

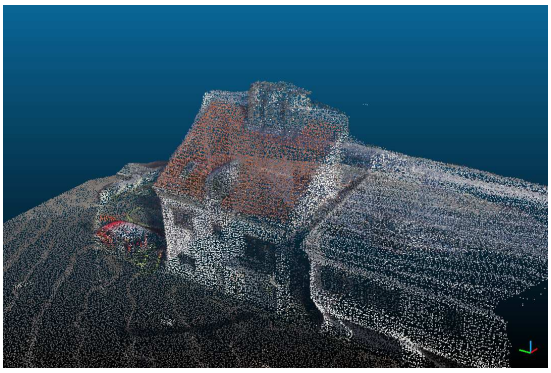
(e) openMVG + MVE

(f) MVE

Figure A.11: First view of the produced point clouds by all pipelines of the burning house at the firefighter school model with the extracted colours. Some smoke noise is visible in the models, behind the burning house.

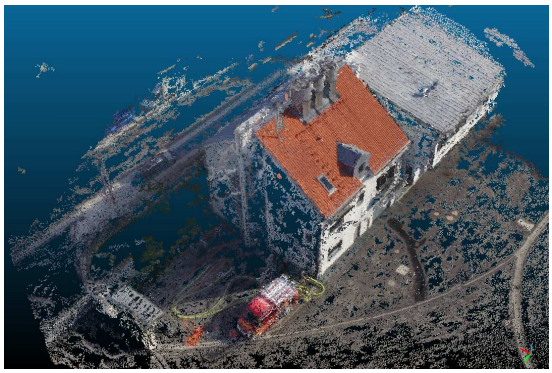
(g) COLMAP Photometric

(h) COLMAP Geometric



(i) PhotoScan

Figure A.11: First view of the produced point clouds by all pipelines of the burning house at the firefighter school model with the extracted colours. Some smoke noise is visible in the models, behind the burning house.

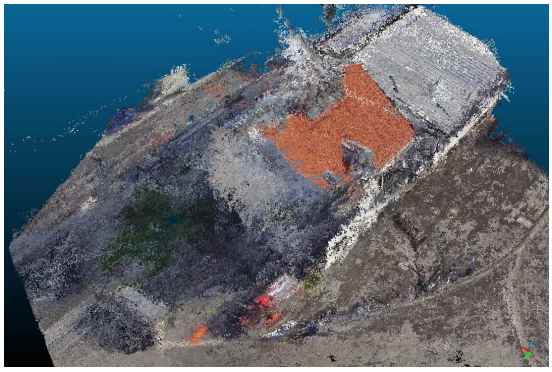


(a) Visual SFM

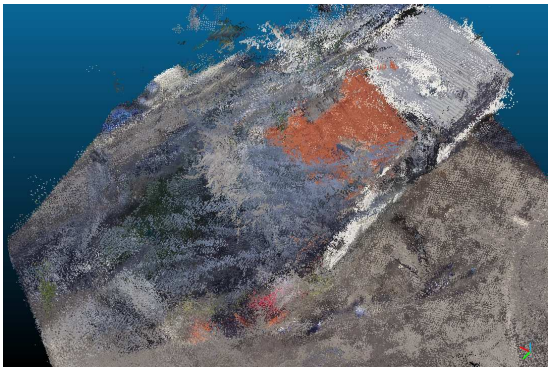
(b) Visual SFM + openMVS

(c) Visual SFM + MVE

(d) openMVG + openMVS



(e) openMVG + MVE

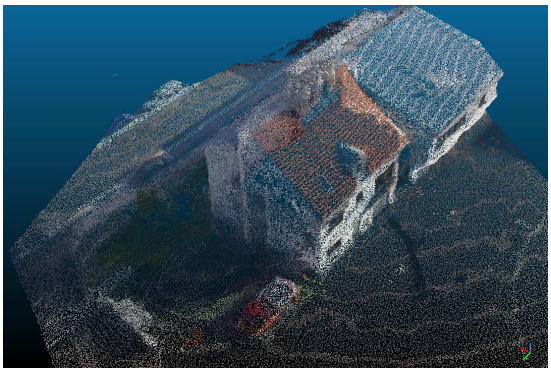


(f) MVE

Figure A.12: Second view of the produced point clouds by all pipelines of the burning house at the firefighter school model with the extracted colours. From this perspective is clearer to see the produced smoke noise. Some algorithms seem invulnerable to this effect though, like Visual SFM, openMVG + openMVS, PhotoScan and COLMAP.

(g) COLMAP Photometric

(h) COLMAP Geometric



(i) PhotoScan

Figure A.12: Second view of the produced point clouds by all pipelines of the burning house at the firefighter school model with the extracted colours. From this perspective is clearer to see the produced smoke noise. Some algorithms seem invulnerable to this effect though, like Visual SFM, openMVG + openMVS, PhotoScan and COLMAP.

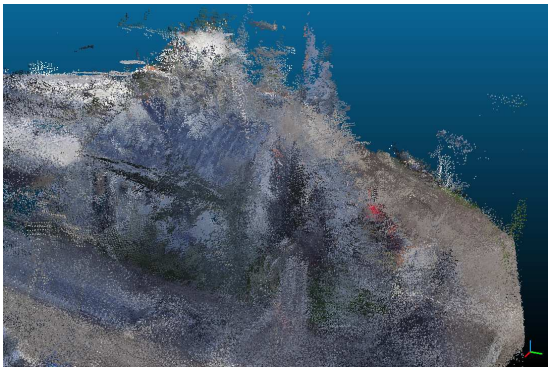
(a) Visual SFM

(b) Visual SFM + openMVS

(c) Visual SFM + MVE

(d) openMVG + openMVS

(e) openMVG + MVE

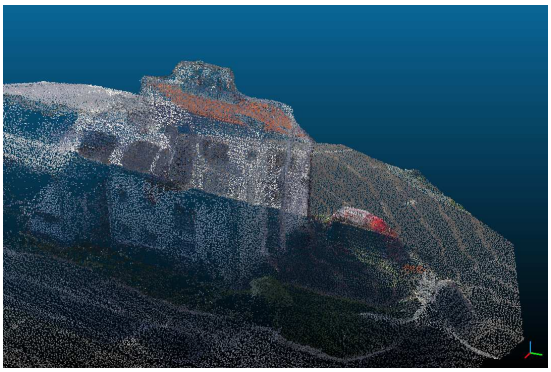


(f) MVE

Figure A.13: Third view of the produced point clouds by all pipelines of the burning house at the firefighter school model with the extracted colours. Noise due to smoke is predominant in this view. It can also be seen that Visual SFM failed to generate the back wall of the building, but COLMAP managed to do so without major visible noise.

(g) COLMAP Photometric

(h) COLMAP Geometric



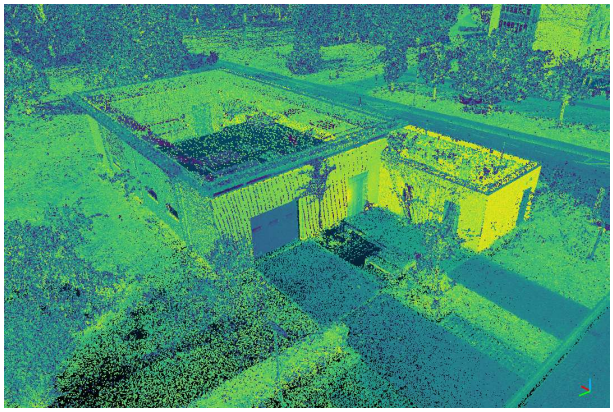
(i) PhotoScan

Figure A.13: Third view of the produced point clouds by all pipelines of the burning house at the firefighter school model with the extracted colours. Noise due to smoke is predominant in this view. It can also be seen that Visual SFM failed to generate the back wall of the building, but COLMAP managed to do so without major visible noise.

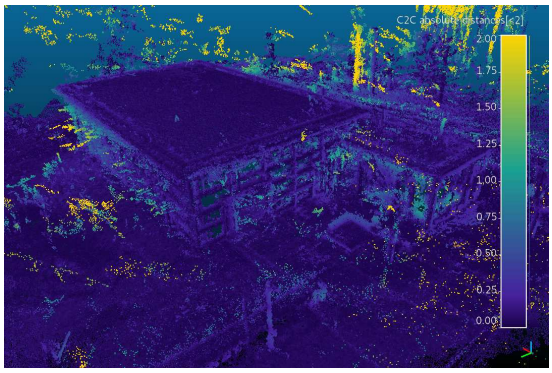
Appendix B

Views of the cloud to cloud distance error

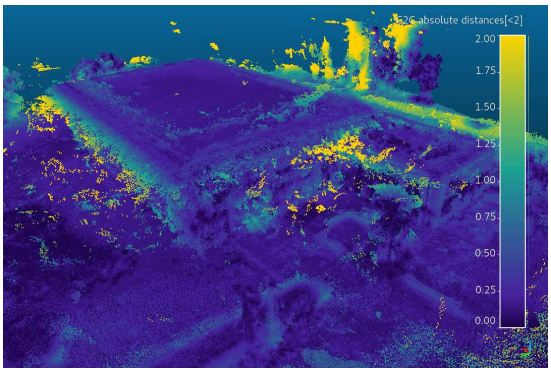
B.1 Robotics Hall



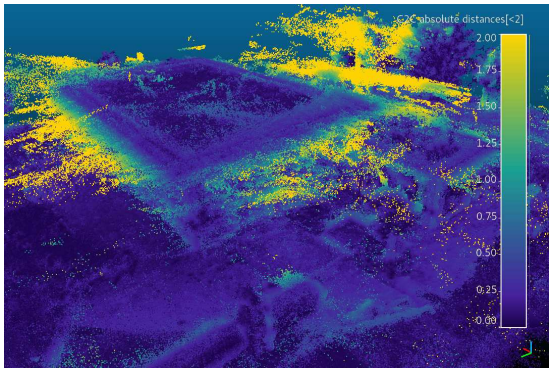
(a) Reference frame from Laser Scanner



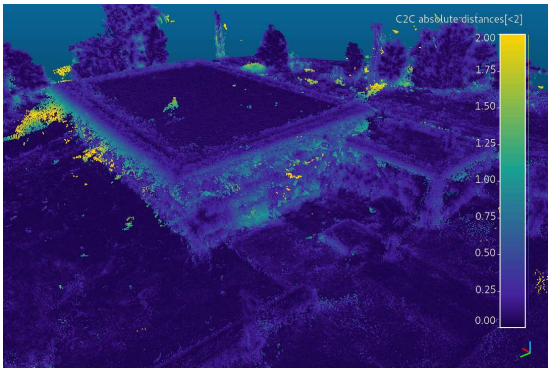
(b) Visual SFM



(c) Visual SFM + openMVS

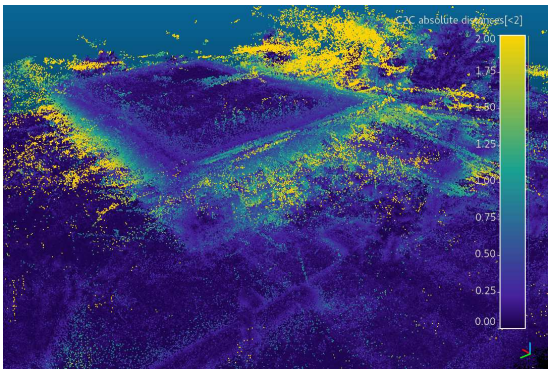


(d) Visual SFM + MVE

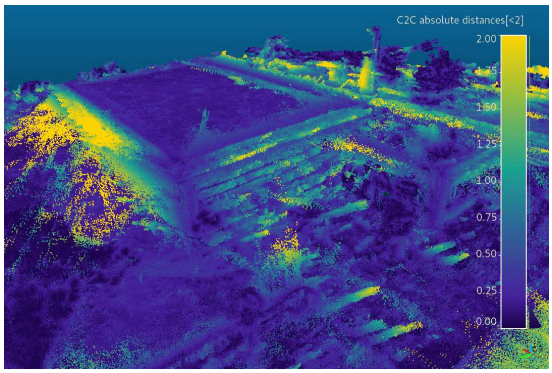


(e) openMVG + openMVS

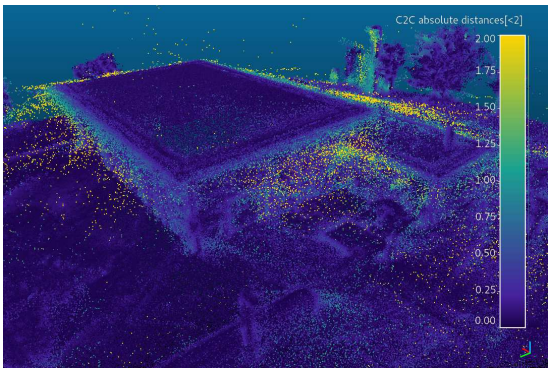
Figure B.1: First view of the produced point clouds by all pipelines of the robotics hall model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. It is visible that most of the error is distributed on the roof plane, specially for Visual SFM + MVE.



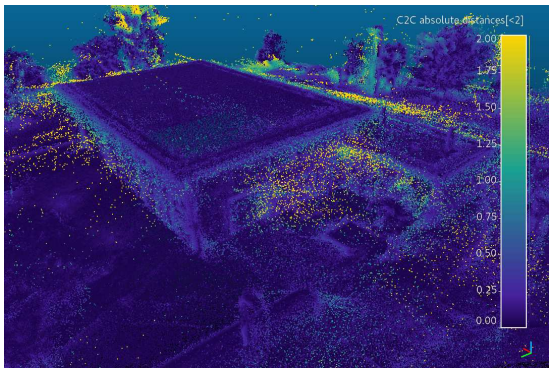
(f) openMVG + MVE



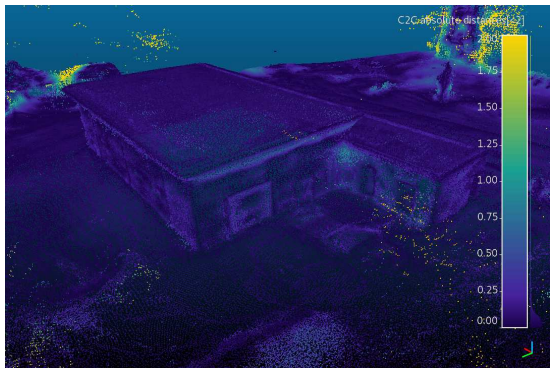
(g) MVE



(h) COLMAP Photometric



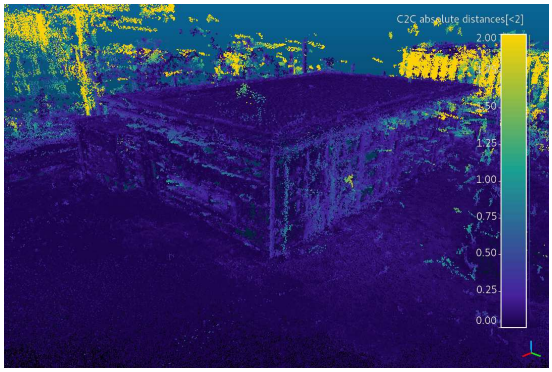
(i) COLMAP Geometric



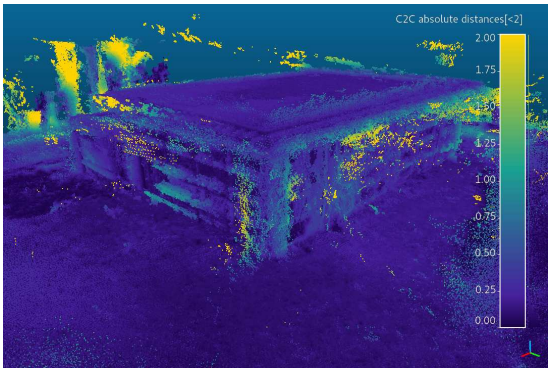
(j) PhotoScan

Figure B.1: First view of the produced point clouds by all pipelines of the robotics hall model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. It is visible that most of the error is distributed on the roof plane, specially for Visual SFM + MVE.

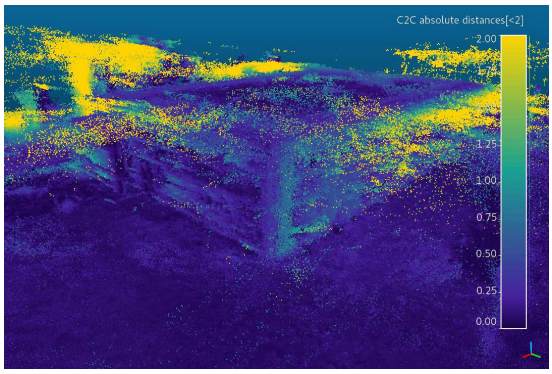
(a) Reference frame from Laser Scanner



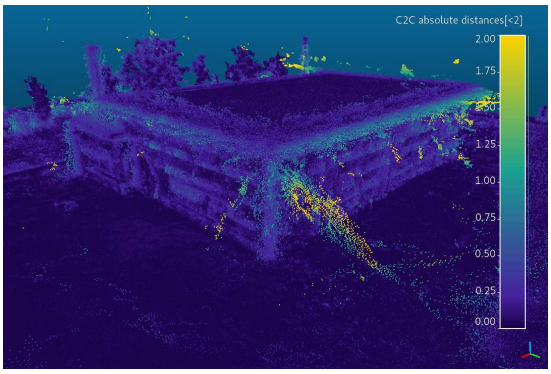
(b) Visual SFM



(c) Visual SFM + openMVS

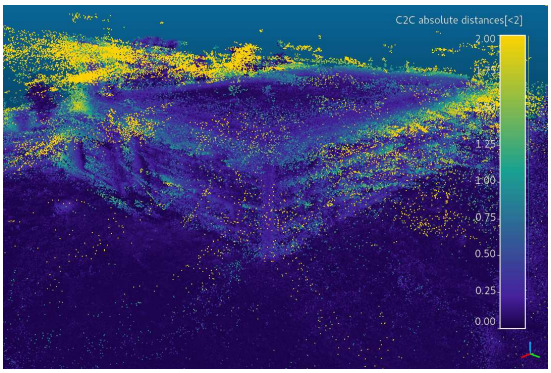


(d) Visual SFM + MVE

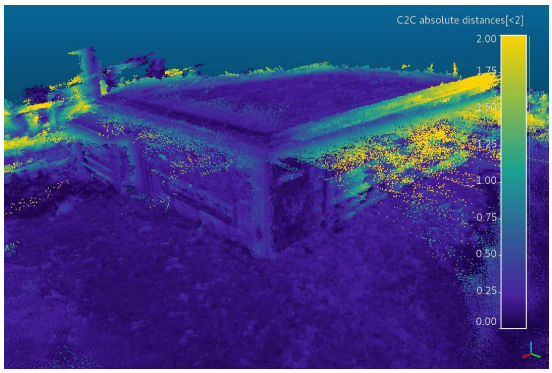


(e) openMVG + openMVS

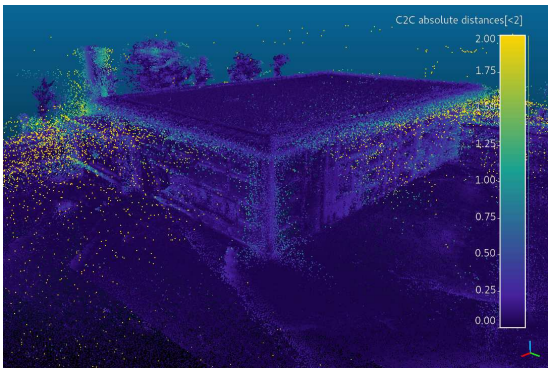
Figure B.2: Second view of the produced point clouds by all pipelines of the robotics hall model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model.



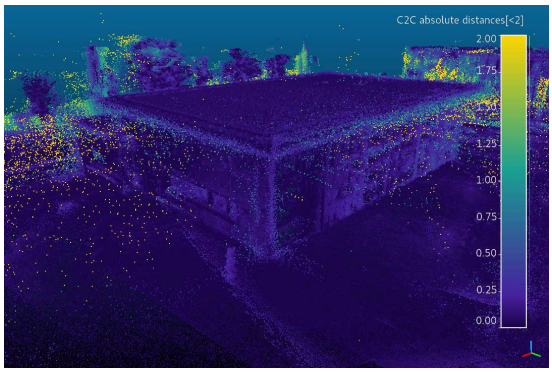
(f) openMVG + MVE



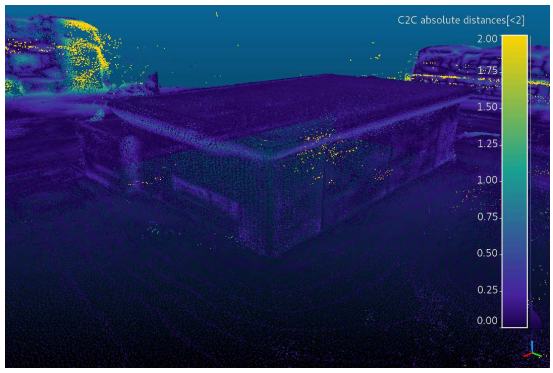
(g) MVE



(h) COLMAP Photometric



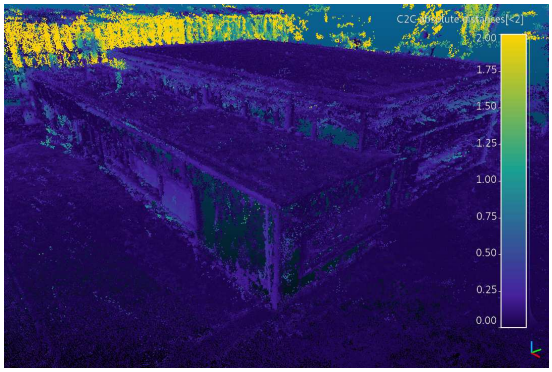
(i) COLMAP Geometric



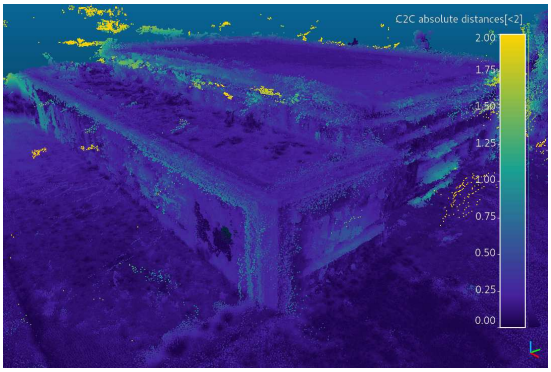
(j) PhotoScan

Figure B.2: Second view of the produced point clouds by all pipelines of the robotics hall model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model.

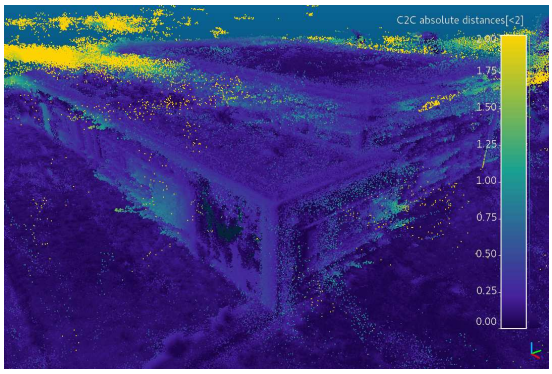
(a) Reference frame from Laser Scanner



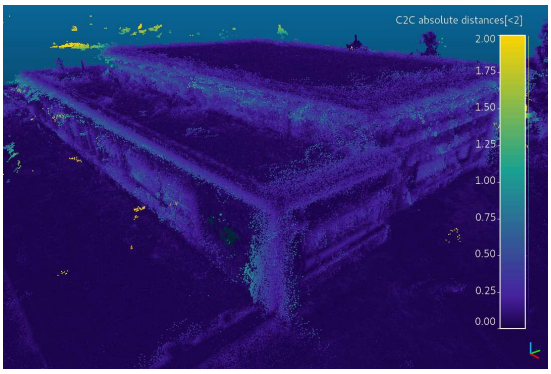
(b) Visual SFM



(c) Visual SFM + openMVS

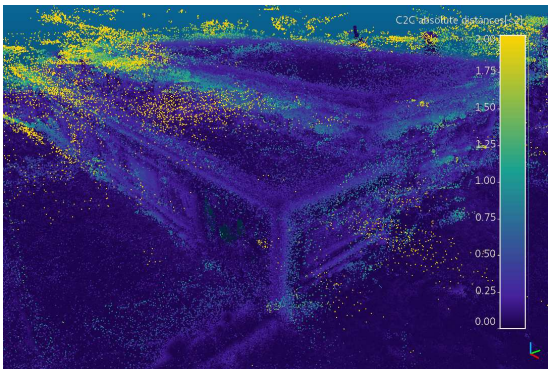


(d) Visual SFM + MVE

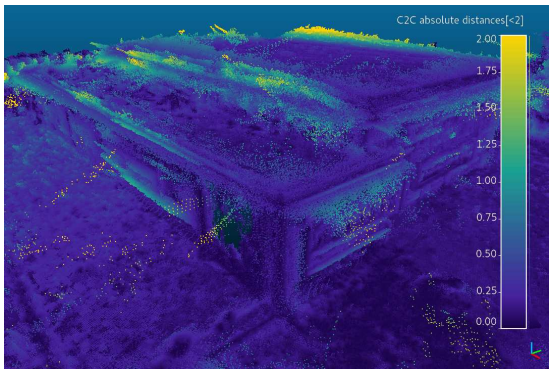


(e) openMVG + openMVS

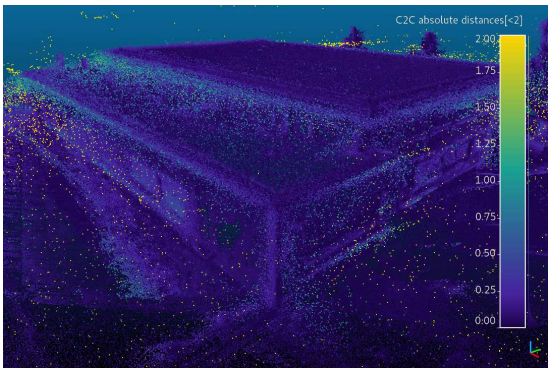
Figure B.3: Third view of the produced point clouds by all pipelines of the robotics hall model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model.



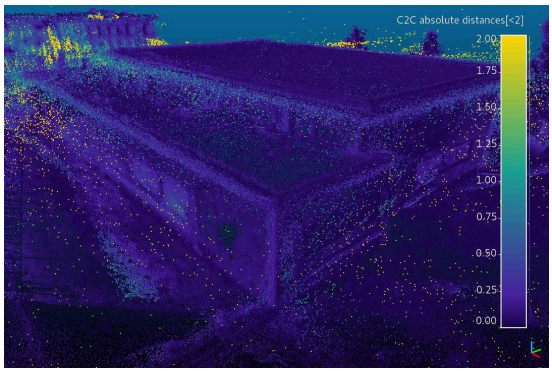
(f) openMVG + MVE



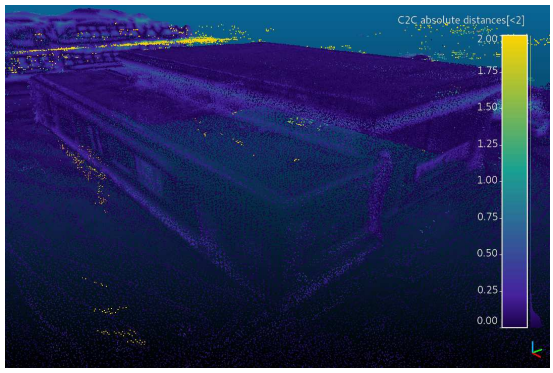
(g) MVE



(h) COLMAP Photometric



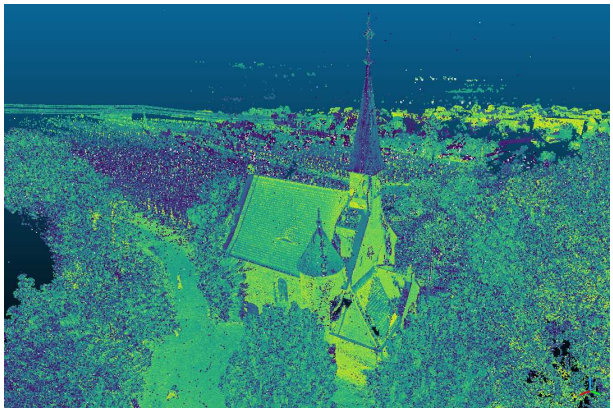
(i) COLMAP Geometric



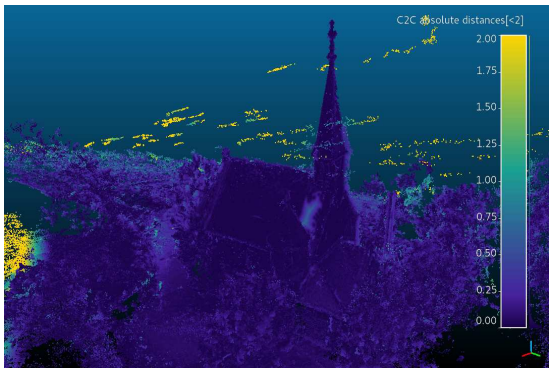
(j) PhotoScan

Figure B.3: Third view of the produced point clouds by all pipelines of the robotics hall model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model.

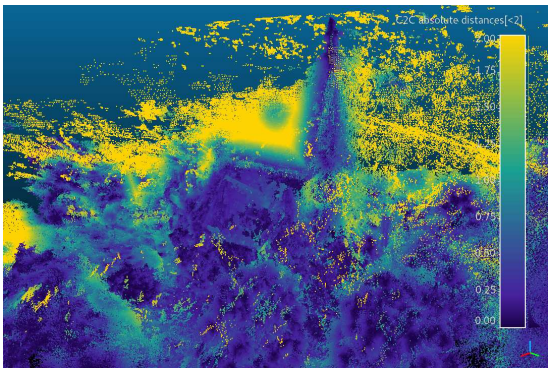
B.2 Chapel at Randersacker



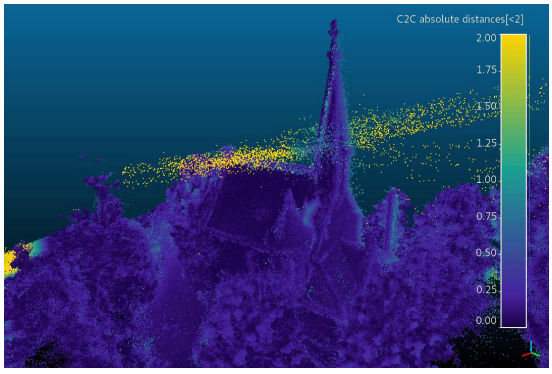
(a) Reference frame from Laser Scanner



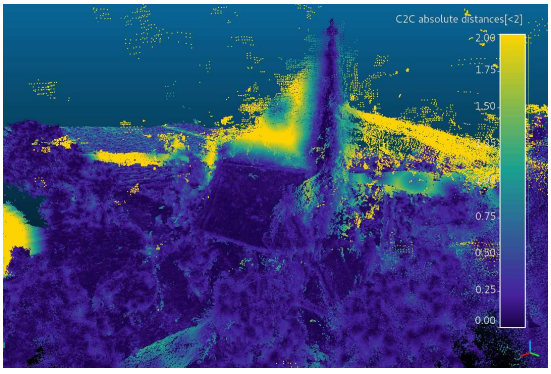
(b) Visual SFM



(c) Visual SFM + openMVS

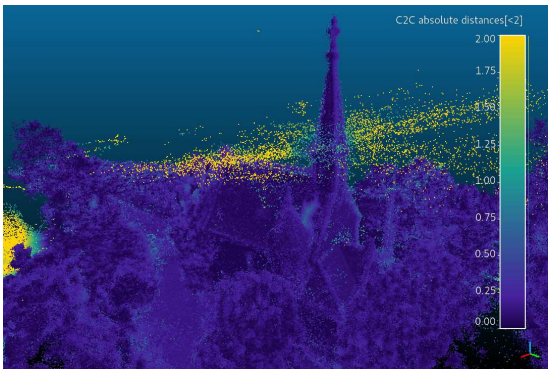


(d) Visual SFM + MVE

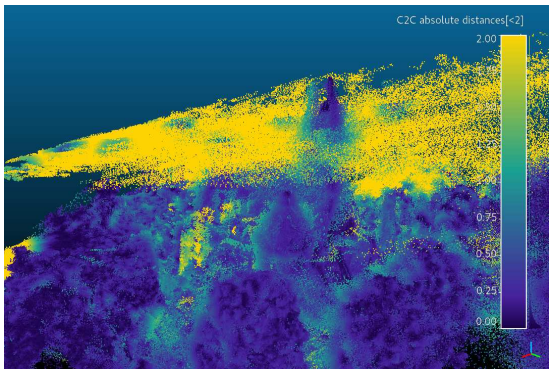


(e) openMVG + openMVS

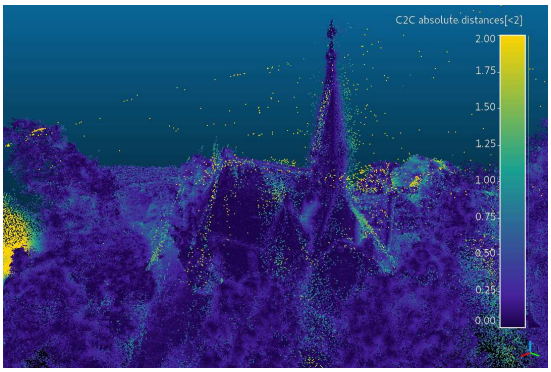
Figure B.4: First view of the produced point clouds by all pipelines of the chapel at Randersacker model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. The error is mostly spread on the generated noise due to the clouds.



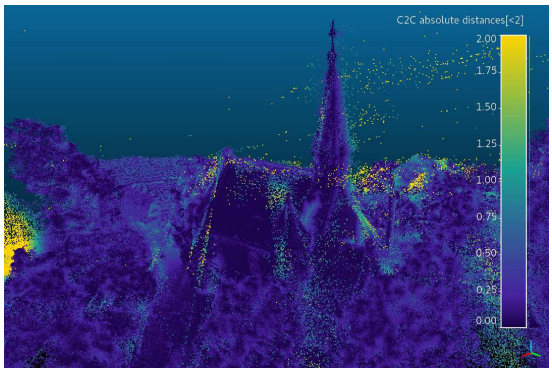
(f) openMVG + MVE



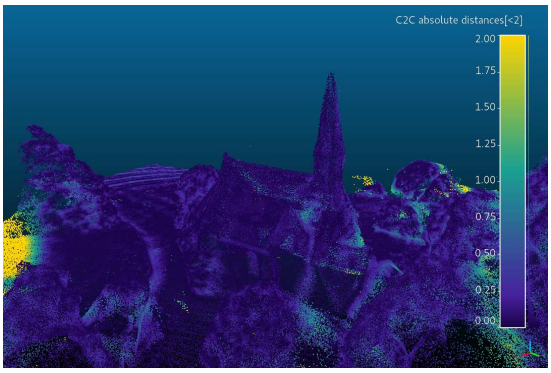
(g) MVE



(h) COLMAP Photometric

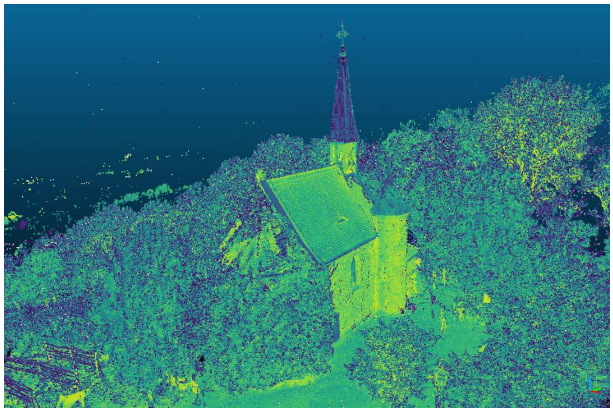


(i) COLMAP Geometric

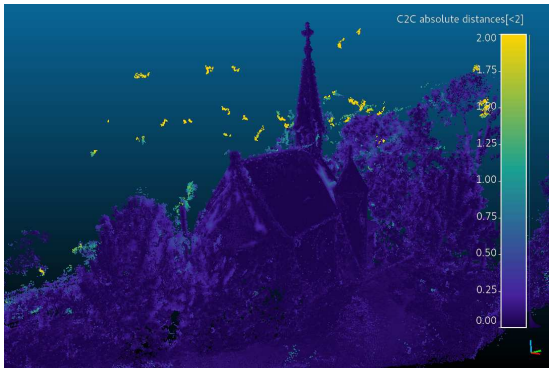


(j) PhotoScan

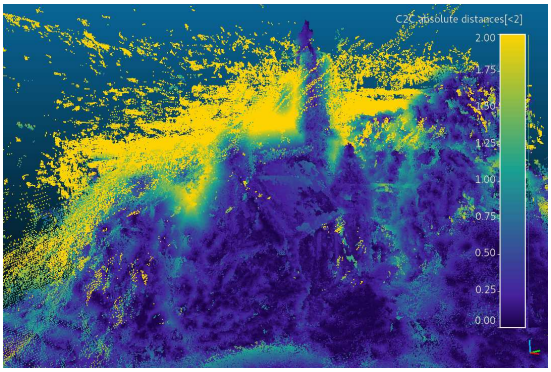
Figure B.4: First view of the produced point clouds by all pipelines of the chapel at Randersacker model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. The error is mostly spread on the generated noise due to the clouds.



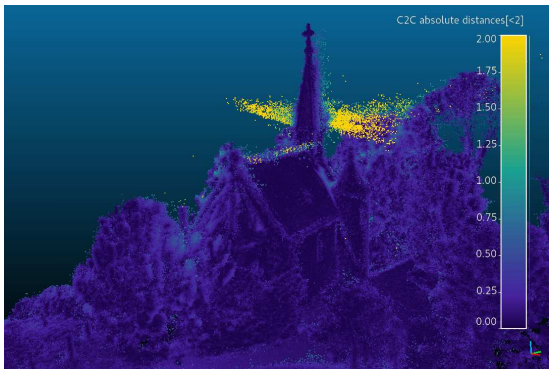
(a) Reference frame from Laser Scanner



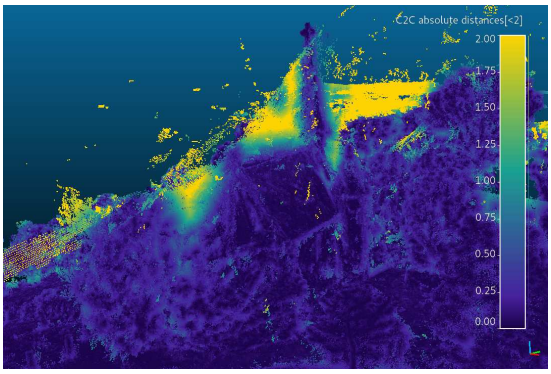
(b) Visual SFM



(c) Visual SFM + openMVS

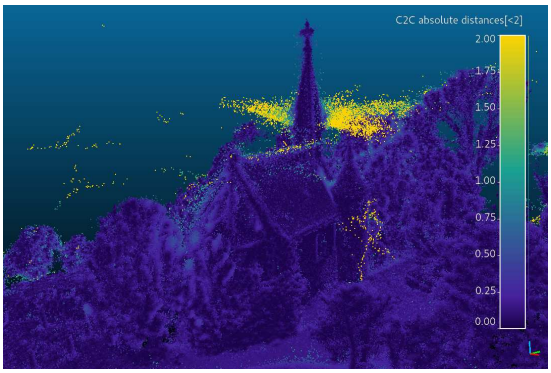


(d) Visual SFM + MVE

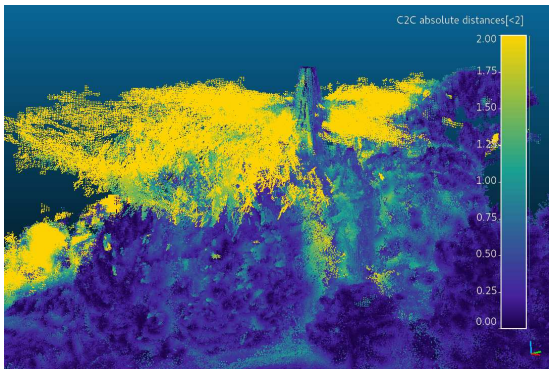


(e) openMVG + openMVS

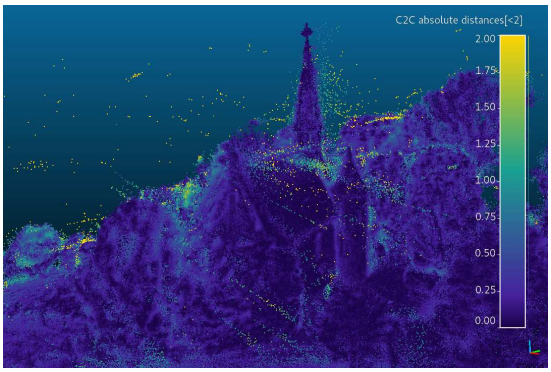
Figure B.5: Second view of the produced point clouds by all pipelines of the chapel at Randersacker model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. The large noise cloud generated by MVE is best visible from this view.



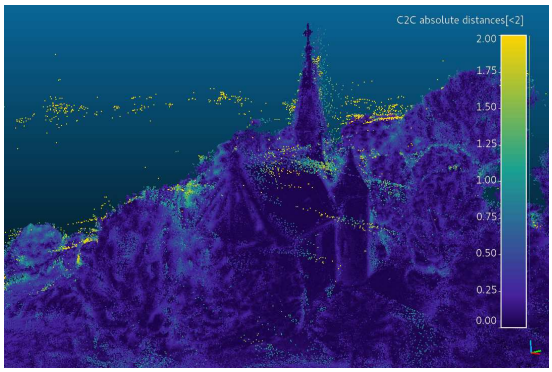
(f) openMVG + MVE



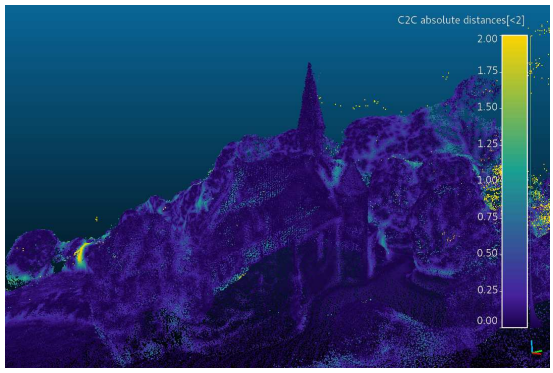
(g) MVE



(h) COLMAP Photometric

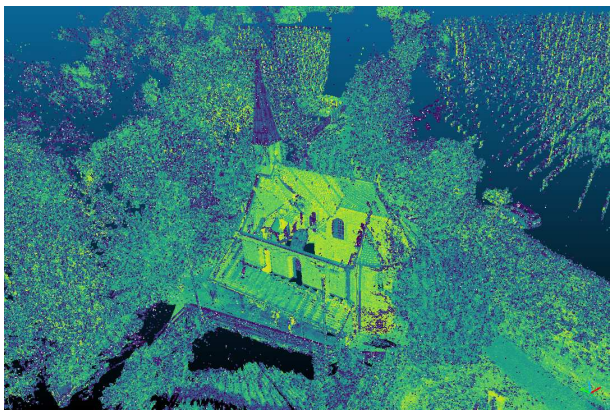


(i) COLMAP Geometric

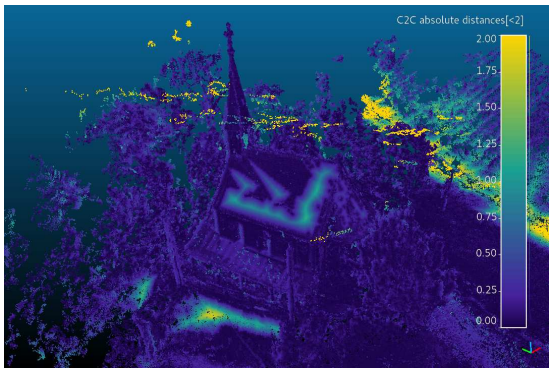


(j) PhotoScan

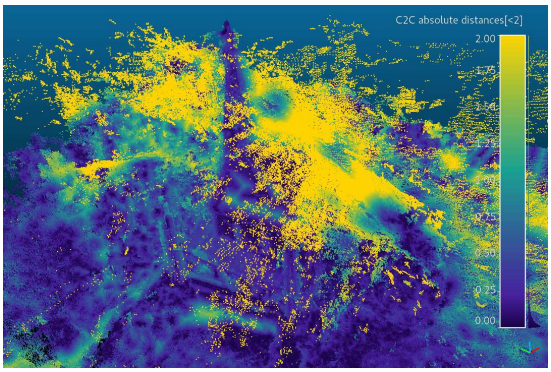
Figure B.5: Second view of the produced point clouds by all pipelines of the chapel at Randersacker model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. The large noise cloud generated by MVE is best visible from this view.



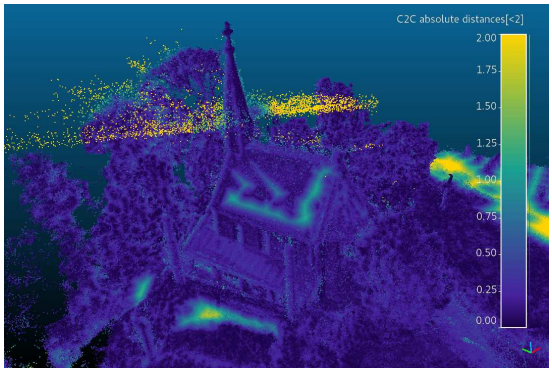
(a) Reference frame from Laser Scanner



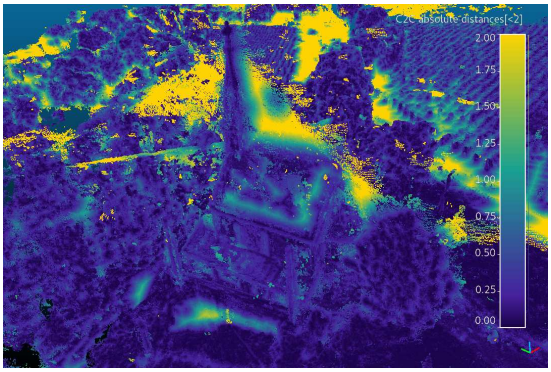
(b) Visual SFM



(c) Visual SFM + openMVS

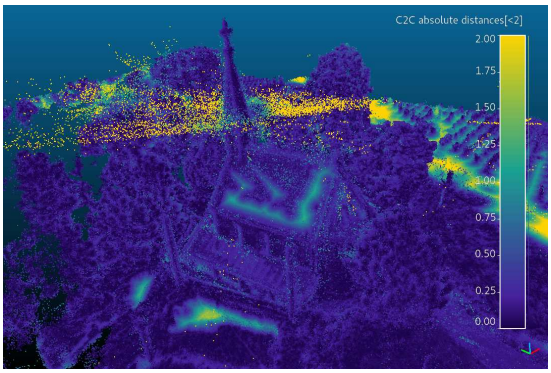


(d) Visual SFM + MVE

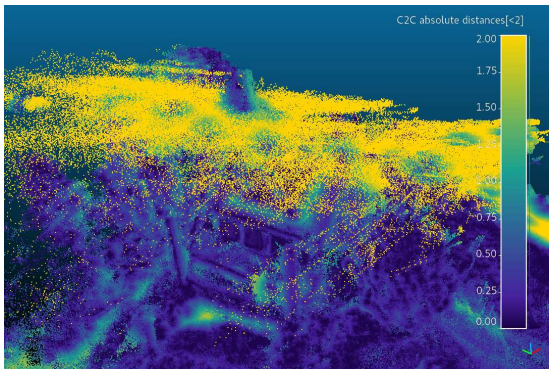


(e) openMVG + openMVS

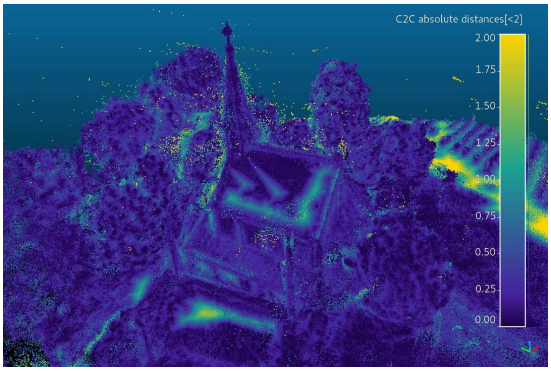
Figure B.6: Third view of the produced point clouds by all pipelines of the chapel at Randersacker model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. The inverted L shape of error due to the missing roof of the reference model is visible from this view.



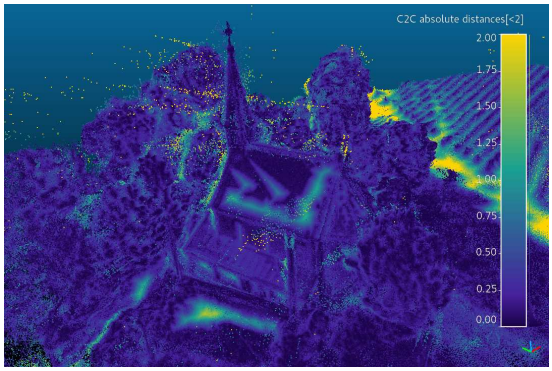
(f) openMVG + MVE



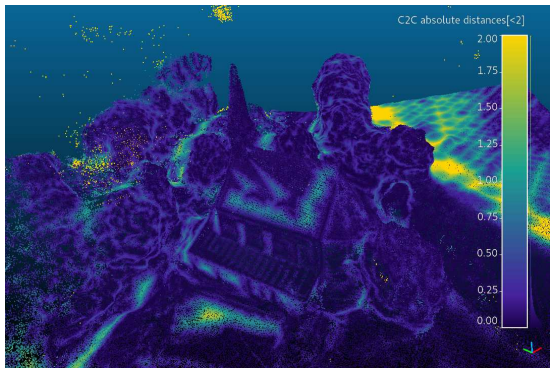
(g) MVE



(h) COLMAP Photometric



(i) COLMAP Geometric

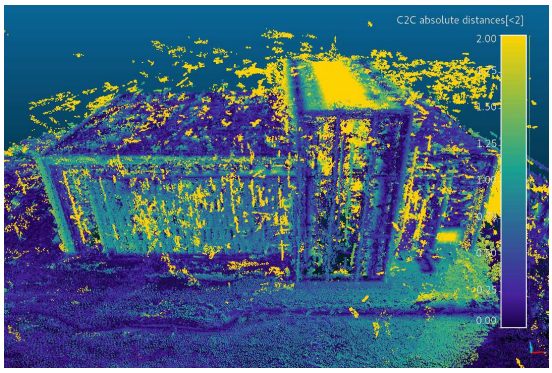


(j) PhotoScan

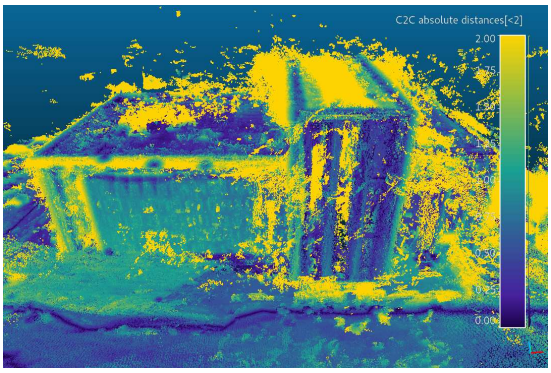
Figure B.6: Third view of the produced point clouds by all pipelines of the chapel at Randersacker model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. The inverted L shape of error due to the missing roof of the reference model is visible from this view.

B.3 Practice hall at the firefighter school

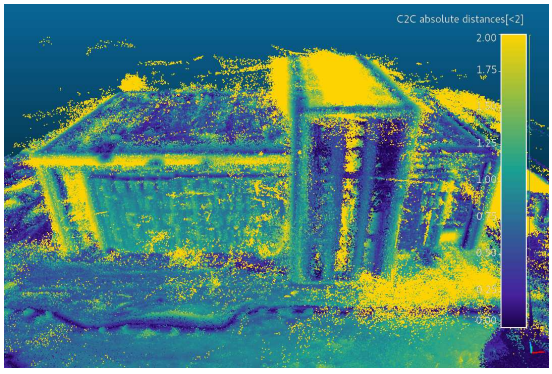
(a) Reference frame from Laser Scanner



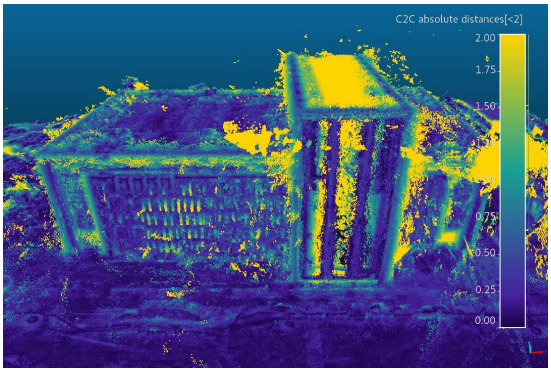
(b) Visual SFM



(c) Visual SFM + openMVS

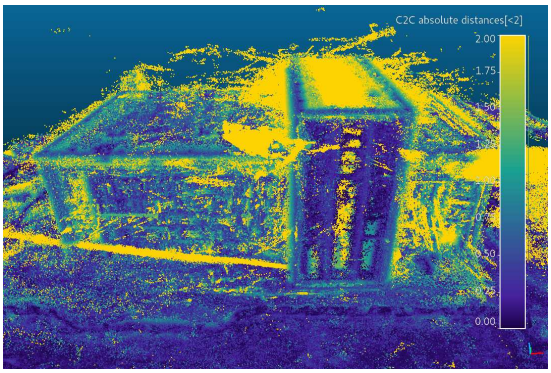


(d) Visual SFM + MVE

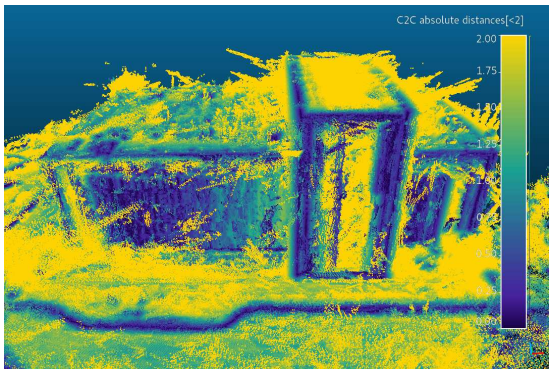


(e) openMVG + openMVS

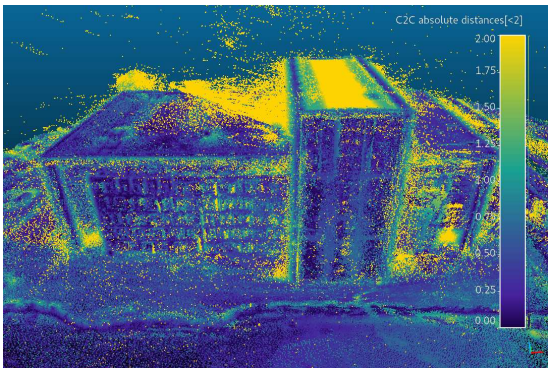
Figure B.7: First view of the produced point clouds by all pipelines of the practice hall at the firefighter school model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. The noise in the models is strongly visible in this view.



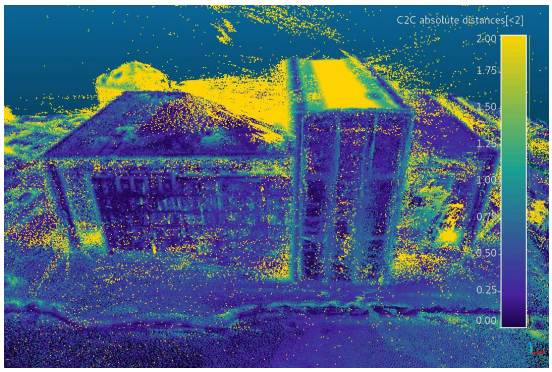
(f) openMVG + MVE



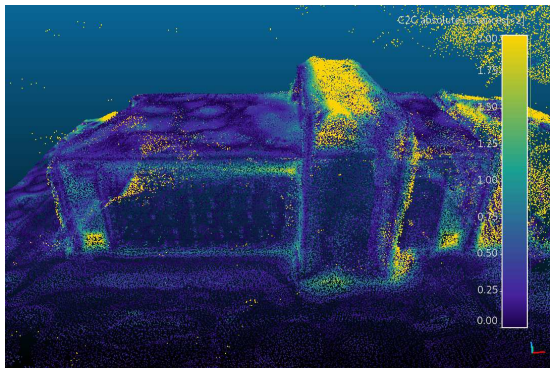
(g) MVE



(h) COLMAP Photometric

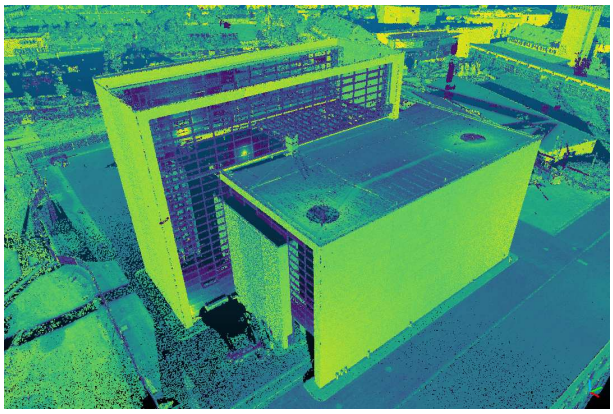


(i) COLMAP Geometric

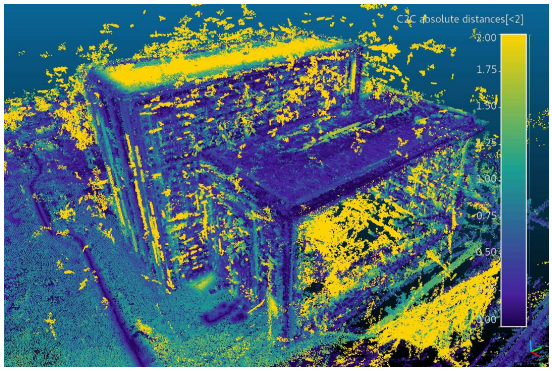


(j) PhotoScan

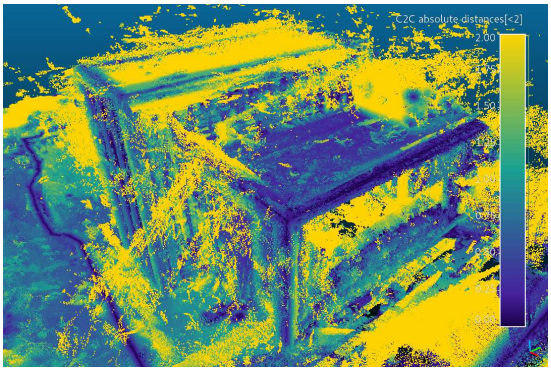
Figure B.7: First view of the produced point clouds by all pipelines of the practice hall at the firefighter school model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. The noise in the models is strongly visible in this view.



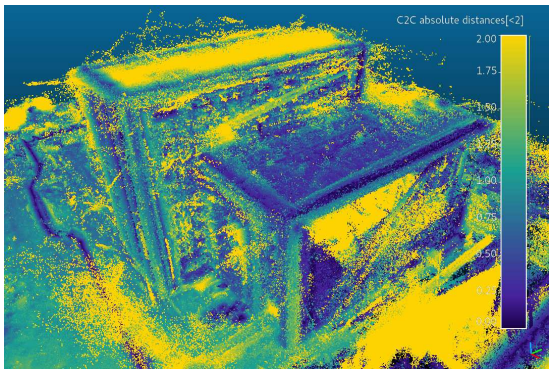
(a) Reference frame from Laser Scanner



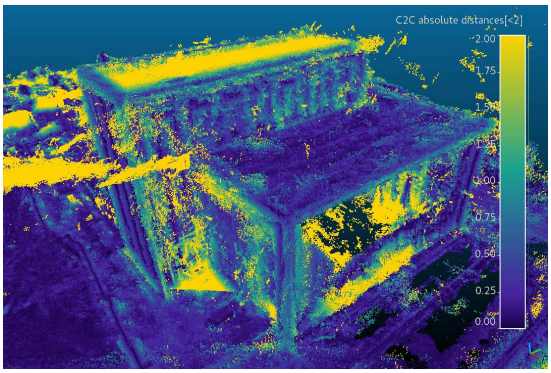
(b) Visual SFM



(c) Visual SFM + openMVS

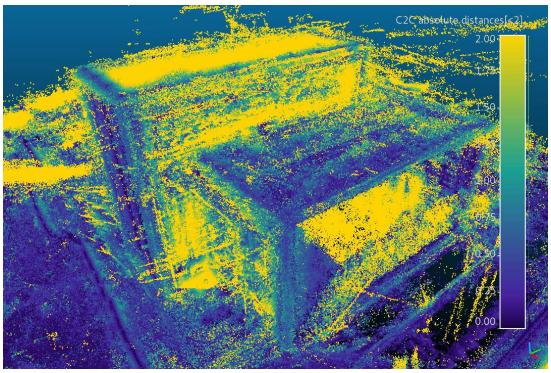


(d) Visual SFM + MVE

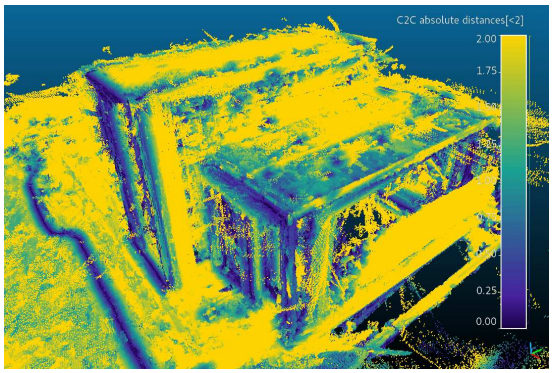


(e) openMVG + openMVS

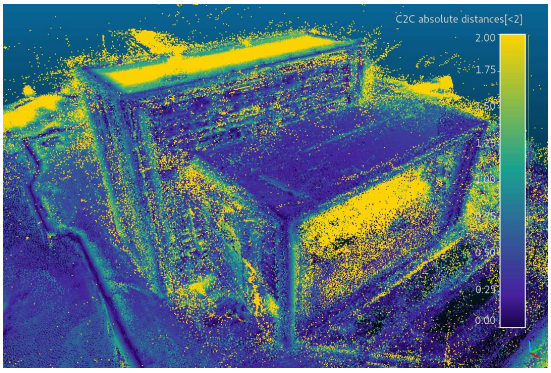
Figure B.8: Second view of the produced point clouds by all pipelines of the practice hall at the firefighter school model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. In this view it can be seen that there are large error structures near the reflective surfaces of the secondary structure of the building, as well as errors generated inside the building. The missing roof of the secondary structure from the reference frame also leads to classify these points as error in the models.



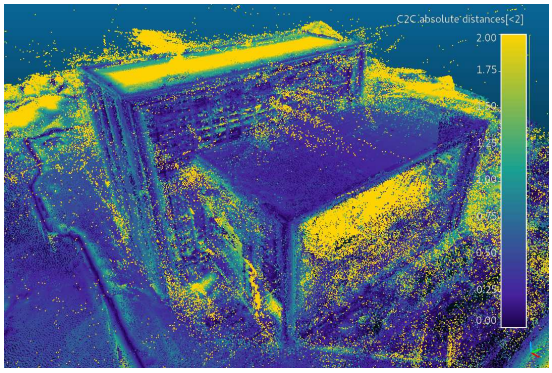
(f) openMVG + MVE



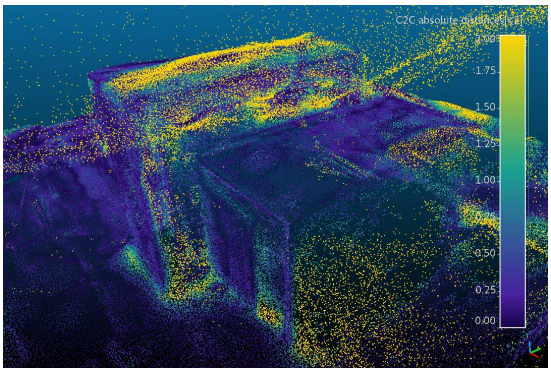
(g) MVE



(h) COLMAP Photometric

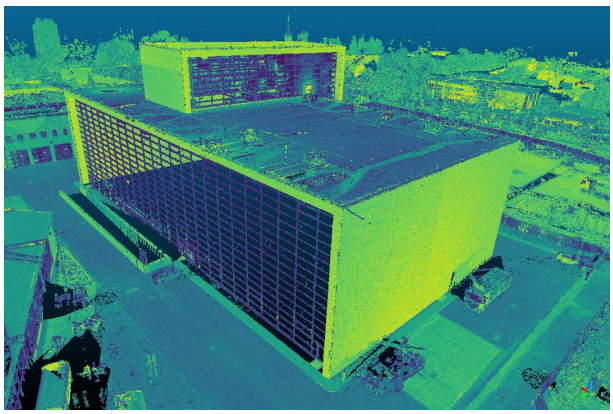


(i) COLMAP Geometric

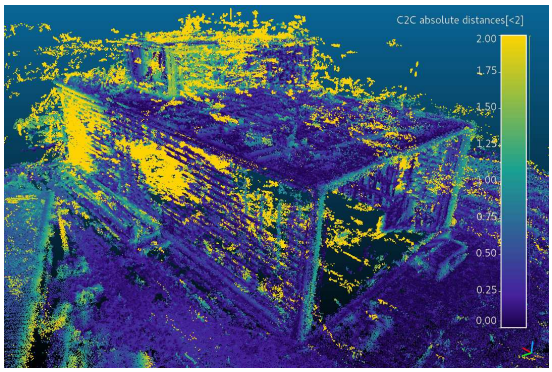


(j) PhotoScan

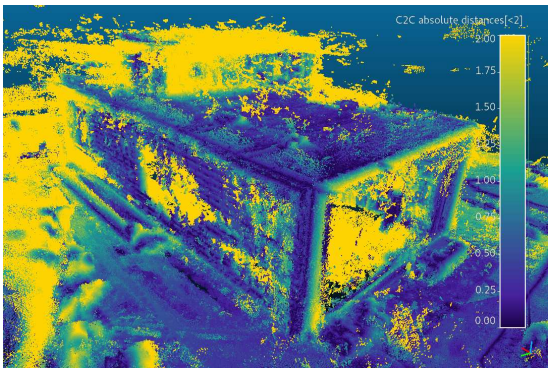
Figure B.8: Second view of the produced point clouds by all pipelines of the practice hall at the firefighter school model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. In this view it can be seen that there are large error structures near the reflective surfaces of the secondary structure of the building, as well as errors generated inside the building. The missing roof of the secondary structure from the reference frame also leads to classify these points as error in the models.



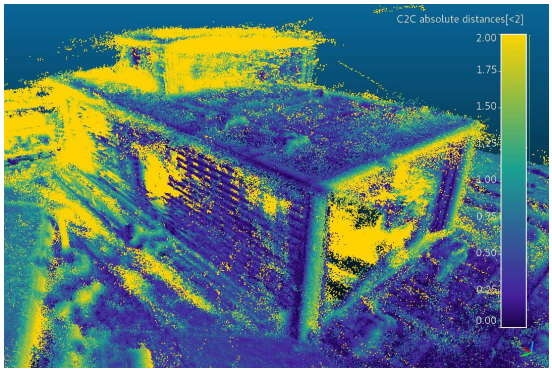
(a) Reference frame from Laser Scanner



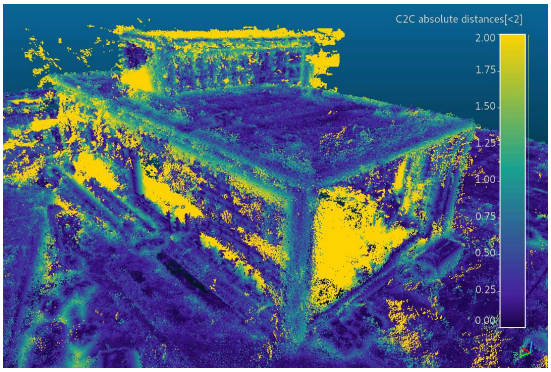
(b) Visual SFM



(c) Visual SFM + openMVS

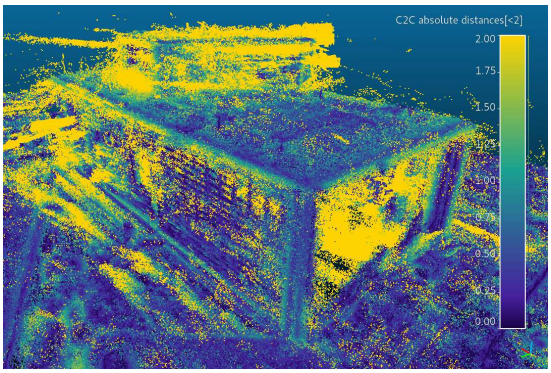


(d) Visual SFM + MVE

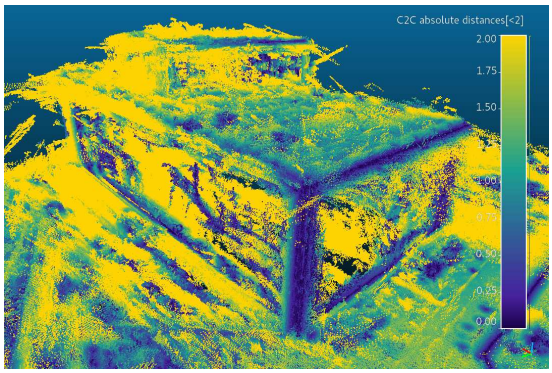


(e) openMVG + openMVS

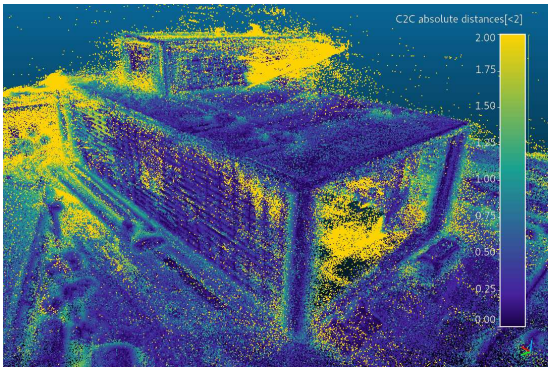
Figure B.9: Third view of the produced point clouds by all pipelines of the practice hall at the firefighter school model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. Errors inside the building are seen as well in this view. The large reflective surface on the back of the building is also incomplete. The side wall of the building is completely missing.



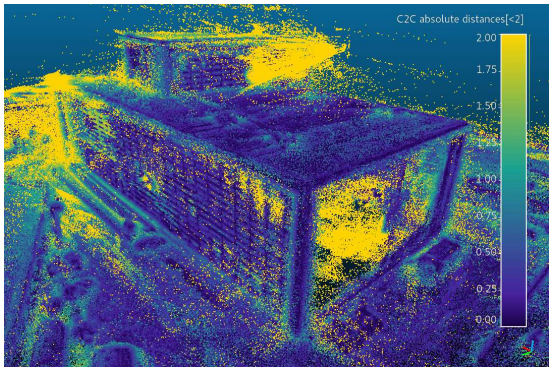
(f) openMVG + MVE



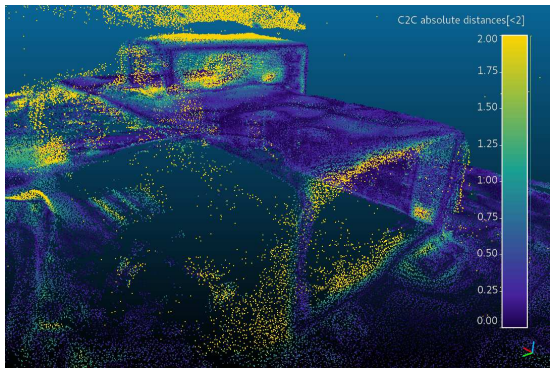
(g) MVE



(h) COLMAP Photometric

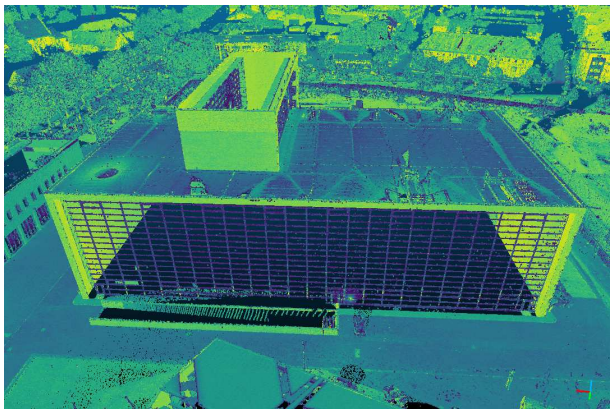


(i) COLMAP Geometric

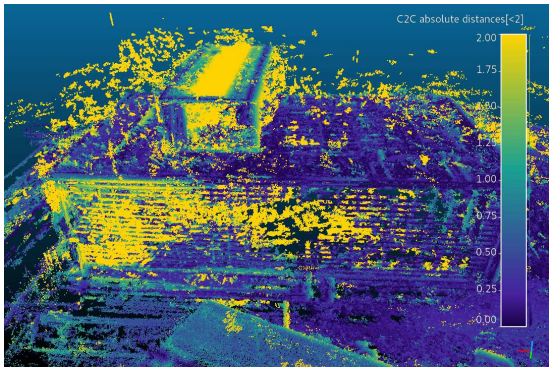


(j) PhotoScan

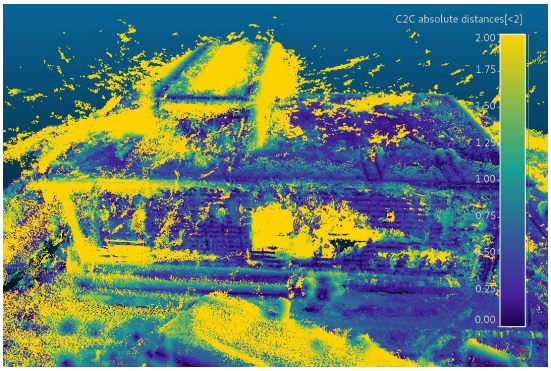
Figure B.9: Third view of the produced point clouds by all pipelines of the practice hall at the firefighter school model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. Errors inside the building are seen as well in this view. The large reflective surface on the back of the building is also incomplete. The side wall of the building is completely missing.



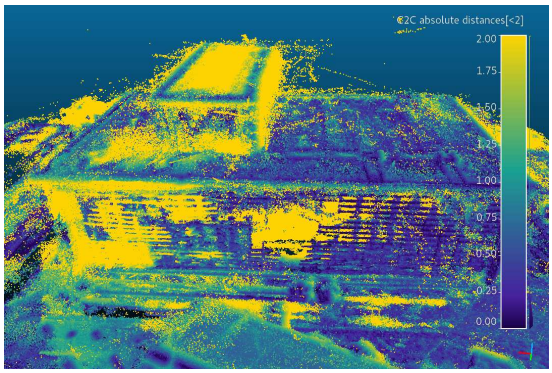
(a) Reference frame from Laser Scanner



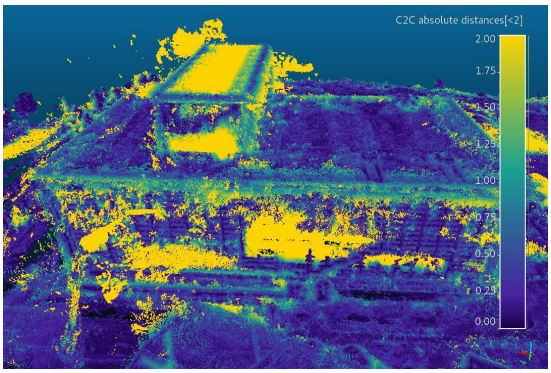
(b) Visual SFM



(c) Visual SFM + openMVS

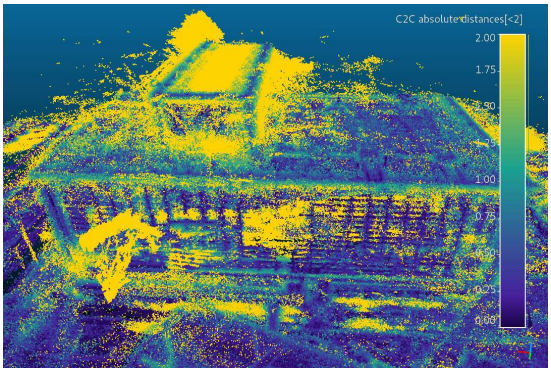


(d) Visual SFM + MVE

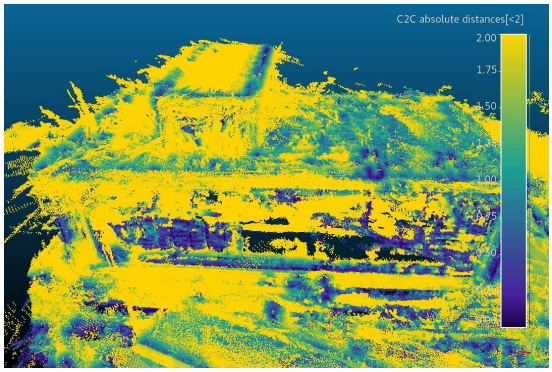


(e) openMVG + openMVS

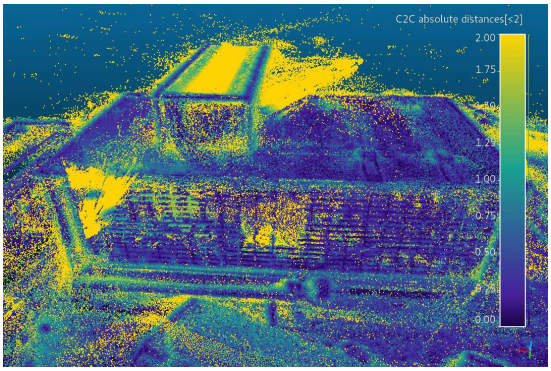
Figure B.10: Fourth view of the produced point clouds by all pipelines of the practice hall at the firefighter school model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. PhotoScan failed to generate any structure for this area of the building. Most pipelines generated large errors in this section as well. The projection artefacts generated by COLMAP are also shown on the top of the building.



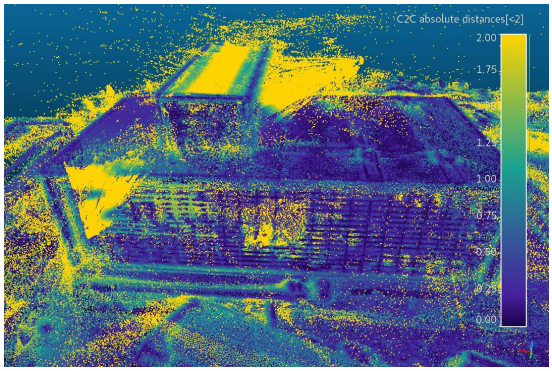
(f) openMVG + MVE



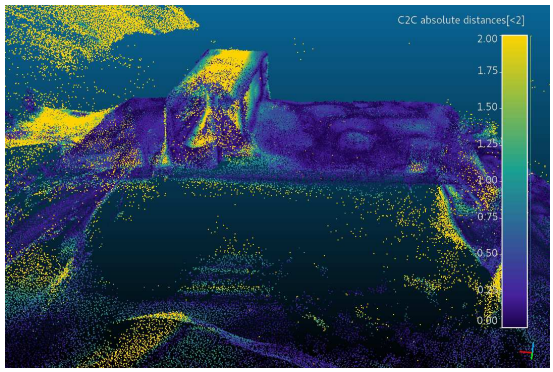
(g) MVE



(h) COLMAP Photometric



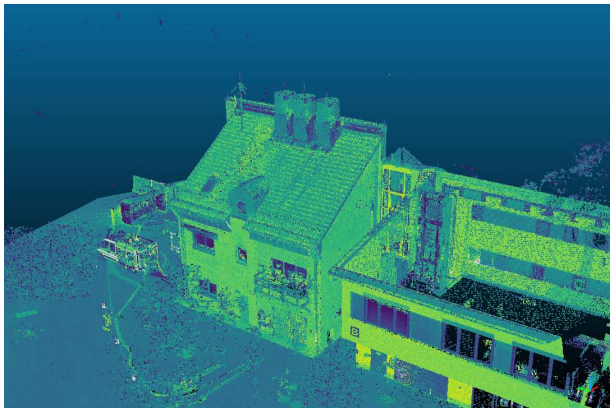
(i) COLMAP Geometric



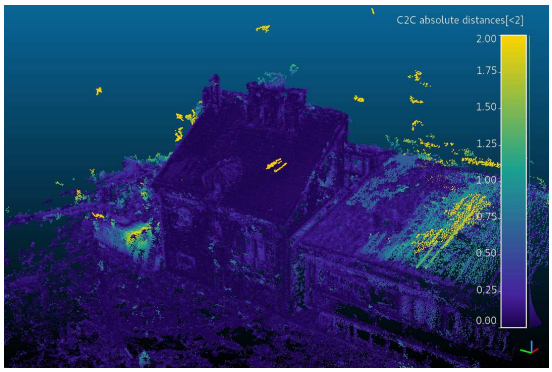
(j) PhotoScan

Figure B.10: Fourth view of the produced point clouds by all pipelines of the practice hall at the firefighter school model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. PhotoScan failed to generate any structure for this area of the building. Most pipelines generated large errors in this section as well. The projection artefacts generated by COLMAP are also shown on the top of the building.

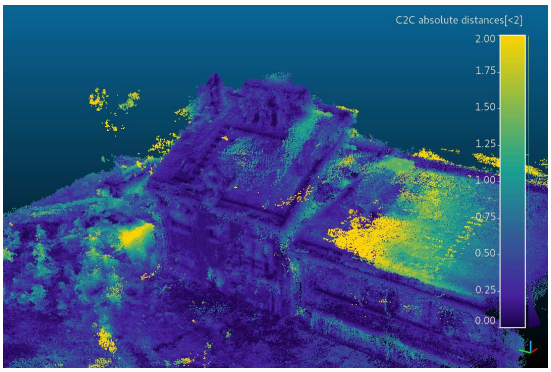
B.4 Burning house at the firefighter school



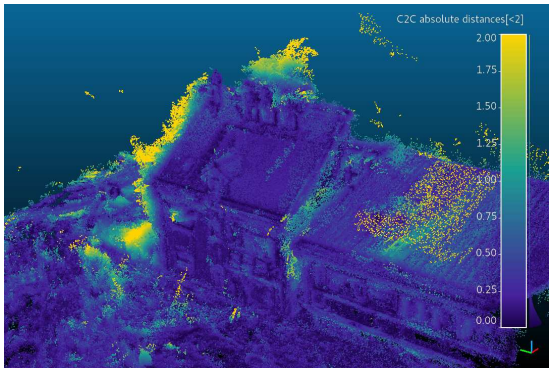
(a) Reference frame from Laser Scanner



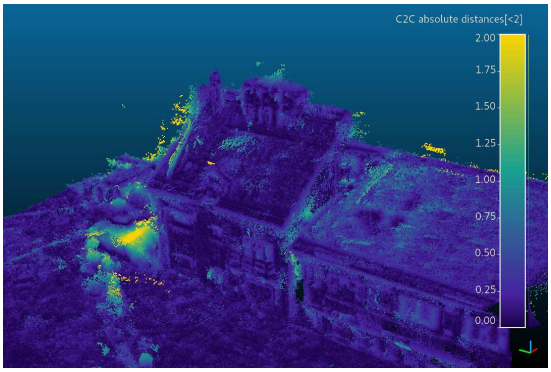
(b) Visual SFM



(c) Visual SFM + openMVS

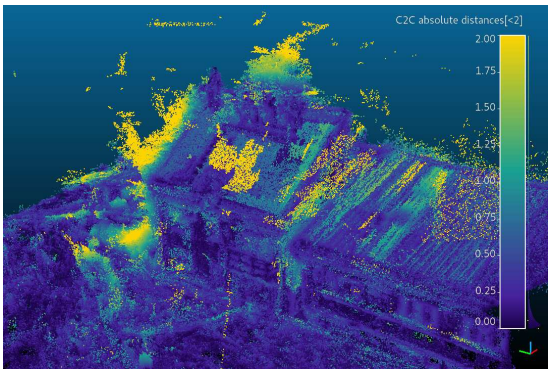


(d) Visual SFM + MVE

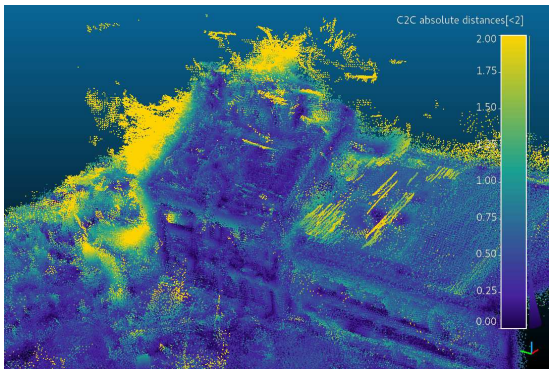


(e) openMVG + openMVS

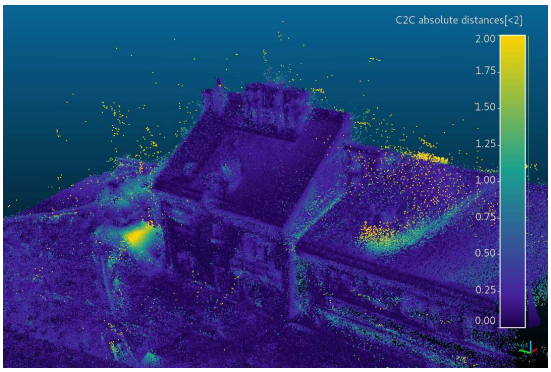
Figure B.11: First view of the produced point clouds by all pipelines of the burning house at the firefighter school model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. Smoke noise is visible for some models behind the structure.



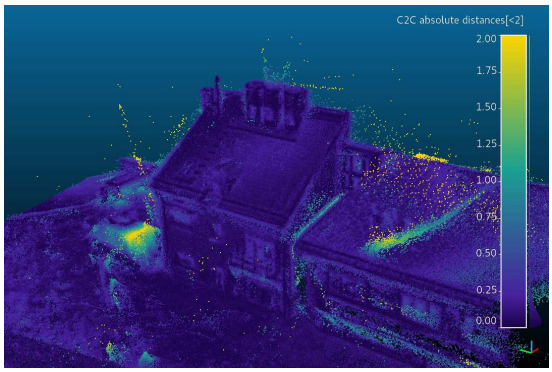
(f) openMVG + MVE



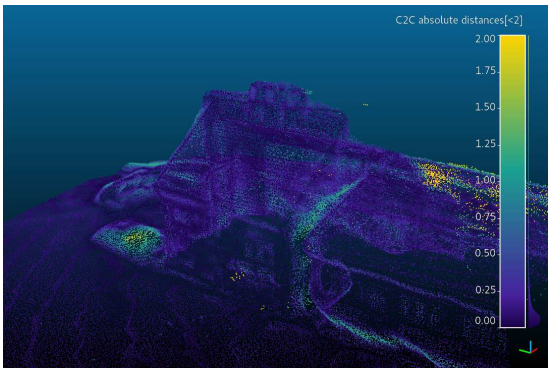
(g) MVE



(h) COLMAP Photometric

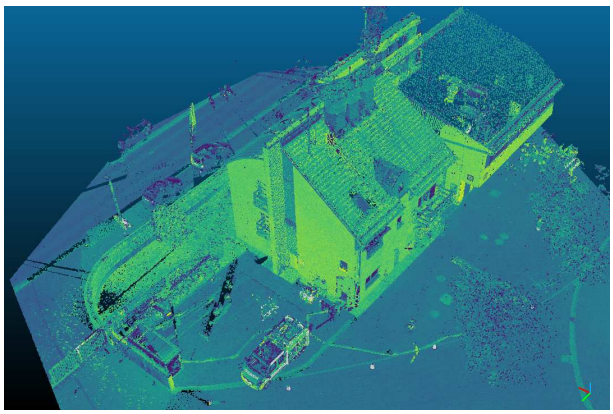


(i) COLMAP Geometric

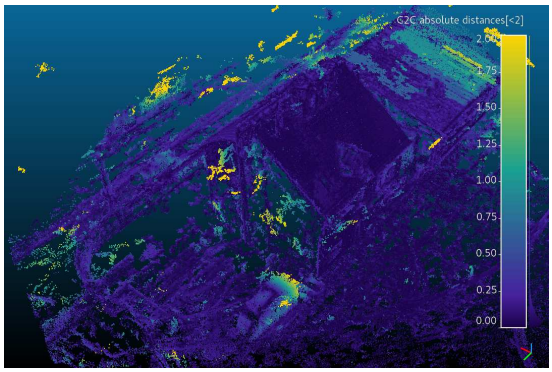


(j) PhotoScan

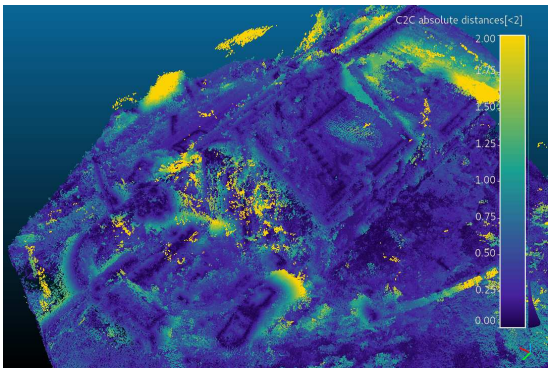
Figure B.11: First view of the produced point clouds by all pipelines of the burning house at the firefighter school model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. Smoke noise is visible for some models behind the structure.



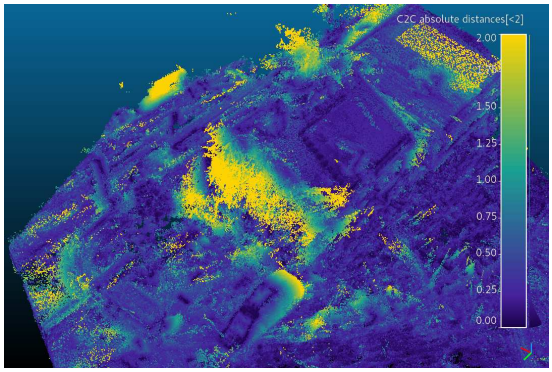
(a) Reference frame from Laser Scanner



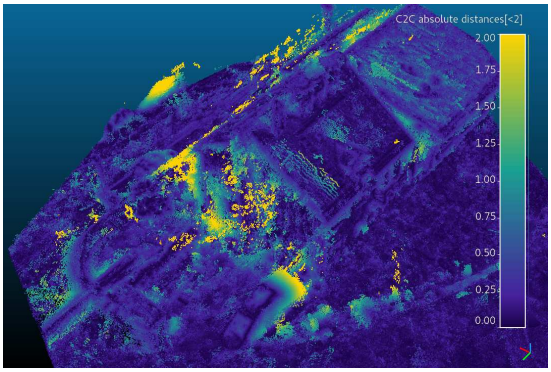
(b) Visual SFM



(c) Visual SFM + openMVS

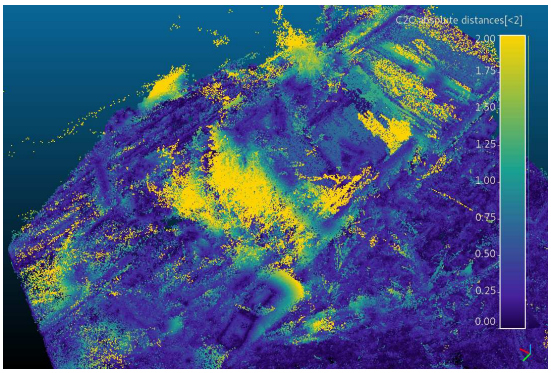


(d) Visual SFM + MVE

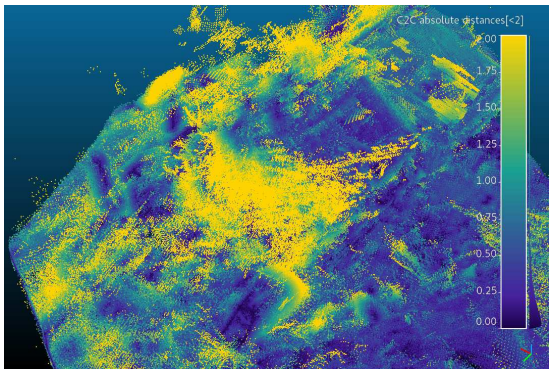


(e) openMVG + openMVS

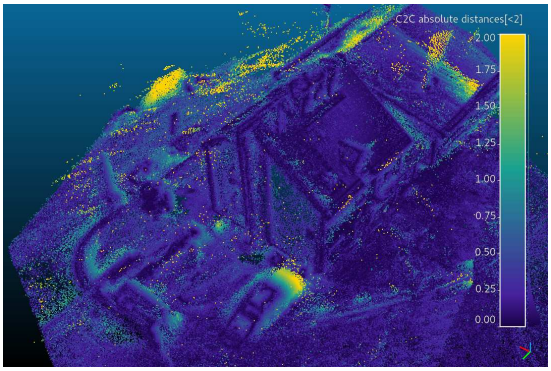
Figure B.12: Second view of the produced point clouds by all pipelines of the burning house at the firefighter school model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. Algorithms where the expansion stage was made by MVE show the most noise in this view.



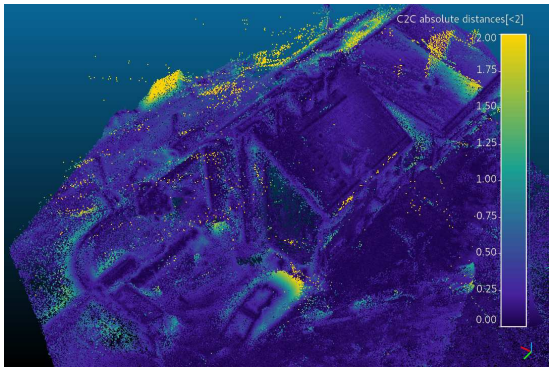
(f) openMVG + MVE



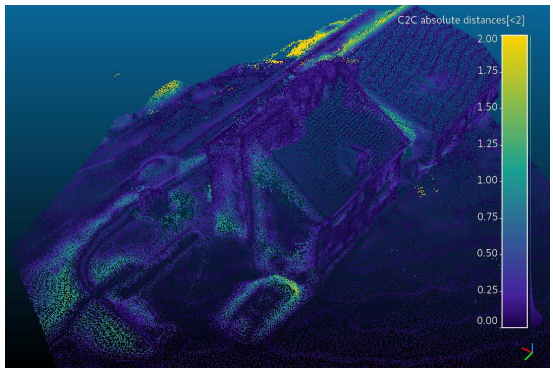
(g) MVE



(h) COLMAP Photometric

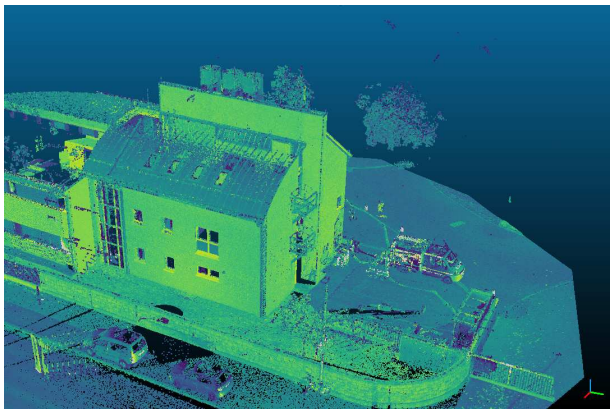


(i) COLMAP Geometric

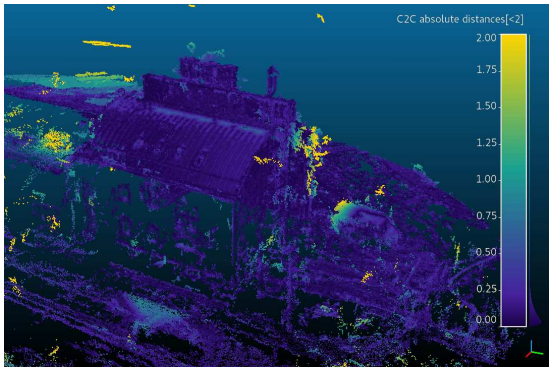


(j) PhotoScan

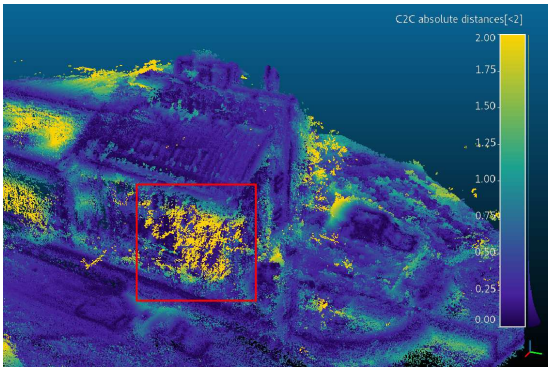
Figure B.12: Second view of the produced point clouds by all pipelines of the burning house at the firefighter school model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. Algorithms where the expansion stage was made by MVE show the most noise in this view.



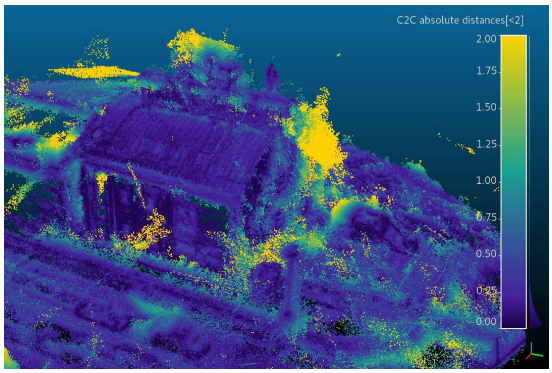
(a) Reference frame from Laser Scanner



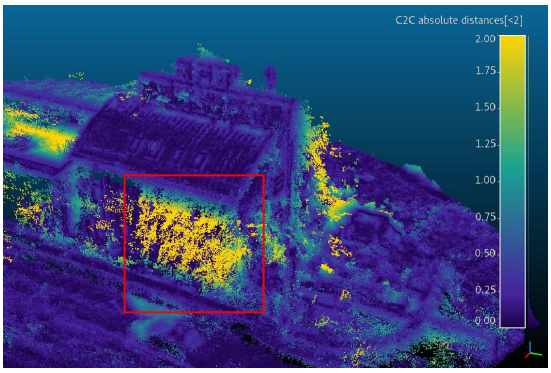
(b) Visual SFM



(c) Visual SFM + openMVS

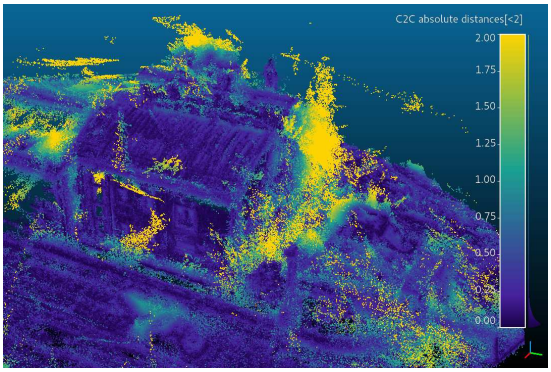


(d) Visual SFM + MVE

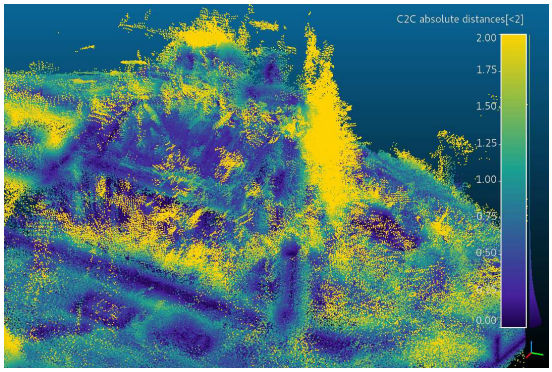


(e) openMVG + openMVS

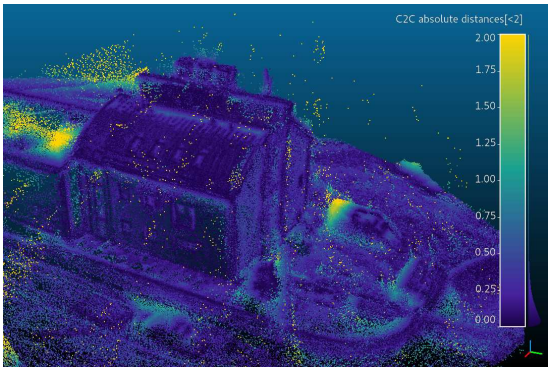
Figure B.13: Third view of the produced point clouds by all pipelines of the burning house at the firefighter school model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. openMVS also generated a small amount of noise due to smoke behind the building. This is marked in red in the figure. This noise is made by the smoke rising over the shadow of the building.



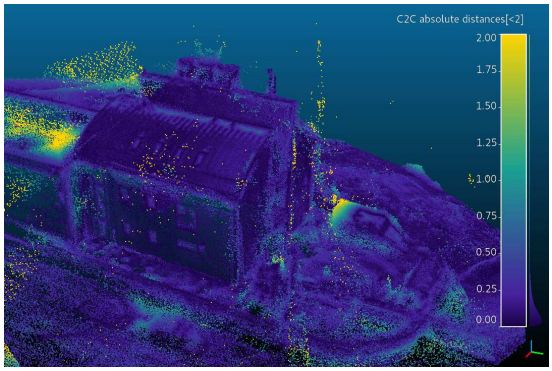
(f) openMVG + MVE



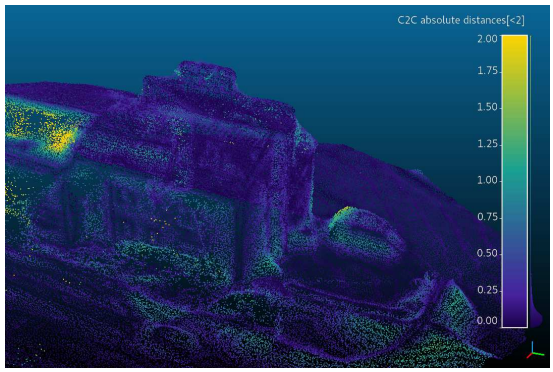
(g) MVE



(h) COLMAP Photometric



(i) COLMAP Geometric



(j) PhotoScan

Figure B.13: Third view of the produced point clouds by all pipelines of the burning house at the firefighter school model, colouring the points with their respective cloud to cloud distance error when registered and compared against the laser scanner reference frame. The first image shows the same view of the laser scanner reference model. openMVS also generated a small amount of noise due to smoke behind the building. This is marked in red in the figure. This noise is made by the smoke rising over the shadow of the building.

Bibliography

[1] C. Barnes, E. Shechtman, A. Finkelstein, and D.B. Goldman. PatchMatch: A Randomized Correspondence Algorithm for Structural Image Editing. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 24:1–24:11, New York, NY, USA, 2009. ACM.

[2] A. Baumberg. Reliable Feature Matching Across Widely Separated Views. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, 09 2001.

[3] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008. Similarity Matching in Computer Vision and Multimedia.

[4] P.A. Beardsley, A. Zisserman, and D.W. Murray. Sequential Updating of Projective and Affine Structure from Motion. *International Journal of Computer Vision*, 23(3):235–259, Jun 1997.

[5] D.C. Brown. The Bundle Adjustment - Progress and Prospects. *XIII. Congress of the International Society for Photogrammetry*, 1976.

[6] M. Brown. *Advanced Digital Photography.* Media Publishing Pty, Limited, 2004.

[7] M. Brown and D.G. Lowe. Invariant Features from Interest Point Groups. In *BMVC*, volume 4, 2002.

[8] A.P. Cracknell and L. Hayes. *Introduction to remote sensing.* CRC Press, 2007.

[9] R. Deriche, Z. Zhang, Q. Luong, and O. Faugeras. *Robust recovery of the epipolar geometry for an uncalibrated stereo rig*, pages 567–576. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[10] A. Desolneux, L. Moisan, and J.M. Morel. *From Gestalt Theory to Image Analysis: A Probabilistic Approach.* Springer Publishing Company, Incorporated, 1st edition, 2007.

[11] O. Faugeras. *Three-dimensional computer vision: a geometric viewpoint.* MIT press, 1993.

[12] O. Faugeras and S. Maybank. Motion from point matches: Multiplicity of solutions. *International Journal of Computer Vision*, 4(3):225–246, Jun 1990.

[13] M.A. Fischler and R.C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[14] S. Fuhrmann and M. Goesele. Floating Scale Surface Reconstruction. *ACM Trans. Graph.*, 33(4):46:1–46:11, July 2014.

[15] S. Fuhrmann, F. Langguth, and M. Goesele. MVE: A Multi-view Reconstruction Environment. In *Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage*, GCH '14, pages 11–18, Aire-la-Ville, Switzerland, Switzerland, 2014. Eurographics Association.

[16] Y. Furukawa and J. Ponce. Accurate, Dense, and Robust Multiview Stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1362–1376, August 2010.

[17] S.A. Genchi, A.J. Vitale, G.M.E. Perillo, and C.A. Delrieux. Structure-from-Motion Approach for Characterization of Bioerosion Patterns Using UAV Imagery. *Sensors*, 15(2):3593–3609, 2015.

[18] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S.M. Seitz. Multi-View Stereo for Community Photo Collections. pages 1–8, 11 2007.

[19] S. Haner and A. Heyden. *Covariance Propagation and Next Best View Planning for 3D Reconstruction*, pages 545–556. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[20] R. Hartley. *Estimation of relative camera positions for uncalibrated cameras*, pages 579–587. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992.

[21] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[22] B.K.P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642, Apr 1987.

[23] M. Jaud, S. Passot, R. Le Bivic, C. Delacourt, P. Grandjean, and N. Le Dantec. Assessing the Accuracy of High Resolution Digital Surface Models Computed by PhotoScan® and MicMac® in Sub-Optimal Survey Conditions. *Remote Sensing*, 8(6), 2016.

[24] F.A. Jenkins and H.E. White. *FUNDAMENTALS OF OPTICS*. McGraw-Hill, 1950.

[25] D.G. Jones and J. Malik. A computational framework for determining stereo correspondence from a set of linear spatial filters. In *Computer Vision — ECCV'92: Second European Conference on Computer Vision Santa Margherita Ligure*, pages 395–410, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[26] P. Koppel. Agisoft Photoscan: Point Cloud accuracy in close range configuration. 2015.

[27] A. Kushal and J. Ponce. Modeling 3D Objects from Stereo Views and Recognizing Them in Photographs. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part II*, ECCV'06, pages 563–574, Berlin, Heidelberg, 2006. Springer-Verlag.

[28] K.N. Kutulakos and E. Steger. A Theory of Refractive and Specular 3D Shape by Light-Path Triangulation. *International Journal of Computer Vision*, 76(1):13–29, Jan 2008.

[29] K. Levenberg. A METHOD FOR THE SOLUTION OF CERTAIN NON-LINEAR PROBLEMS IN LEAST SQUARES. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

[30] T. Lindeberg and J. Gårding. Shape-adapted smoothing in estimation of 3-D shape cues from affine deformations of local 2-D brightness structure. *Image Vision Comput.*, 15:415–434, 1997.

[31] H. C. Longuet-Higgins. In Martin A. Fischler and Oscar Firschein, editors, *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms*, chapter A Computer Algorithm for Reconstructing a Scene from Two Projections, pages 61–62. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[32] D.G. Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision, 1999*, pages 1150–1157. IEEE, 1999.

[33] D.G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004.

[34] L. Mach. Sift keypoints filtering, 2008. Used under CC-BY 3.0. Last access on [2017-08-19]. URL: `https://commons.wikimedia.org/wiki/File:Sift_keypoints_filtering.jpg`.

[35] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings National Institute of Science, India*, volume 2, pages 49–55, April 1936.

[36] F. Mancini, M. Dubbini, M. Gattelli, F. Stecchi, S. Fabbri, and G. Gabbianelli. Using Unmanned Aerial Vehicles (UAV) for High-Resolution Reconstruction of Topography: The Structure from Motion Approach on Coastal Environments. *Remote Sensing*, 5(12):6880–6898, 2013. `doi:10.3390/rs5126880`.

[37] L. Moisan, P. Moulon, and P. Monasse. Automatic Homographic Registration of a Pair of Images, with A Contrario Elimination of Outliers. *Image Processing On Line*, 2:56–73, 2012.

[38] L. Moisan and B. Stival. A Probabilistic Criterion to Detect Rigid Point Matches Between Two Images and Estimate the Fundamental Matrix. *International Journal of Computer Vision*, 57(3):201–218, May 2004.

[39] P. Moulon, P. Monasse, and R. Marlet. Adaptive Structure from Motion with a Contrario Model Estimation. In *Computer Vision – ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part IV*, pages 257–270, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[40] I. Nikolov and C. Madsen. *Benchmarking Close-range Structure from Motion 3D Reconstruction Software Under Varying Capturing Conditions*, pages 15–26. Springer International Publishing, Cham, 2016.

[41] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann. 6D SLAM – 3D mapping outdoor environments. *Journal of Field Robotics*, 24(8-9):699–722, 2007.

[42] D. Panagiotidis, P. Surovỳ, K. Kuželka, et al. Accuracy of Structure from Motion models in comparison with terrestrial laser scanner for the analysis of DBH and height influence on error behaviour. *J. FOR. SCI*, 62(8):357–365, 2016.

[43] F. Remondino, M.G. Spera, E. Nocerino, F. Menna, and F. Nex. State of the art in high density image matching. *The Photogrammetric Record*, 29(146):144–166, 2014.

[44] D.P. Robertson and R. Cipolla. *Practical Image Processing and Computer Vision*, chapter Structure from Motion. John Wiley, 2009.

[45] P. Saponaro, S. Sorensen, S. Rhein, A. Mahoney, and C. Kambhamettu. Reconstruction of textureless regions using structure from motion and image-based interpolation. In *2014 IEEE International Conference on Image Processing, ICIP 2014*, pages 1847–1851, 01 2015.

[46] J.L. Schönberger and J.M. Frahm. Structure-from-Motion Revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[47] J.L. Schönberger, E. Zheng, J.M. Frahm, and M. Pollefeys. *Pixelwise View Selection for Unstructured Multi-View Stereo*, pages 501–518. Springer International Publishing, Cham, 2016.

[48] D. Semyonov. Algorithms used in Photoscan, 2011. Last access on [2017-09-04]. URL: `http://www.agisoft.com/forum/index.php?topic=89.msg323#msg323`.

[49] S. Solbø and R. Storvold. Mapping Svalbard glaciers with the Cryowing UAS. In *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XL-1/W2, 09 2013.

[50] P. Sturm and B. Triggs. *A factorization based algorithm for multi-image projective structure and motion*, pages 709–720. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

[51] K. Thoeni, A. Giacomini, R. Murtagh, and E. Kniest. A Comparison of Multi-view 3D Reconstruction of a Rock Wall using Several Cameras and a Laser Scanner. In *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XL-5, pages 573–580, 06 2014.

[52] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, Nov 1992.

[53] P.H.S. Torr, A.W. Fitzgibbon, and A. Zisserman. The Problem of Degeneracy in Structure and Motion Recovery from Uncalibrated Image Sequences. *International Journal of Computer Vision*, 32(1):27–44, Aug 1999.

[54] B. Triggs, P.F. McLauchlan, R.I. Hartley, and A.W. Fitzgibbon. *Bundle Adjustment — A Modern Synthesis*, pages 298–372. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[55] G. Verhoeven. Taking computer vision aloft – archaeological three-dimensional reconstructions from aerial photographs with photoscan. *Archaeological Prospection*, 18(1):67–73, 2011.

[56] W. Weibull. A statistical distribution function of wide applicability. *Journal of Applied Mechanics*, 18:293–297, 1951.

[57] "M.J. Westoby, J. Brasington, N.F. Glasser, M.J. Hambrey, and J.M. Reynolds. 'Structure-from-Motion' photogrammetry: A low-cost, effective tool for geoscience applications. *Geomorphology*, 179(Supplement C):300 – 314, 2012.

[58] A.P. Witkin. Scale-space Filtering. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'83, pages 1019–1022. Morgan Kaufmann Publishers Inc., 1983.

[59] C. Wu. Towards Linear-Time Incremental Structure from Motion. In *Proceedings of the 2013 International Conference on 3D Vision*, 3DV '13, pages 127–134, Washington, DC, USA, 2013. IEEE Computer Society.

[60] C. Wu, S. Agarwal, B. Curless, and S.M. Seitz. Multicore bundle adjustment. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 42, pages 3057–3064, 06 2011.

Proclamation

Hereby I confirm that I wrote this thesis independently and that I have not made use of any other resources or means than those indicated.

Würzburg, September 2017