Automated Tool for Visual Progress Monitoring in Construction

by

Oleksii Martens

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Construction Engineering and Management

Department of Civil and Environmental Engineering

University of Alberta

Abstract

This thesis focuses on determining innovative computer vision algorithms suitable for progress tracking and forming them into automated visual progress tracking framework. The main concept of this approach is reconstruction of an as-built 3D point cloud model of the object of interest based on the spatial information extracted from photographs or videos. The reconstructed as-built model is then compared to the as-planned model, and the progress is reported based on correlation of the as-built and the as-planned models.

The perspective framework key components are Structure from Motion, Multi-View Stereo, Coherent Point Drift or Iterative Closest Point, and the Hausdorff distance. At the initial phase, Structure from Motion takes a set of images as an input, estimates camera parameters for each image and produces a sparse point cloud. Next, the obtained data passes to Multi-View Stereo and the dense point cloud is generated. At this stage, the acquired point cloud is the as-built model. The next phase is aligning of the reconstructed as-built model to the corresponding as-planned model. The alignment transformation is calculated with either Coherent Point Drift or Iterative Closest Point. Finally, having the point cloud aligned, the progress is estimated. This phase is performed in three steps. First, the Hausdorff distance is calculated. Second, the as-planned model is color coded with a binary palette, where one color corresponds to the completed parts of the construction object and another corresponds to the parts that are to be built. Third, the ratio of completed points to the number of all points is computed. Finally, the color-coded progress model and percentage of completion are reported to the end user.

The perspective cutting edge libraries for Structure from Motion are COLMAP, OpenMVG, VisualSFM, TheiaSFM, and MVE. The chosen libraries for Multi-View Stereo are COLMAP, OpenMVS, CMVS, CMPMVS and MVE. The libraries selected for point cloud alignment are CPD (Coherent Point Drift) and libpointmatcher (Iterative Closest Point). Finally, the Hausdorff distance is computed with Meshlab.

The chosen libraries are integrated into frameworks and tested on real case study data obtained from a construction company. The case study experiment indicated successful performance of the following frameworks: COLMAP-COLMAP-CPD-Meshlab, COLMAP-COLMAP-libpointmatcher-Meshlab, and OpenMVG-OpenMVS-CPD-Meshlab. These pipelines demonstrated their capability of performing reliable and fast progress identification. Thus, the specified software combinations are suggested for construction progress tracking. The proposed frameworks improve the cutting-edge computer vision construction project progress monitoring approaches in terms of reconstruction quality and computation time. In fact, the suggested progress monitoring approach allows to reduce progress identification time from a few hours to a few minutes.

Contents

List of Tables

List of Figures

Abbreviations

HSI	Hue Saturation Intensity
ICP	Iterative Closest Point
LiDAR	Light Detection And Ranging
MSE	Mean Square Error
MVS	Multi-View Stereo
PCA	Principal Component Analyses
QA	Quality Assessment
QC	Quality Control
RANSAC	Random Sample Consensus
RFID	Radio Frequency Identification
SGM	Semi-Global Matching
SfM	Structure from Motion
SIFT	Scale Invariant Feature Transform
SVM	Support Vector Machine
TLS	Terrestrial Laser Scanning
UAV	Unnamed Aerial Vehicles
WBS	Work Breakdown Structure

Chapter 1

Introduction

1.1 Background

Project progress monitoring is a crucial section of project management (Zubair, Zaimi, Majid, and Mushairry (2006)). It undermines capturing an actual project completion phase, comparing it to the schedule and reporting project performance (Marr (2007)). Progress monitoring of construction projects traditionally includes daily and/or weekly foremen handwritten progress reports. Then, these reports are typed in to planning software such as Primavera P6, Microsoft Project or even Microsoft Excel. The updated project information is later reviewed by project managers along with as-planned 3D model or 2D drawings, initial schedules and other construction documentation. Finally, project managers analyze project progress and develop a plan for further work. Hence, the limitation of manual monitoring is that it is an approximate, subjective, slow and time-consuming method of progress tracking (Turkan, Bosche, Haas, & Haas, 2012).

Because of the manual progress monitoring nature, the ongoing progress is also often reported with delays and errors. These circumstances make it difficult to identify possible schedule delays and estimate overruns in advance (Sacks, Navon, Brodetskaia, and Shapira (2005)). About 12.4 % of construction cost is determined to be due to the delays in problem identification (Leung, Mak, and Lee (2008)).

Hence, project monitoring is critical for in-time and on-budget project completion. As such, progress monitoring is considered to be one of the most demanding problems of the construction industry (Jin and Le (2014)).

Effective project monitoring (or project tracking) can be achieved by establishing and pursuing the following practices: meaningful performance metrics at the planning phase, well-defined milestones, rigorous tracking of the project performance, achievements reward schemes and reviews and audits (Jin and Le (2014)). While manual project tracking struggles with accomplishing the above mentioned programs, automated progress tracking can be a key solution for making project monitoring more efficient.

Automated progress monitoring is a mechanism allowing minimization of site visits, up-to-date more precise control of project schedule and earned value, and even quality control in some cases (X. Zhang et al. (2009a)). It also permits early correction of human errors, decreases the number of man-hours needed for progress control and eliminates project monitoring delays (Ahmed et al. (2012)). Further, automated progress monitoring can be extended to the following features:

- automated warning alarms (for required/constrained parameters);
- continuous risk analysis;
- continuous updating and improvement of schedules, estimates, and processes;
- control of human recourses allocation;
- equipment administration (especially heavy equipment);
- keeping supply chain informed;
- updating as-built model;
- quality control;
- safety control;

- supplied/used/required materials quantity control.

There are many technologies allowing automated or partially automated progress tracking: bar coding, Radio Frequency Identification (RFID), Global Positioning Systems (GPS), multimedia and pen-based computers, 3D laser scanning, and computer vision (El-Omari and Moselhi (2011), Turkan, Bosche, et al. (2012)). These technologies have some similarities as well as some differences. Thus, Figure 1.1 shows the automated progress tracking approaches schema organized based on their similarities. The benefits and limitations of each of these methods are discussed below.

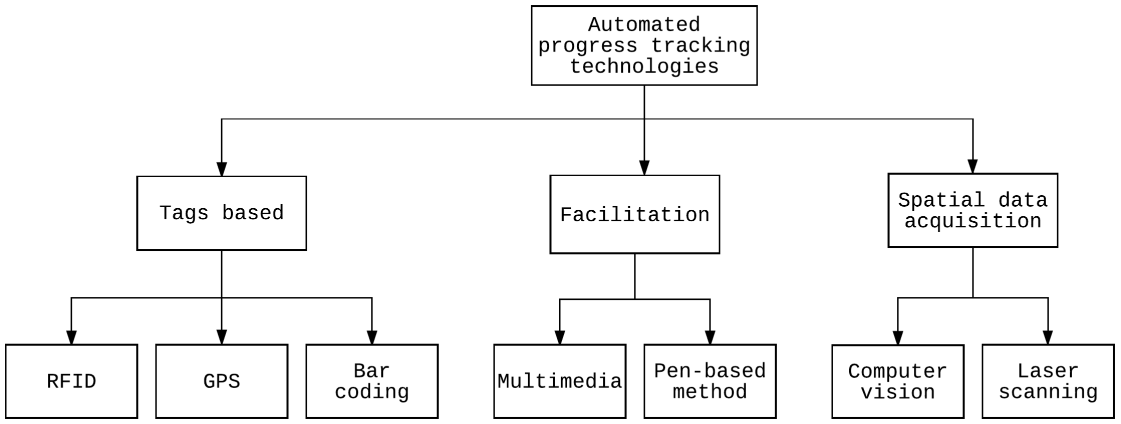


Figure 1.1: Automated technologies for progress tracking

Bar coding, RFID, UWB and GPS is one group of methods for progress monitoring based on their similarity of logic of employment. These methods may assist with materials, tools and equipment tracking.

Bar coding stands for optical machine-readable data representation corresponding to an object of interest. Bar coding uses one and two dimensional codes (QR codes). Bar codes scanners require a close contact to a sticker. The procedures of this approach are typically labor intensive. Bar coding is very simple to use; furthermore, it is even compatible with most modern mobile phones.

RFID involves tags data representation which is also to be scanned. However, RFID tag scanning does not require close contact. Its distance range is generally up to 10 m. RFID uses magnetic tags with integrated chips. Chips are capable of containing much more information than bar codes, and they are easily traceable and might be encrypted. Finally, RFID requires specific equipment, which is much more expensive and difficult to install and use.

GPS is quite similar to the RFID tracking approach. It involves employing special tags, but in contradistinction to RFID this approach does not demand any special tag-reading equipment. The limitation of GPS is that it possesses moderate location accuracy and works only out-of-doors.

Multimedia and pen-based methods are used to facilitate other project tracking approaches either manual or automatic. Multimedia data acquisition supplements project information with all possible types of multimedia files. One example of the multimedia approach might be setting up on-site web-cameras for construction project surveillance. The pen-based method implicates using tablets on site. In the mentioned scenario, the pen-based method could be potentially used for making notes on the photos obtained from these cameras. The indicated data acquisition technologies are likely to go along with the specially designed software.

Computer vision and 3D laser scanning can be combined in one group of methods as they both acquire data in point cloud format. The first step of these methods with respect to progress tracking is 3D point cloud reconstruction. After the point cloud is obtained, the rest of the steps are similar. However, the computer vision and 3D laser scanning data acquisition techniques are entirely different as well as point clouds obtained by these methods have contrasting properties.

3D laser scanners capture real-world objects and/or environments to collect spatial data by the use of laser light. Collected information is stored as a 3D point cloud with three dimensional coordinates assigned to each point. The point clouds produced by a laser scanner are dense. The scanning equipment uses a high-speed laser scanner that, for example, is capable of scanning 360° view around the horizontal axis and about 270° around the vertical axis (Golparvar-Fard et al. (2011)). Laser scanners make quick and precise measurements. However, they are extremely high priced, quite heavy, not able to correctly handle moving objects and do not recognize colors unless combined with a camera. Laser scanners also require elaborated logistics for scanning medium and large size objects, likewise combining scanned point clouds afterwards. Thus, laser scanning is not likely to be used on construction sites in working hours and is not likely to be used for materials structure identification or any kind of visual analysis.

Computer vision (also referred to as photogrammetry in this context) captures real-world objects and/or environments to extract spatial information from photo and video camera data. To clarify the terminology, computer vision is an interdisciplinary field that attempts to duplicate the effect of human vision by understanding digital images (Sonka, Hlavac, and Boyle (2008)). In turn, photogrammetry is the science of obtaining quantitative data form photos (Linder (2009)). Because these two subjects intersect, they commonly implicate the same thing in terms of progress measurement.

The photos or videos might be taken by static cameras, a drone, a worker with his smart-phone or any combination of those. As such, the photogrammetry data is easily obtainable. Hence, it is a legitimately affordable and flexible approach to progress monitoring. Nevertheless, regardless of all the advantages, computer vision method is based on complex mathematics and extremely demanding on computational resources. Lastly, a point cloud produced by computer vision algorithms are sparser than the ones obtained with laser scanners and they are prone to have some noise.

To conclude, each described method of progress tracking is convenient for specific purposes (El-Omari and Moselhi (2011)). RFID/bar-coding/GPS might be good for tracking materials or labor hours. Multimedia and pen-based methods are convenient for facilitating construction processes by, for example, taking progress photos of the construction site and making notes on these photos. Finally, computer vision and laser scanning are capable of rapidly tracking changes of the amount of work completed. In fact, a combination of some or all of these methods can be an effective way of progress monitoring.

The focus of this work is on tracking the amount of work completed automatically. Computer vision and laser scanning have a good fit to the stated purpose. These approaches flow is reconstructing an as-built construction object point cloud model, comparing it to the corresponding as-planned model and identifying the progress based on the models matching. These progress tracking techniques development, upsides and downsides are identified in Chapter 2.

Finally, considering the technology potential, equipment price, and feasibility of fast embedding of new progress tracking approaches to the actual construction industry, I chose to investigate the potential of visual progress tracking of construction projects. In this way, visual project tracking can be done without any considerable investment. Likewise, this approach is hypothetically easy and quickly embeddable to construction processes.

1.2 Computer Vision Applications

Computer vision is a powerful tool with various application. For example, computer vision algorithms have been used in the following industries other than construction:

- archeology, for heritage data documentation, reconstruction and visualization (Brutto and Meli (2012), Remondino, Del Pizzo, Kersten, and Troisi (2012));
- clothing industry, for virtual try-on application (Hauswiesner, Straka, and Reitmayr (2011));
- food industry, for external quality inspection of fruits and vegetables (B. Zhang et al. (2014), Saldaña, Siche, Luján, and Quevedo (2013)), for quality and freshness control of fish and meat (Huang et al. (2016), Jackman, Sun, and Allen (2011));
- manufacturing, for visual inspection, production process control, parts identification (Jain (2011));

- medicine, for surgery trainings and preparation (Kolivand, Tomi, Zamri, and Sunar (2015)), for detection of tumors or other divergence (Beneder, Fuechsel, Krause, Kuhn, and Mueller (2008));
- meteorology, for snow depth estimation and snow-melt high flows prediction (Miziński and Niedzielski (2017));
- security, for visual surveillance (Kale and Sharma (2012), Patrick and Bourbakis (2009)), for measurement of the human individual (Das and Meher (2013)), Sansoni, Trebeschi, and Docchio (2009));
- topography, for terrain modeling (Fonstad, Dietrich, Courville, Jensen, and Carbonneau (2013));
- wildlife management, for identification of animal species (Yu et al., 2013);
- wood industry, stock pile inventory assessment (Gibelli et al. (2016)).

In fact, computer vision has many prospective applications in the construction industry other than progress tracking:

- asset monitoring and risk management (Tsoulkas, Kostopoulos, and Leventakis (2014));
- damage assessment (Zhou, Gong, and Guo (2015));
- equipment tracking (Zhu, Ren, and Chen (2016), J. Kim and Chi (2017), Yuan, Li, and Cai (2016));
- interior finishing monitoring (Kropp, Koch, and König (2014), Hamledari and McCabe (2016));
- safety assessment (Seo, Han, Lee, and Kim (2015));
- tracking workers (Yang, Arif, Vela, Teizer, and Shi (2010));

- visual inspection of structures (Ellenberg, Branco, Krick, Bartoli, and Kontsos (2014), Eschmann and Wundsam (2017), Kropp et al. (2014), Yan, Guldur, and Hajjar (n.d.)).

This diversity of computer vision applications is an additional argument in support of investigating additional computer vision progress tracking approaches. A cutting-edge visual progress tracking tools in construction is proposed by Golparvar-Fard et al. (2010) and Golparvar-Fard, Peña-Mora, and Savarese (2012). The details of this approach are described in Sections 2.2 and 2.5. While existing approaches provide a variety of benefits, existing tools are associated with certain deficiencies described as follows.

1. Existing visual progress tracking approaches do not take advantages of top-notch computer vision algorithms.
2. As-built and as-planned model registration still requires manual input.
3. The existing computer vision progress tracking tools are not available in the public domain.

The solutions to the listed issues are delivered in this research work.

1.3 Goal and Objectives

The hypothesis underlying this work is that that recent development of computer vision techniques can be used to enhance progress tracking framework performance by targeting the following advancements:

1. Improving 3D reconstruction speed and quality by using top-notch computer vision algorithms.
2. Automation of an as-built and an as-planned model registration using appropriate techniques.

3. Repositioning a cutting-edge computer vision progress tracking implementation to the public domain by utilizing open-source software packages.

The accomplishment of the identified objectives is expected to result in the establishment of an open-source, refined progress tracking tool capable of providing reliable and fast construction progress identification.

1.4 Research Methodology

The multiphase investigation was carried out for reaching the declared goal and objectives. Initially, the current approaches to progress tracking in construction were reviewed in the existing body of knowledge, focusing on the employed algorithms, methods' merits and shortcomings, and experiment data. The gap for improvement in current techniques was identified. Afterwards, the existing approaches to progress tracking algorithms were reviewed in the computer vision domain. At this stage, the modern state of the art techniques for point cloud reconstruction and alignment were analyzed.

The identified computer vision algorithms were tested and debugged for correct compilation and stable work. Next, the algorithms combinations were integrated into frameworks for as-planned model point cloud reconstruction, as-planned and as-built models alignment, and progress identification.

Finally, the proposed frameworks were examined on a real case data obtained from the Ledcor construction company. The experiments filtered out slow and unstable algorithm combinations. Eventually, the remained frameworks demonstrated fast and reliable progress tracking performance.

1.5 Thesis Organization

The following chapters are organized in the presented below manner.

Chapter 2 provides insights into the state of the art of the spatial automated progress tracking techniques. More specifically, it describes laser scanning and computer vision approaches to construction progress tracking as well as development of theirs concepts. Then, the key aspects of computer vision and laser scanning are compared. Afterwards, the chapter closes with the computer vision research gap.

Chapter 3 introduces specific computer vision algorithms underlying the construction visual progress tracking concept. It focuses on the algorithms for point cloud reconstruction, registration and progress identification.

Chapter 4 presents the practical experiments with the cutting-edge computer vision algorithms on a real construction case data. It describes all the conducted experiments either they produced meaningful results or not as well as the challenges behind them. The effective algorithms are organized into progress tracking frameworks. Finally, the proposed framework results are reported and summarized in this chapter.

Chapter 5 finalizes the proposed computer vision progress tracking approach. It compares the obtained results to the existing approaches; as well, this chapter discusses the proposed method limitations and perspective developments.

Chapter 2

Literature review

2.1 Introduction

The principal approaches to visual progress monitoring in the construction domain are computer vision and laser scanning. These approaches are reviewed in this chapter. Particular attention however is paid to the computer vision technique as this is principally relevant to this work.

2.2 Computer Vision

Early computer vision approaches to progress measure in the construction industry are based on comparing photographs from static cameras to corresponding views of the as-planned model. Detailed description of these approaches are presented in the “Early approaches to photogrammetry progress tracking” section of this work. The modern approaches start to appear. Their core is based on reconstruction of an as-built 3D point cloud model and comparison the obtained model to the corresponding as-planned model. These approaches are conditionally split into three main streams: proprietary, commercial and open source software. Each of the streams is presented in the following subsections of this work.

2.2.1 Early Approaches to Photogrammetry Progress Tracking

Some early approaches to visual progress monitoring are presented in the research work of Rebolj, Babič, Magdič, Podbreznik, and Pšunder (2008). They proposed and implemented Automated Construction Activity Tracking System (4D-ACT), a progress analyses tool based on the 4D model and photos of construction site. The progress is reported by superimposing site images on corresponding 2D views of the time framed 3D models (as parts of the 4D model data structure). The images are segmented with the user's help. Then the images are iteratively compared based on the minimum differences between element features, and if the the difference is under a certain threshold, then the 3D model at this point of time is determined to be an actual construction progress state. In addition to the 4D-ACT, this research team proposed to facilitate their progress tracking approach with supply chain coordination and communication enhancement tools.

Around the same time, Ibrahim, Lukins, Zhang, Trucco, and Kaka (2009) presented an approach to visual automated determination of work breakdown structure (WBS) stages. This approach to automated progress monitoring compares photos from static cameras to the corresponding 3D model. At first, the WBS structure is automatically generated. Then, the photos are taken at different points of time, and the pairs of successive images are compared to each other. The difference under a certain threshold is considered to be a segment of one of the work packages. Subsequently, the recognized component identifies the current work package by comparing the acquired segment to different parts of the 3D model. Finally, the system reports progress based on the ongoing WBS stage.

Similarly, X. Zhang et al. (2009b) proposed the semi-automated work progress quantification method based on comparing the consecutive images form static cameras. The researchers used a comprehensive threshold system for change detection. As a result of this detection, altered regions are identified to be newly constructed components. The new components are then semi-automatically tied to the corresponding work package, estimate, and schedule, and the work progress is evaluated. The scientists also suggested that the newly constructed components can theoretically be automatically compared to the as-planned 3D model for automatic progress identification.

One more related research was conducted in Y. Wu, Kim, Kim, and Han (2009). The researchers monitored concrete column activities via comparing static camera photos to a corresponding perspective view of a 3D CAD model in the AutoCAD environment. The on-site photos were manually aligned to the matching 3D model perspective view, preprocessed with a canny edge detector (Canny (1986)) and watershed transformation (Beucher et al. (1992)) in parallel, then filtered using the image mask created from a 3D model perspective view of the concrete columns and finally fused. Having only columns information in the image, the columns were allocated and quantified. However, the proposed method indicated low accuracy, likely due to different lighting conditions.

The next part of this section continues development of the approaches to computer vision progress tracking in three streams based on the software application copyright.

2.2.2 Proprietary Software Approach

The modern baseline approach was introduced by Golparvar-Fard et al. (2009). A 4-four dimensional augmented reality (D^4AR) environment was proposed for progress monitoring based on unordered progress photos of a construction site. The focus of this approach is on large sites. This approach acquires casually taken images on a construction site and generates a sparse 3D point cloud with an incremental Structure from Motion (SfM) algorithm. In this research, a Scale Invariant Feature Transforms algorithm (SIFT, Lowe (2004)) is used as the foundation of the photo features detection for the SfM algorithm. Random Sample Consensus (RANSAC, Fischler and Bolles (1981)) is used for choosing the correspondent features with a consistency under different view points. Then, when the SfM point cloud is reconstructed, it is aligned with the as-built model with human input by employing the closed-form solution of absolute orientation using unit quaternions. Finally, the progress is determined visually by overlaying the as-planned model to the as-built point cloud.

In a later study by Golparvar-Fard et al. (2010), constantly improving and more affordable technologies allowed a drastic improvement of the D^{4}AR tool with comprehensive computer vision and machine learning algorithms. The as-built and as-planned clouds alignment were improved; the first as-built model is aligned to the as-planned model with the same closed-form solution of absolute orientation using unit quaternions, whereas each of the posterior as-built models is aligned to the previous as-built model with an Iterative Closest Point (ICP, Besl and McKay (1992)) algorithm. This approach to the alignment requires the user to determine corresponding points only for the first iteration, thus all the rest of the iterations are performed automatically. Further, the volumetric reconstruction of a construction object was added by applying the Multi-View Stereo (MVS, Furukawa and Ponce (2010)) package and voxel coloring and labeling algorithm. And lastly, the progress evaluation is implemented by a Bayesian probabilistic model with dynamic thresholds learned through a support vector machine (SVM) classifier.

In the successive research by Golparvar-Fard et al. (2012), the D^{4}AR environment was complemented with a dense reconstruction module and slightly improved. However, the proposed pipeline still requires some minor user input. The computational time was reported to be more than a few hours, and the researchers suggested that this computational time was adequate as the progress monitoring is generally expected to be no more than one report per day.

Similar to D^{4}AR, Karsch, Golparvar-Fard, and Forsyth (2014) developed a product called ConstructAide. ConstructAide based on the same main principles as D^{4}AR. The Karsch et al. (2014) paper describes the overall product concept as well as makes a comparison of the product to contemporary computer vision tools such as VisualSFM (C. Wu (2013)) and Potosynth4. For the first time, the domain expert evaluation was given; the participants described automated project monitoring as a desired tool, but criticized the current approach for being not scalable and exhausting.

The concept of D^{4}AR was further enhanced in the Bae, Golparvar-Fard, and White (2014) paper. The D^{4}AR environment was supplemented with the mobile augmented reality subroutine allowing field personnel to have access to the project monitoring data and make some notes to the construction 3D models with their mobile devices. To allow these data manipulations, the computation power is relocated to a server. The upgraded system name is a hybrid 4-dimensional augmented reality (HD^{4}AR). Similar to HD^{4}AR, Zollmann et al. (2014) presented augmented reality tool for visualization of construction progress.

The later works by Dimitrov and Golparvar-Fard (2014) and Han and Golparvar-Fard (2014) emphasized importance of materials recognition and classication for improving D4AR progress monitoring reliability. Han, Cline, and Golparvar-Fard (2015) proposed a theory of construction sequence rules for more stable progress tracking under limited visibility or fewer detailed 4D models and WBS. The research by Han and Golparvar-Fard (n.d.) proposed to register construction object photographs over the corresponding 3D model. This improvement produced more dense output point cloud, although required more manual work in the reconstruction process. Lin, Han, and Golparvar-Fard (2015) suggested to perform visual progress monitoring data collection for construction projects via Unnamed Aerial Vehicles (UAV). The interest to UAV was also expressed in the review by Ham, Han, Lin, and Golparvar-Fard (2016) and the 3D mapping for surveying earthwork research by Siebert and Teizer (2014).

Rashidi, Brilakis, and Vela (2014) proposed determining as-built point cloud scale automatically. The research study recommends adding a specic size plywood cube for outdoor and a letter-size piece for indoor visual data acquiring scenarios. As the dimensions of the artificial object is known, the further point cloud reconstruction is performed in absolute scale. The reported average length error for outdoor and indoor applications are 0.27 and 0.14 cm/m respectively.

The most current progress monitoring advancements are labeling of the construction object parts as walls, floor and ceiling (Perez-Perez, Golparvar-Fard, and El-Rayes (n.d.)), static occlusions filtering (Han, Muthukumar, and Golparvar-Fard (n.d.)), facilitating construction project staff communication via web-based interface (Lin and Golparvar-Fard (2016)) and improving HD^{4}AR localization process (Bae et al. (2016)).

2.2.3 Commercial Software Approach

Son and Kim (2010) proposed using Bumblebee XB3 stereo vision system along with Triclops commercial software to acquire 3D data. The specified approach produced a point cloud output from each location. Next, the produced models are combined using ICP algorithm. Finally, the as-built model is compared to the corresponding CAD model. The results are presented visually in the form of an as-built 3D CAD model and quantitatively in the form of a number of built elements out of total element quantity. The proposed approach has been tested on a steel structure, and the researchers reported 100% accuracy.

In another paper, C. Kim et al. (2011) suggested that the existing progress tracking methods lack automated as-built and as-planned models alignment. The researchers proposed using two stage alignment. In this scenario, a point cloud model was acquired by the commercial photogrammetric software PhotoModeler Scanner based on 12 photos. After acquiring the point cloud, the first alignment step Principal Component Analysis (PCA) is used for coarse registration. The second ICP is used for fine registration. The scientists reported that the proposed two-step alignment approach successfully determined the transformation parameters on the experiment data set.

The consecutive work, Son, Kim, and Kim (2011) advanced the proposed previous method with concrete detection features in the photos based on its color. The scientists suggested a few methods based on combinations of different color spaces and machine learning algorithms. Ultimately, the combination of Support Vector Machine (SVM) and Hue Saturation Intensity (HSI) color space was proposed as outperforming all the rest of the approaches in the performed experiment.

In Ahmed et al. (2012), the researchers suggested using photogrammetry for piping works monitoring. The researchers have done much research with laser scanners equipment; however, they had great motivation to continue their research with photogrammetry because of:

- small initial investment required;
- possibility to use a photo-camera on an unstable platform;
- prospect of using photogrammetry for texture identification;
- camera distance to an object of interest is flexible; in fact, it is in the range of an inch to thousands of feet;
- no logistic challenges as with laser scanners (easy repeatable);
- no eye safety or health issues;
- working temperature range of photo-cameras is much higher than that of laser scanners;
- no intervention with workers or machines;
- easy transferable;
- smoothly upgradeable equipment over time.

The basic concept is reconstruction of the pipes based on specially coded target tags. The tags are to be manually marked. Then, they are to be identified using commercial software such as PMScanner, and ultimately, the pipe network is reconstructed and compared to a 3D CAD model.

An alternative tool for progress monitoring was presented in Braun et al. (2015). The researchers proposed to use the combination of VisualSfM (C. Wu (2013)) and SURE Semi-Global Matching (SGM) commercial software (Rothermel, Wenzel, Fritsch, and Haala (2012)) for an as-built model reconstruction. The proposed concept is almost fully automated, although it still requires manual input of control points for scale identification.

2.2.4 Open Source Software Approach

A full open source approach to progress tracking of construction project progress has not been introduced to the best of the author's knowledge. However, "Comparison of Laser Scanning, Photogrammetry and SfM-MVS Pipline Applied in Structures and Artificial Surfaces" by Skarlatos and Kiparissi (2012) suggested using open-source software for surface reconstruction, including surfaces of existing buildings. In the proposed open source reconstruction pipeline, the researchers used Bundler and PMVS software packages. Bundler (Snavely, Seitz, and Szeliski (2006b)) is a SfM tool for reconstruction based on unordered image collections. It takes a set of images as an input, estimates camera parameters, detects feature correspondences, and ultimately produces a sparse point cloud as an output. PMVS (Patch-based Multi-view Stereo software, Furukawa and Ponce (2010)) is a multi-view stereo (MVS) tool. In the proposed pipeline, it takes the Bundler output and produces a dense point cloud output. Thus, the result of the proposed method is a dense point cloud.

The researchers tested their method on three different data sets taken by a Nikon D90 camera: 5 photos of a sphere of 300 mm diameter, 75 photos a facade approximately 13 m wide and 5.5 m height, and 212 photos of an electricity power station approximately 180 m^2 and 25 m in height. The Bundler-PMVS surface reconstruction pipeline demonstrated impressive results on these datasets. More information about the results is in Section 2.4.

2.3 Laser Scanners

Simultaneously with the computer vision approach to progress tracking, the laser scanning method was proposed for progress tracking automation. Bosche and Haas (2008) introduced approaching automated progress tracking with a laser scanner. The researchers suggested comparing laser scanned as-built 3D point cloud models to the as-planned ones. According to this concept, the as-planned point cloud model is created out of a geo-referenced 3D CAD model. At the same time, the as-built model is to be geo-referenced while scanned. Ultimately, the point clouds are aligned using the corresponding geo-referencing and compared to each other.

The given approach is considered to be fully automated but with only one manual step, point set registration. To be more specific, each as-built model and the as-planned model are to be registered in the mutual coordinate system using a standard n-point registration algorithm. The scientists conducted an experimented with progress tracking on a laboratory column-slab structure. The laboratory experiment results indicated 80% accuracy on object recognition, less than 30mm of dimensional accuracy and 5 min processing time of one scan. In Bosche and Haas (2008), the researchers proposed to using a specified approach for automated project control in terms of Quality Assessment (QA) and Quality Control (QC).

The proposed project control method was later advanced with a couple of major enhancements (Bosché (2010)). The first enhancement is implementation of the ICP algorithm instead of the standard n-point registration. This improved the accuracy, robustness and efficiency of the laser scanning method. Furthermore, this enhancement allowed a decrease in the number of manual steps required for the proposed project control approach. Only the initial registration should be performed manually at this point, as it still requires n-point registration. The second enhancement is adding a dimensional compliance checkup to the project control algorithm. The reported Mean Square Error (MSE) is reported to be lower than 40 mm^2. The performance of the developed approach was tested on the construction of an industrial steel structure building. Following this study, semi-automated plane-based coarse registration accompanied by ICP-based fine registration software processing of laser scanned point clouds was proposed in Bosché (2012).

In Turkan, Bosche, et al. (2012), the proposed progress monitoring tool was improved utilizing a 4D as-planned model. The updated algorithm was tested on a superstructure construction and achieved up to 98% of precision recognition rate. The proposed approach was updated with an earned value tracking feature (Turkan, Bosché, Haas, and Haas (2012)). This function was added as a spreadsheet and the earned value is computed based on the project monitoring control tool output.

In C. Kim, Son, and Kim (2013), the complex hybrid approach to structural components progress monitoring was proposed. The researchers are used a laser scanner for coordinate detection and a camera for color detection. In this way, the as-planned model is a 4D BIM model and all the structural components are pre-specified in this as-planned model. Furthermore, this study assumes that the construction process is performed strictly according to the specified BIM model plan with certain logical association. Following the assumptions, the obtained as-built model is aligned to the corresponding as-planned 3D CAD model according to the researchers' previous work on computer vision progress tracking (C. Kim et al. (2011)). Then, matching an as-built model to an as-planned one is performed via a comprehensive matching three steps pipeline; segmenting the as-planned model into corresponding structural components of the BIM model, feature extracting from both models, and verifying each component's as-built status by classifying the corresponding 3D data set as a column, a beam, a slab or other. Thus, the as-built status is identified for each pre-specified structural component.

Nahangi and Haas (2014) suggested using an updated automated progress approach for pipe spool fabrication quality control. The defects methodology in this paper is two-stage PCA-ICP alignment. Noticeably, in such an approach noise is filtered by manually adding desired objects to the region of interest. Only the region of interest is further analyzed; the other regions are ignored. Testing of this approach was performed on a laboratory set. The researchers reported more than 90% accuracy. The scientists also reported difficulties with identification of symmetrical geometries as a limitation of the proposed method. Further, this concept was improved for more robust detection of incomplete spool point clouds in Czerniawski, Nahangi, Walbridge, and Haas (2015).

The further work of Bosché and Guenet (2014) proposed automated surface flatness control as the development of the idea introduced by Bosché (2012). The researchers examined their approach on two actual concrete slabs and reported that the proposed approach is comparable and even very likely outperforms the commonly used manual control practices. The latest studies on this topic by Bosché and Biotteau (2015) and Valero and Bosché (2016) expanded the surface flatness control approach with the surface waviness analysis and automatic detection of any area of concern having unusual flatness.

Biswas, Bosché, and Suna (2015) presented an approach of smart planning of the scanner locations for optimization of the number of scans required to fully cover a construction object.

Bosché, Ahmed, Turkan, Haas, and Haas (2015) proposed a circular cross-section (e.g. pipes) object detection technique integrating the Hough transform based circular cross-section detection approach.

The further development of the C. Kim et al. (2013) concept resulted in automated schedule update product by Son, Kim, and Kwon Cho (2017). The researchers proposed to update the project schedule in Microsoft Project tied to Autodesk Revit. The 4D BIM model was made in the Syncro software package. The alignment of as-built and as-planned point cloud is performed by Cyclone from Leica Geosystems utilizing the user's manual point selection. The ongoing project status is analyzed by each model component status, and the activities plan (the start and finish dates) in Microsoft Project are updated.

2.4 Computer Vision vs Laser Scanners

In Golparvar-Fard et al. (2011), the image-based SfM 3D reconstruction approach is compared to laser scanning in terms of accuracy, data processing time, equipment price and additional features. The experiments were based on four objects: a masonry block indoors, a masonry block outdoors, an exterior column at a construction site, and an interior column at a construction site. The obtained by laser scanning and computer vision as-built models were superimposed over each other and compared. The computer vision approach is sufficiently accurate but still has slightly less precision than the laser scanning approach. The computer vision data processing time is indicated to be somewhere in the range form 10 min to 7 hrs whereas the laser scanning data is processed in 1.5 - 2 hrs (Golparvar-Fard et al. (2011)). Also, the researchers pointed out that the laser scanning method requires 8 - 16 labor-hours for equipment set up versus 0 - 1 labor-hours required for taking photographs. The equipment price for photogrammetry was estimated to be in the range of 100 - 500 USD versus the laser scanning equipment cost of 10,000 - 130,000 USD (Golparvar-Fard et al. (2011)). Regarding the additional features, the scientists indicated that both methods are capable of the progress monitoring data collection, quality control, static progress visualization and safety analysis as well as comprehensive emergency building assessment. Other than that, computer vision possesses the options of remote visualization inspection, remote decision making, dynamic progress visualization and safety analysis, site logistics visualization, construction crew and machinery analysis and rapid emergency building assessment. In contrast to all the advantages of computer vision, laser scanning's unique features are only alignment and defect inspection (Golparvar-Fard et al. (2011)). Furthermore, the laser scanning approach requires special training to operate as well as adds new project management tasks for project tracking when in fact the computer vision approach requires neither of these. As the conclusion of the above mentioned comparison, the computer vision approach to progress tracking of construction projects is likely to be more applicable on a construction site than laser scanning.

One more study by Skarlatos and Kiparissi (2012) compared the point clouds produced by lasers scanning versus the ones produced by photogrammetry SfM-MVS pipeline in term of data density, quality, registration and methodology. The experiments were based on three sets of spatial data: a small sphere, a facade and an electricity power station. There were three approaches tested: commercial software for point cloud acquisition through digital camera Zscan, free software packages for point cloud acquisition through digital camera Bundler-PMVS, and laser scanning (or terrestrial laser scanning, TLS). In conclusion, Bundler-PMVS combination is comparable with laser scanning in terms of point cloud density, accuracy and processing time as well as allowing a higher degree of automation in cases of small and medium reconstruction size, whereas on large scale objects laser scanning performs better in terms of quality and processing time. In turn, Bundler-PMVS and laser scanning certainly outperformed Zscan. Furthermore, the scientists noticed that Bundler-PMVS is open-source software, giving a dramatic cost upside to this method. At the same time, the main drawbacks of laser scanning are the high price, lack of portability, noise from moving objects and poor color acquiring.

The research by Nahangi et al. (2015) compared applying photogrammetry and laser based approaches for quality control and quality assurance (QA/QC). Autodesk 123D Catch was used for the image-based as-planned model reconstruction. The experiment was held on a multi-spool laboratory installation. The reported accuracy of the photogrammetry model is 0.369° and 0.184 cm versus 0.237° and 0.02 cm of laser scanning. Thus, photogrammetry tolerance is a legitimate method for quality control and quality assurance, although the laser based approach is more accurate. Additionally, the researchers defined the image-based technique to be significantly cheaper than the other one as well as it requires only basic prior familiarity. Based on this research, the computer vision versus laser scanning comparison summary is represented in Table 2.1.

Table 2.1: Main characteristics of the visual progress tracking approaches

Properties	Laser scanning	Computer vision
Price of the required equipment, USD	10,000 - 50,000	100 - 500
Extensive personnel training required	yes	no
Dedicated personnel required	yes	no
Accuracy	high	high enough, but less than TLS
Computation time, hrs	1.5 - 2	0.2 - 7
Equipment set up time, labor-hours	8 - 16	0 - 1
Equipment transferability	laborious	easy
Temperature sensitivity	yes	almost no
Handling dynamic objects and occlusions	no	yes
Visual inspection	no	yes
Easy upgradeable	no	yes

The table reveals that the computer vision excels laser scanning in almost all the major categories of comparison. At the same time, computer vision is not as accurate as laser scanning, although it is accurate enough for progress identification and even quality control.

2.5 Research Gap

The cores of modern computer vision approaches to construction progress tracking were dramatically developing and evolving in the 2009 - 2012 years (Golparvar-Fard et al. (2009), Golparvar-Fard et al. (2012), Y. Wu et al. (2009), C. Kim et al. (2011), Skarlatos and Kiparissi (2012)). Later on, the existing approaches were complemented with advanced features such as materials recognition (Dimitrov and

Golparvar-Fard (2014), Han et al. (n.d.), Son, Kim, Hwang, Kim, and Kang (2014)), progressive WBS sequencing (Han and Golparvar-Fard (n.d.)), construction objects labeling (Perez-Perez et al. (n.d.)), and filtering of static occlusions (Han et al. (n.d.)). However, the cutting edge computer vision technologies have greatly improved during the recent years, such that, the core technologies of computer vision progress tracking could be potentially upgraded to more stable and fast point cloud reconstruction and alignment methods. In addition, the computer vision approach has not been integrated in much of the working processes of construction companies, to the best of the author's knowledge.

A lack of testing of the computer vision approach on a variety of construction objects and in different weather conditions might cause companies to be cautious, patiently waiting until the proposed approach is better tested. Furthermore, the fact that this software is proprietorial or commercial in most of the cases makes visual progress tracking expensive or even impossible to try on actual construction. A summary of all major reported computer vision datasets is reported in Table 2.2 along with the software and algorithms employed for analysis.

According to the literature review, only 14 data sets have been used for as-built model reconstruction with only 6 of these sets used for full progress tracking. All the data sets from the literature review are shown in Table 2.2. Notably, 6 full progress tracking data sets were employed for progress tracking using only proprietary and commercial software. Overall data sets size is from 5 to 288 photos. The construction objects captured in the reported data sets are mostly of large and small sizes and of simple geometry. Some of the examples of the computer vision data are presented in Figures 2.1-2.10.

All the algorithms for alignment but the one proposed by C. Kim et al. (2011) require manual input of reference points for correct as-built and as-planned point clouds scaling and coordinate alignment. The approach to alignment introduced by C. Kim et al. (2011) is two step PCA-ICP registration. Noticeably, the proposed two steps registration has not been referenced in the later research.

Regarding the progress identification algorithms, there are three main streams: overlaying as-built models, point cloud density overlaying CAD model elements and the Bayesian probabilistic model with SVM binary classifier. The first algorithm iteratively superimposes as-built models over each other. The second algorithm determines number of points of as-built model overlaying surfaces of the corresponding as-planned model and makes a decision based on the point density on different elements. In details, if the overlaying point cloud density is higher than the specified threshold, the element is considered to be built, otherwise the element is considered not in place. The third approach is the advanced version of the second one. The threshold here is iteratively determined by SVM, and the final decision is made on probability statistics.

To conclude, the author of this work is motivated to pursue computer vision progress monitoring of construction projects based on the following reasons:

- existing progress tracing approaches employ outdated algorithms;
- existing approaches are likely to fail in complex geometry objects reconstruction or work extremely slow.

The existing computer vision progress tracking techniques could be improved in terms of reliability and computational speed and tested on industrial modules assembling process.

Table 2.2: List of major reported computer vision datasets and progress tracking algorithms/software

Paper	Dataset description	# photos resolution camera	Reconstruction algorithms/ software	Alignment algorithms/ software	Progress identification algorithm/ software	Computation time
Golparvar-Fard et al. (2009), Golparvar-Fard et al. (2010), Golparvar-Fard et al. (2012), Han and Golparvar-Fard (2014)	Residence Hall, 2008	112, down-sized to 1920x1080	incremental SfM - MVS (CMVS, Furukawa and Ponce (2010)) - voxel-coloring and labeling	closed-form solution of absolute orientation using unit quaternions (1st iteration) and ICP (rest iterations)	Bayesian probabilistic model with SVM binary classifier	not reported
	Residence Hall, 2008	160, down-sized to 1920x1080				a few hrs
	Student Dining, 2008	288, down-sized to 1920x1080				not reported
Golparvar-Fard et al. (2011)	Masonry block	53, 3872x2592, 2323x1555, Nikon D80	incremental SfM	not performed	not performed	29 - 45 min
	Masonry block	242, 1936x1296, 1162x778, Nikon D80	incremental SfM	not performed	not performed	5 - 7 hrs
	Column	80, 3872x2592, 2323x1555, Nikon D80	incremental SfM	not performed	not performed	49 - 73 min

Table 2.2: List of major reported computer vision datasets and progress tracking algorithms/software

Paper	Dataset description	# photos resolution camera	Reconstruction algorithms/ software	Alignment algorithms/ software	Progress identification algorithm/ software	Computation time
	Column	54, 3872x2592, 2323x1555, Nikon D80	incremental SfM	not performed	not performed	10 - 18 min
Ahmed et al. (2012)	Pipelines at Engineering 6 building	not reported, Canon XSi 450D	PMScanner	PMScanner, based on color-coded targets	overlaying as-built models	a few hours
Skarlatos and Kiparissi (2012)	300mm sphere	5, 4288x2848, Nikon D90	incremental SfM - MVS (Bundler-PMVS)	not performed	not performed	not reported
	13m x 5.5m facade	75, 4288x2848, Nikon D90	incremental SfM - MVS (Bundler-PMVS)	not performed	not performed	3 hrs
	Electricity power station, 180m^2 x 25m	212, 4288x2848, Nikon D90	incremental SfM - MVS (Bundler-PMVS)	not performed	not performed	not reported

Table 2.2: List of major reported computer vision datasets and progress tracking algorithms/software

Paper	Dataset description	# photos resolution camera	Reconstruction algorithms/ software	Alignment algorithms/ software	Progress identification algorithm/ software	Computation time
Son and Kim (2010)	steel building	not reported, 1280x960, Bumblebee XB3 stereo vision system	Bumblebee, Triclops		point cloud density overlaying CAD model elements	not reported
C. Kim et al. (2011)	High-rise building	12, not reported	PMScanner	Two steps registration PCA-ICP, automatic	not performed	not reported
Braun et al. (2015)	5-storey office building	not reported	PMScanner	manual location of control points	graph identification based on the as-built point cloud density overlaying as-planned model	not reported

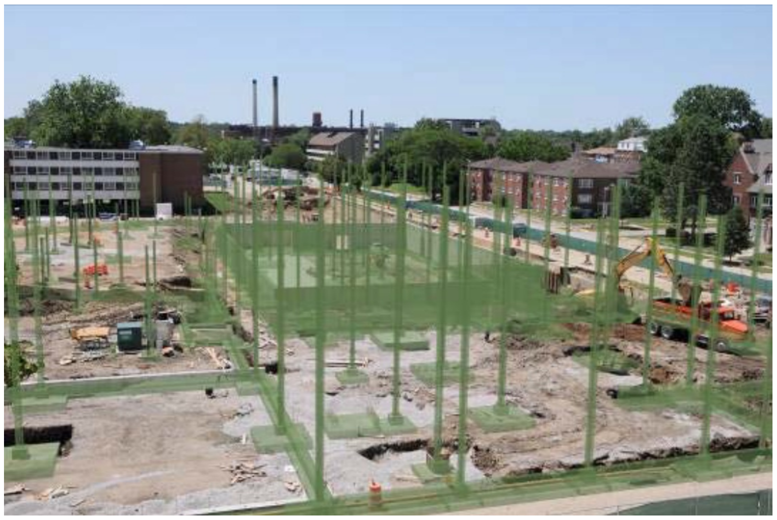


Figure 2.1: Construction progress registration over a photograph (Golparvar-Fard et al. (2009))

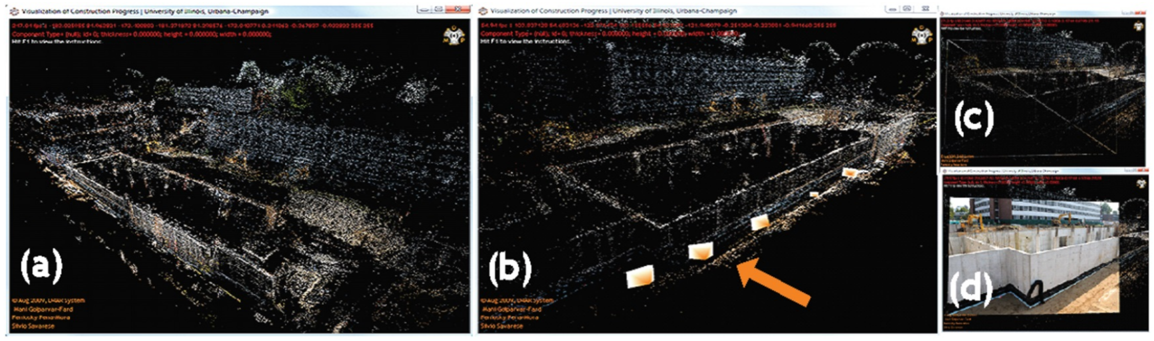


Figure 2.2: As-built point cloud example (Golparvar-Fard et al. (2010))

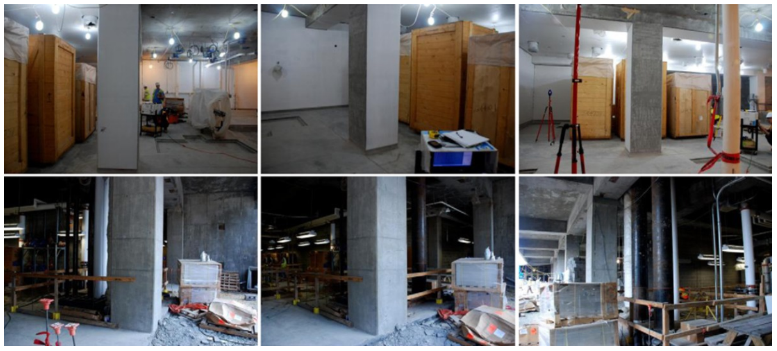


Figure 2.3: Examples of an interior and an exterior column photographs (Golparvar-Fard et al. (2011))

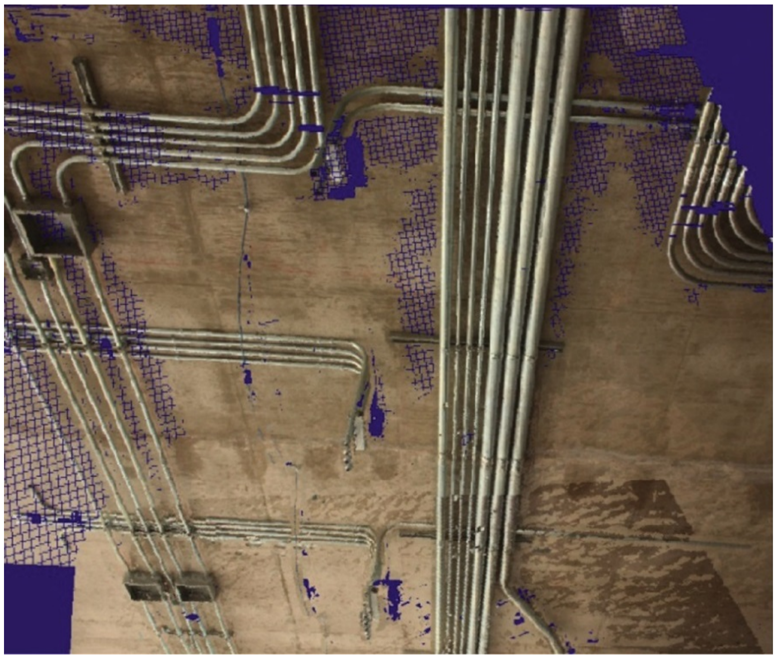


Figure 2.4: Point cloud of pipes resulting from registering several stereo-pairs together (Ahmed et al. (2012))

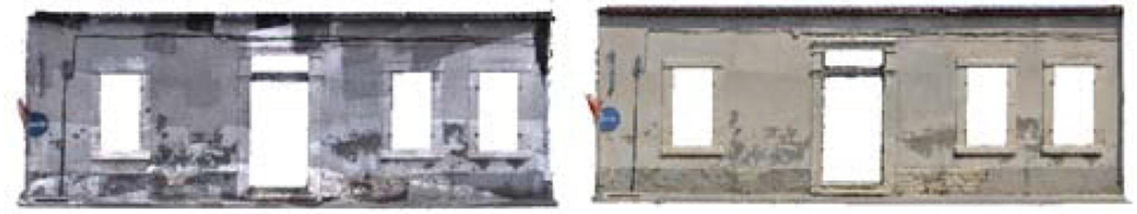


Figure 2.5: TLS model (left), PMVSall (right) (Skarlatos and Kiparissi (2012))

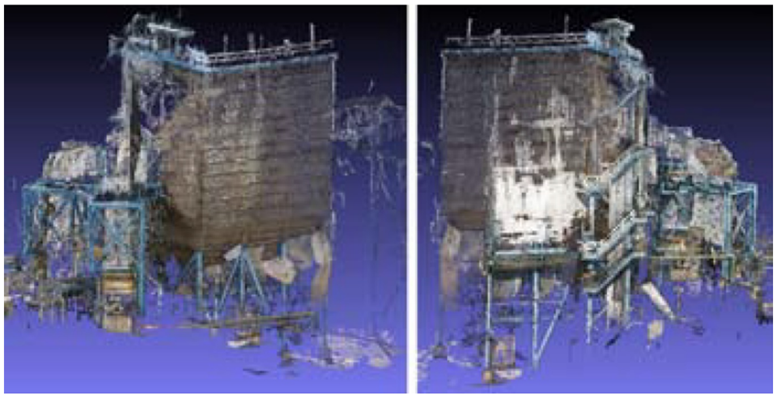


Figure 2.6: Power station reconstructed model with Bundler-PMVS (Skarlatos and Kiparissi (2012))

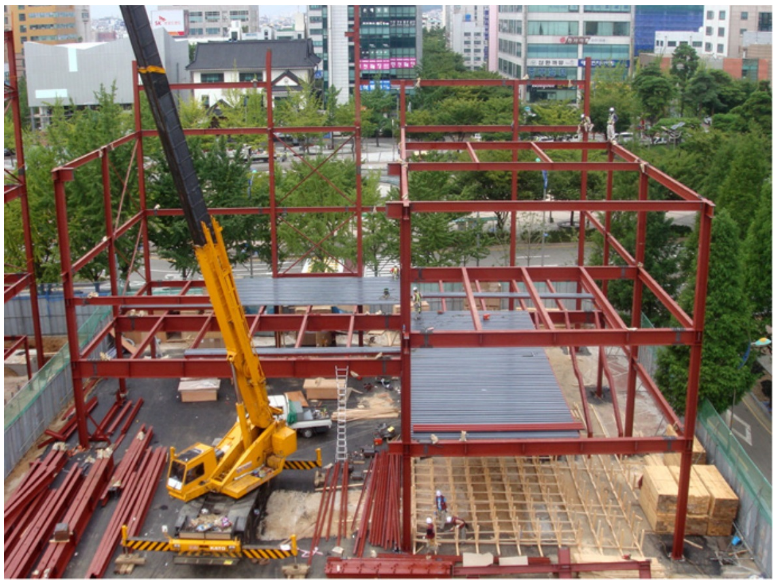


Figure 2.7: Construction-site scene example (Son and Kim (2010))

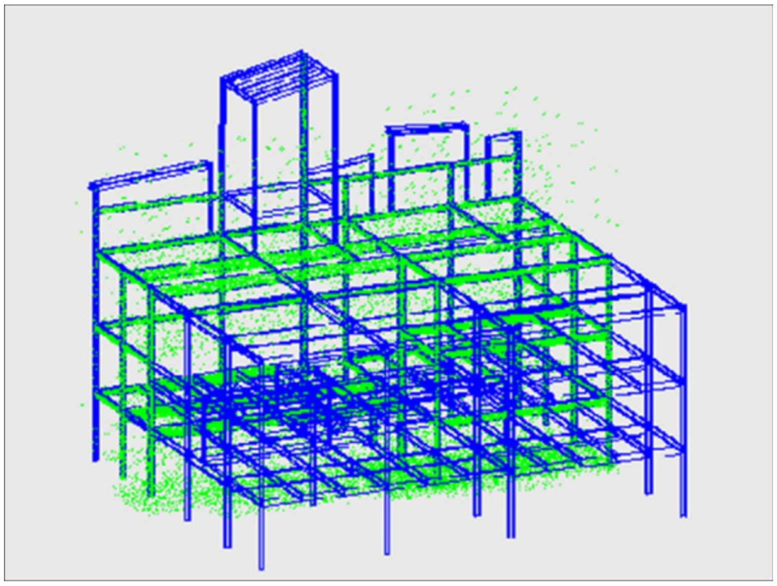


Figure 2.8: Matching of as-planned model to as-built one (Son and Kim (2010))

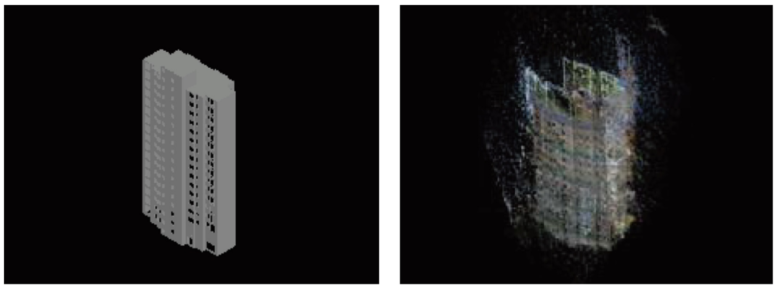


Figure 2.9: Examples of as-planned and as-built models (C. Kim et al. (2011))

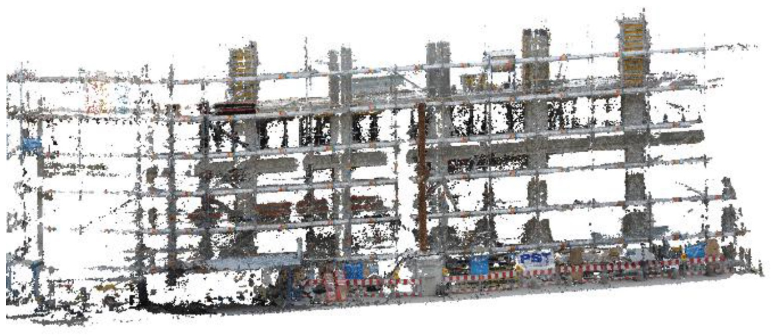


Figure 2.10: As-built point cloud (Braun et al. (2015))

Chapter 3

Proposed Methodology

3.1 Introduction

The concept of computer vision progress tracking is organized by employing a sequence of algorithms. The proposed framework is demonstrated in Figure 3.1.

The system requires construction object photographs and an as-planned 3D model as an input. The on-site photographs are passed to the point cloud reconstruction algorithm for an as-built point cloud creation. Meanwhile, the corresponding as-planned 3D model is sampled to a point cloud. The next step after having as-built and as-planned point clouds ready is their registration. This step implicates aligning point clouds in terms of scale and coordinate system. Next, the progress identification is performed based on the correspondence between the as-built and as-planned point clouds. Ultimately, the framework output is progress a report.

The final results are reported in the form of an as-planned point cloud or 3D model with parts color coded corresponding to the current progress phase. The report might additionally include the current WBS stage and the percentage of completion.

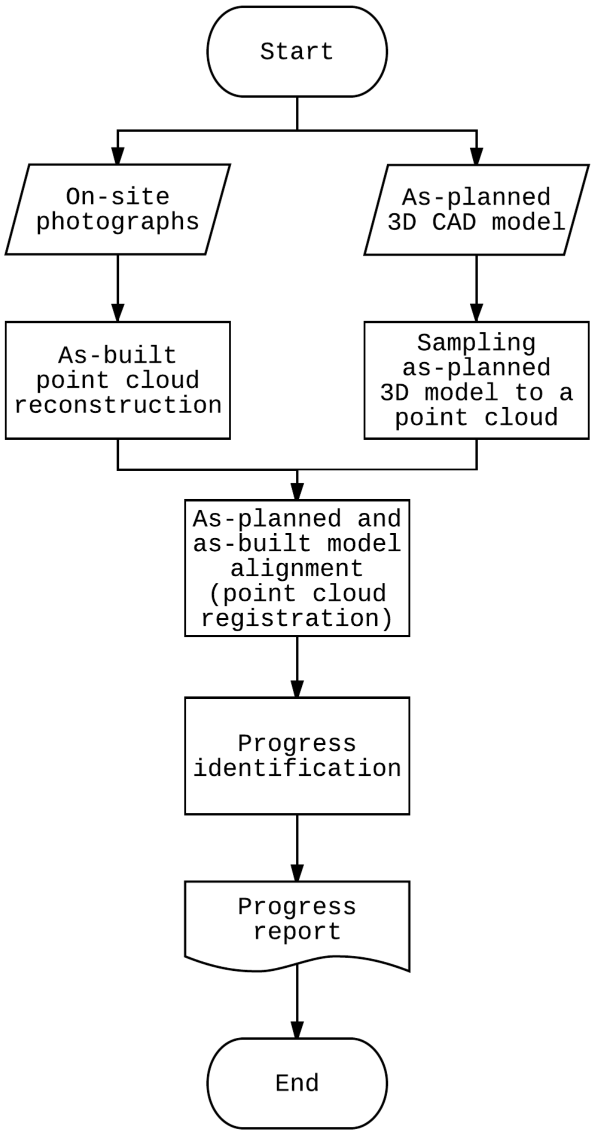

Start
On-site photographs
As-planned 3D CAD model
As-built point cloud reconstruction
Sampling as-planned 3D model to a point cloud
As-planned and as-built model alignment (point cloud registration)
Progress identification
Progress report
End

Figure 3.1: Computer vision progress monitoring flowchart

3.2 Point Cloud Reconstruction

Point cloud reconstruction creates a spatial object consisting of points, a 3D point cloud, based on 2D images. Considering that visual progress tracking input data is on-site photographs, and output is as-built point cloud model. Point cloud reconstruction is a fundamental part of computer vision progress monitoring.

A pair of Structure from Motion (SfM) and Multi-View Stereo (MVS) algorithms are commonly used for point cloud reconstruction from multiple photos in different domains such as robotics, medicine, gaming, archeology and construction. SfM is used for sparse point cloud reconstruction of an object of interest and camera parameters estimation based on a series of photographs. The SfM output is passed to MVS. The MVS algorithm produces a final dense point cloud. The produced point cloud is an as-built model. The SfM and MVS algorithms are covered in details later in this chapter.

3.2.1 Structure from Motion

Structure from Motion (SfM) is a computer vision method for three-dimensional object (the "structure") reconstruction from multiple two-dimensional images of a static scene (Hartley and Zisserman (2003)). This algorithm also computes camera position and orientation (the "motion") corresponding to each of the images. The SfM problem is not trivial because the image capturing process is not normally invertible. A point in space captured with a camera has two known parameters, which are its projected 2D space coordinates and one unknown parameter, its distance from the camera. For that reason, additional information is required to solve the 3D from 2D reconstruction problem. One of the solutions of the stated problem is to use geometric information from multiple views, which is done using structure from motion method.

SfM ultimate output is sparse point cloud along with camera parameters which commonly includes rotation and translation matrices, focal length and lens distortion coefficients for each image. This algorithm can be considered as a skeleton for the MVS dense point cloud reconstructed. It is crucial to have a legitimate camera parameters, as failure to acquire a proper skeleton will result in poor accuracy and completeness of MVS dense point cloud reconstruction. In the context of this work, poor dense point cloud quality implies deficient as-planned model. Subsequently, this results in an inadequate progress report and in unwanted product for the industry. That is why particular attention is paid to the structure from motion algorithm in this work.

Snavely et al. (2006a) presented an Structure form Motion tool for 3D reconstruction of a scene based on large unstructured collections of photographs. This approach produced accurate reconstructions in scenarios with hundreds and thousands of randomly captured photographs. An example of some of the steps of the proposed method is shown in Figure 3.2. This concept provoked colossal interest in the further development of SfM among researchers (Özyeşil, Voroninski, Basri, and Singer (2017)).

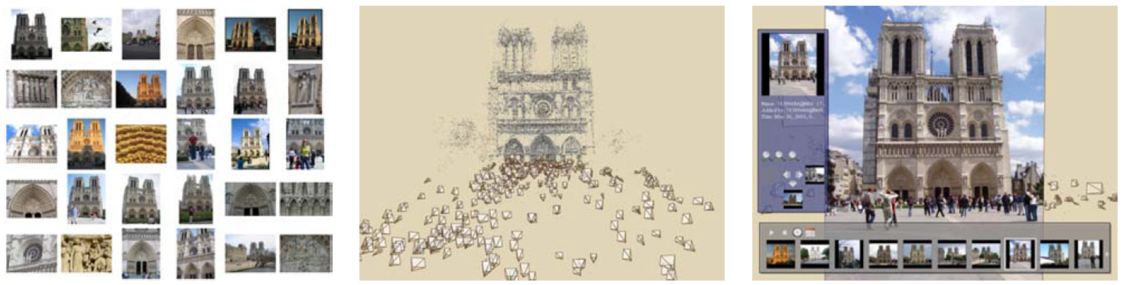


Figure 3.2: Collection of photographs, reconstructed point cloud and photo browser (Snavely et al. (2006a))

The novel approach embodies three major developments: image-based modeling, image-based rendering, and image browsing, retrieval, and annotation. Considering the focus of this thesis, image-based modeling is the object of interest in Snavely et al. (2006a), as it uses SfM (Hartley and Zisserman (2003)) for recovering camera parameters, pose estimation, and sparse point cloud reconstruction. A general pipeline flowchart for the incremental SfM is demonstrated in Figure 3.3. More information on SfM and each of its step can be found in Baeza-Yates, Ribeiro-Neto, et al. (1999), Hartley and Zisserman (2003), Nkanza (2005), Snavely et al. (2006a), Beder and Steffen (2006), Ko and Ho (2016), Schonberger and Frahm (2016), Özyeşil et al. (2017).

The first group of processes in the incremental SfM flowchart (Fig. 3.3) is correspondence search. This part is common for various computer vision applications such as SfM, image alignment, object recognition, motion tracking, indexing and

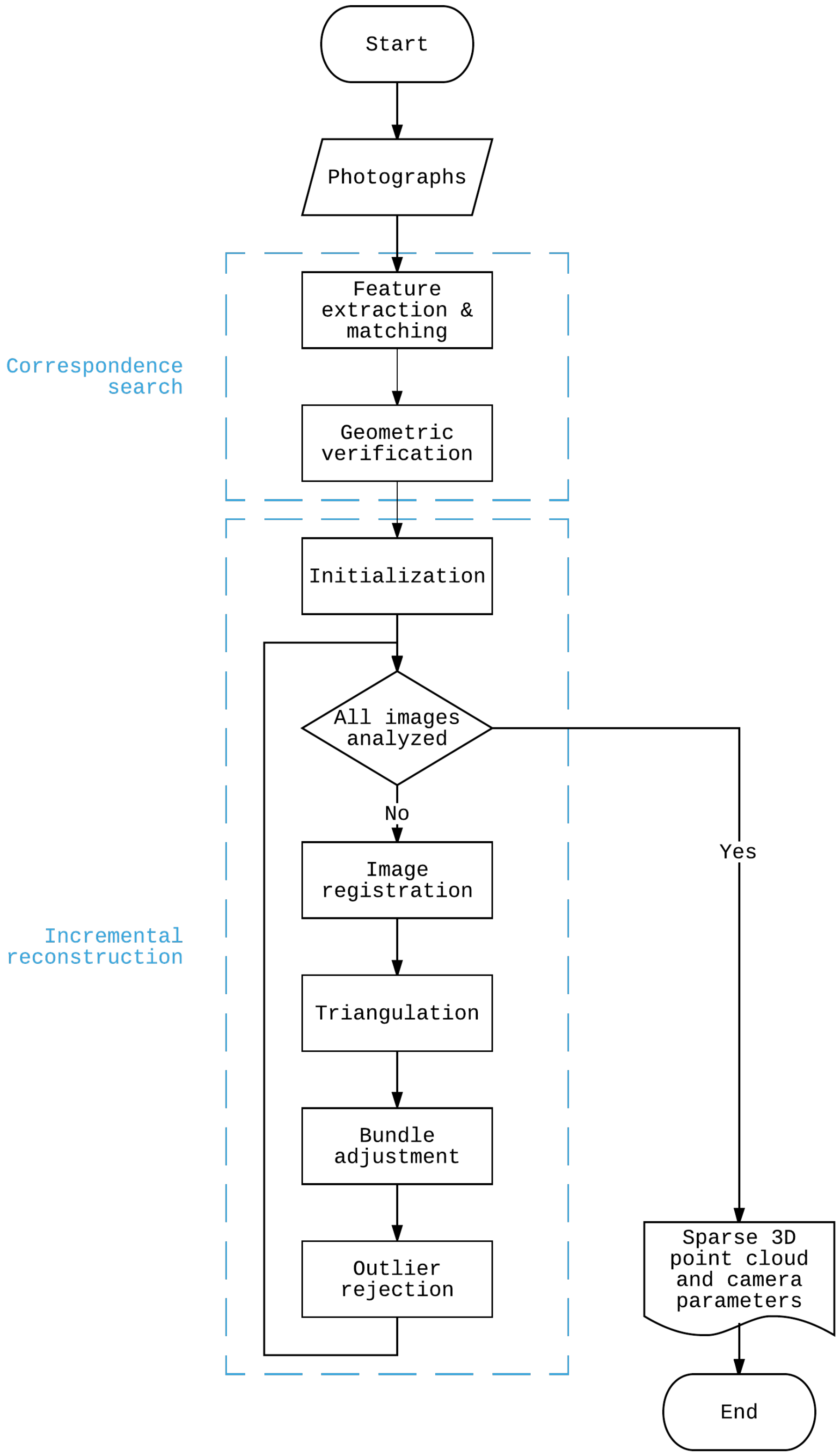


Figure 3.3: Incremental Structure from Motion flowchart

content-based retrieval and robot navigation. The second group of processes is incremental reconstruction, and it is specific to the SfM approach.

In the correspondence search step, the distinctive key points (or features) in each image are detected with the Scale Invariant Feature Transform (SIFT) algorithm (SIFT, Lowe (2004)). Several thousand points that are normally identified in a photograph. The detected features are invariant to image transformation. Next, the features are matched with the Random Sample Consensus algorithm (RANSAC, Fischler and Bolles (1981)) for the accuracy of the correspondence. At this stage matches are detected and the features without matches are discarded. A simple example of SIFT-RANSAC features detection and matching on a pair of images is demonstrated in Figure 3.4.

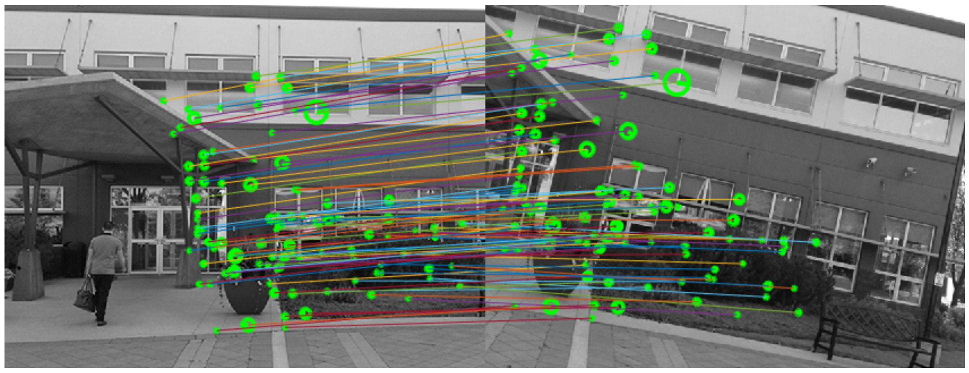


Figure 3.4: SIFT-RANSAC features detection and matching example

The features are matched based on appearance only, which is not enough for detecting actual point matches on different images. That is why geometric verification is required. Geometric verification reviews feature matches on potentially overlapping images. Hence, transformations that map features between all image pairs are sequentially computed based on projective geometry. If a transformation maps a tolerable number of features between an image pair, then the images are geometrically verified. In this case, the mapped features are considered to be inliers, and the features which are failed to map are outliers and they are dismissed for feature computation. The output of this stage is a set of geometrically verified image pairs with corresponding inlier matches.

A visual explanation of geometric verification on a simple one point scenario is shown in Figure 3.5. Two cameras are illustrated with their camera centers O_L and O_R, and left and right image views correspondingly. The point X is an arbitrary point in 3D space. The camera centers and the point X create a plane $O_L X O_R$. $X_L X$ and $X_R X$ are back-projected rays of a 3D point X to the left and right views respectively. Thus, if after transformation the point O_L maps to O_R then this this match is geometrically verified. Otherwise, if the transformed O_L point maps to any other place in space such as an $X_?$ point then this match is rejected.

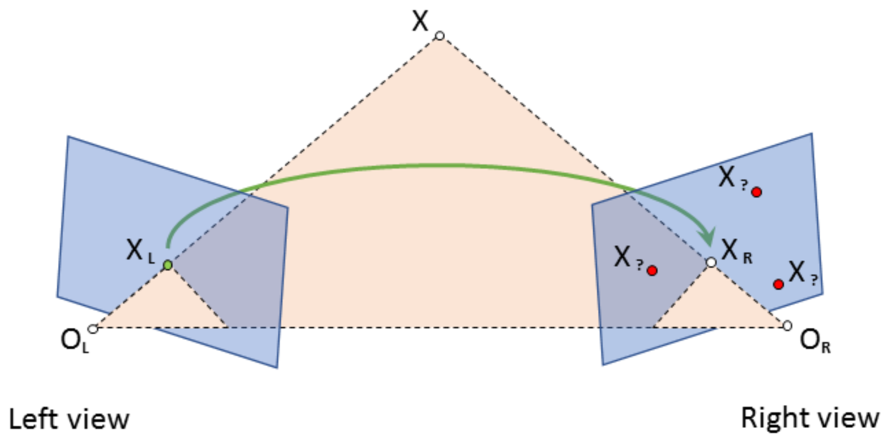


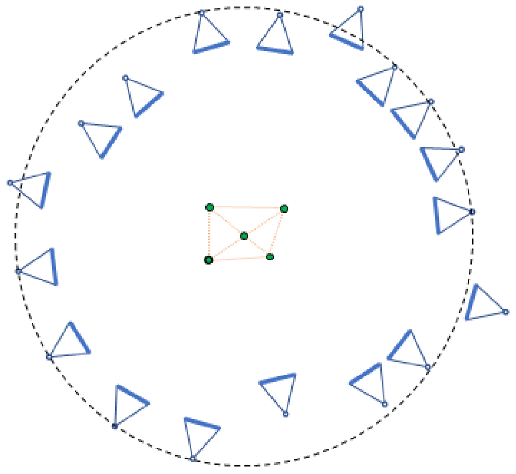
Figure 3.5: Point correspondence geometry

Then, the incremental reconstruction phase takes place. The reconstruction is initialized from a pair of images with vast number of inlier matches and many overlapping views. Choosing an appropriate initial image pair is vital for robustness, accuracy, and performance of the reconstruction. Failing to choose a proper pair of images might result in increasing the reconstruction time, decreasing point cloud density, or even failing of the overall reconstruction process.

Image registration spatially align images from different viewpoints. This process is required because photographs captured from different points are distorted with respect to each other. More specifically, optimal transformation for image alignment is estimated at this stage.

Having estimated the image transformation, triangulation determines the location of 3D points by forming triangles to their projections on 2D multiple photographs. Triangulation creates a triangle between camera centers or the point projections and a 3D point. Visual example of triangulation is the plane $O_L X O_R$ in Figure 3.5. New scene points are triangulated and added to the point cloud if at least one more image covers these new points from a different angle. In ideal scenario, triangulated 3D point locations should lie at the intersection of the back-projected rays as it shown in Figure 3.5. However, practically this is not generally the case. The back-projected rays do not exactly intersect because, for example, geometric noise from lens distortion or point detection error, which leads to inaccuracy in 3D points coordinates. Hence, only 3D points which optimally fit their projection points should be found. Error optimization algorithms are employed for error minimization of the measured and predicted point positions in all views where a point is visible. Triangulation is an essential part of the overall SfM pipeline, as it raises the stability of a point cloud through repetition and allows registration of new photographs by providing more corresponding points.

The next stage is bundle adjustment, it refines a visual reconstruction to jointly produce optimal 3D point cloud and viewing parameters. To be more specific, bundle adjustment is used for simultaneous refining of all the 3D point coordinates and all the camera parameters by minimization of an appropriate reprojection error cost function. In other words, this mathematical optimization minimizes the discrepancy between actual image measurements and their predictive models. A visual example of the bundle adjustment effect is illustrated in Figure 3.6. The left part of the figure represents a reconstruction stage before bundle adjustment, and the right part of the figure is the scene after bundle adjustment. The camera centers with their photographs are lying approximately on the circle line and a point cloud is inside the circle. The difference between the left and right parts of Figure 3.6 indicates that the bundle adjustment algorithm aligns the point cloud along with the camera views. This stage heavily depends on a proper initialization. A failure to appropriately choose the initial image pair is likely to result in the bundle adjustment cost function convergence to a local minimum. Consequently, this leads to a decrease in accuracy, an increase in computation time or even to bundle adjustment failure.



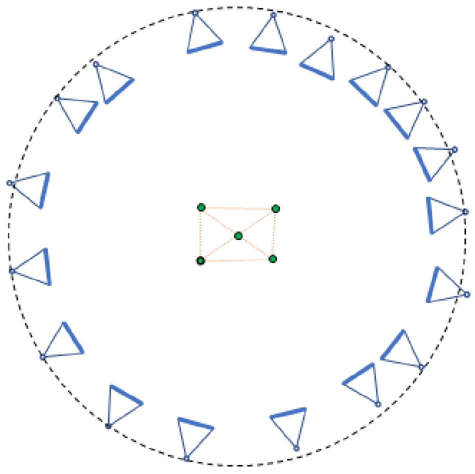


Figure 3.6: Bundle adjustment effect

The removing statistical outliers ends an image pair processing loop. Next, the remaining views are added incrementally into a merged representation. After all photographs have passed the incremental reconstruction loop, the structure form motion algorithm is terminated. The ultimate SfM output is a sparse point cloud. Additionally, SfM provides camera parameters data for further point cloud processing. Overall illustration of the incremental SfM pipeline is shown in Figure 3.7.

The maximally efficient algorithm performance is achieved by adhering to the following recommendations:

- images should have high visual overlap;
- each object of interest should be seen on multiple images;
- images should be captured from different viewpoints;
- images should be captured at identical illumination conditions. This advice includes avoiding capturing photographs against the sun as well as capturing

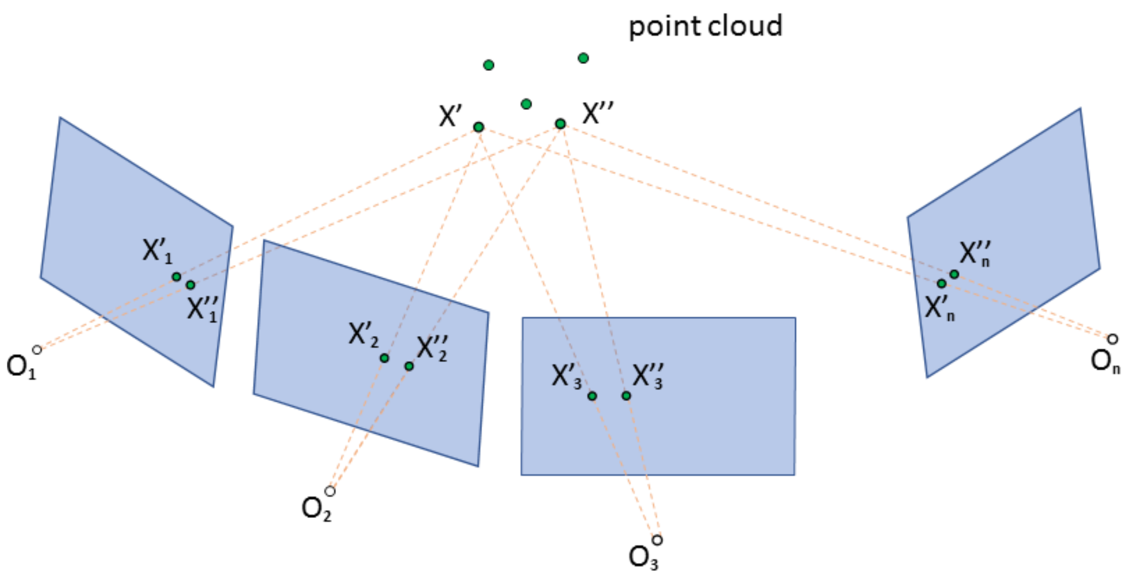


Figure 3.7: Illustration of SfM reconstruction

shadows;

- the objects of interest should be properly textured. Otherwise, if the objects are of similar texture, transparent, or shiny, it will reduce the number of identified unique points. Consequently, these circumstances will decrease the number of correspondences and the number of points in a point cloud.

To demonstrate the SfM algorithm, an experimental set of photographs of a soda can was taken. This data set included 90 photographs with a pixel resolution of 3264 x 2448, and it was taken by a simple iPhone 5s camera. The photographs were then passed to the SFM algorithm for sparse point cloud reconstruction and camera parameters estimation. Ultimately, a sparse point cloud of 8,562 points was reconstructed. An example photograph from the photo set is shown in Figure 3.8. Figure 3.9 indicates a sparse point cloud of the soda can, which is the ultimate output of the SfM algorithm.

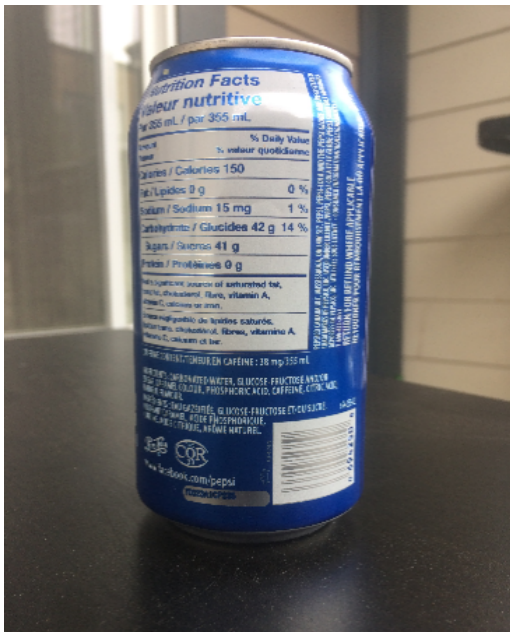


Figure 3.8: Soda can photo from the soda can experimental dataset

Figure 3.9: Sparse point cloud of the soda can

3.2.2 Multi-View Stereo

Multi-view stereo (MVS) is a group of methods for reconstruction of an entire 3D object model from an unstructured collection of photographs using images stereo correspondence (Furukawa, Hernández, et al. (2015); Seitz, Curless, Diebel, Scharstein, and Szeliski (2006)). MVS takes the output of SfM, which is a sparse point cloud and camera parameters, along with the photographs and outputs a dense point cloud.

The MVS part of the overall reconstruction process is as important as the SfM part. If we metaphorically represent structure from motion output as a skeleton, than multi-view stereo output is a body. Having the best superior sparse point cloud and camera parameters does not mean having an accurate and complete ultimate dense point cloud if there is no efficient MVS algorithm available. That is why an MVS technique is significant for having adequate final results.

There are over 50 different approaches to Multi-view stereo available today (Bailer, Finckh, and Lensch (2012)). One of this techniques is based on multiple depth maps. This approach demonstrated superior reconstruction results on its own as well as in comparison to different cutting edge techniques (Bailer et al. (2012); Schönberger, Zheng, Frahm, and Pollefeys (2016); Schöps et al. (n.d.)). A general multiple depth maps MVS algorithm is described in this section. An example of a depth maps MVS algorithm pipeline is illustrated in Figure 3.10.

Having SfM output, an image for further depth map computation is selected. For each photograph a proper corresponding image is chosen. This image is chosen based on the list of specific requirements, such as similarity to the first image scale, large enough view angle difference, and a big overlap of the scene of view. Having an image pair, the depth map is computed. The depth map is an image that contains information related to the distance from the objects on a photograph to the corresponding viewpoint.

Depth map calculation is possible because an image pair provides sufficient information to estimate the distances of interest. This is demonstrated in Figure 3.11. Two points with distinctive depth in space X and Y have the same projection X_L and Y_L onto the left view image. Simultaneously, X_R and Y_R have different positions, which is to say that the second camera resolves the ambiguity, enabling measurement of depth via triangulation.

Having O_LXO_R and X_LXX_R similar triangles, the distance values can be computed. Figure 3.12 shows the computation geometry. O_L and O_R are left and right views camera centers correspondingly, B is the baseline distance between the camera centers, X (x, z) is the real world point and X_L (x_L, f) and X_R (x_R, f) are its projections to the corresponding views, f is the focal length, and z is the distance from X to the baseline. Finally, there are the algebraic equations of the depth map computation:

$$x_L = f\frac{x}{z} \tag{3.1}$$

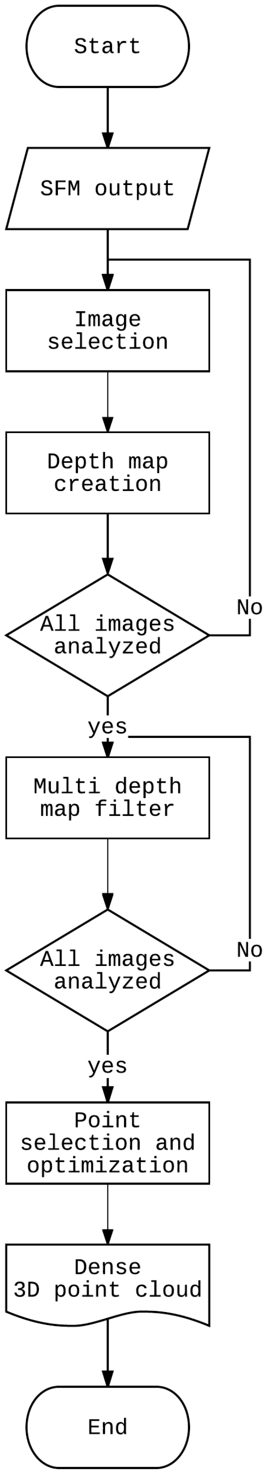


Figure 3.10: Multi-View Stereo flowchart

$$x_R = f\frac{B - x}{z} \tag{3.2}$$

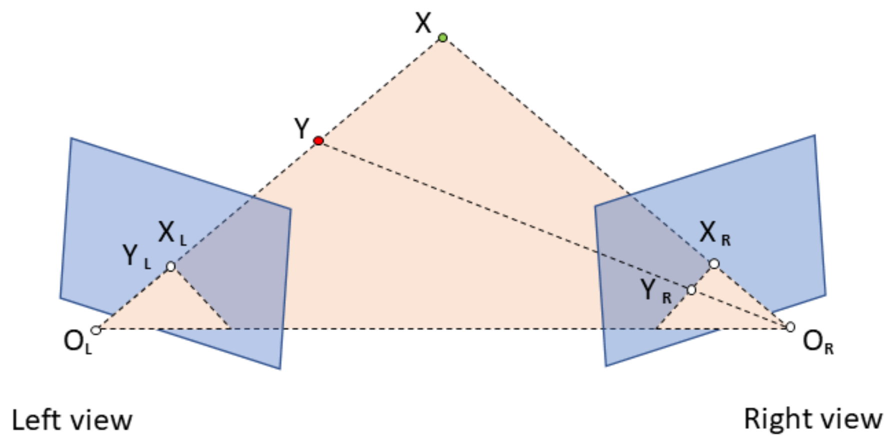


Figure 3.11: Triangulation from a stereo image pair

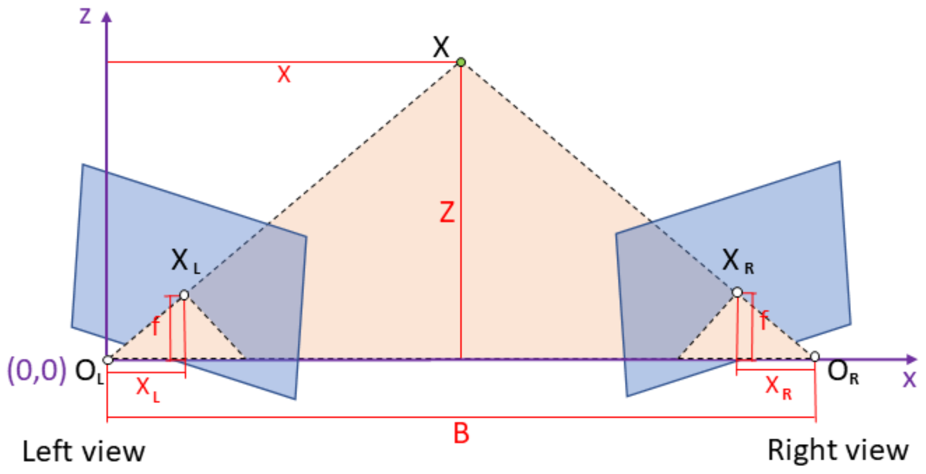


Figure 3.12: Depth calculation

$$x_R + x_L = f\frac{B}{z} \tag{3.3}$$

$$z = f\frac{B}{x_R + x_L} \tag{3.4}$$

The obtained value of z in the Equation 3.4 is the distance from the real world point X to the baseline (or depth). The ultimate depth map consists of the computed depth for each pixel. $x_R + x_L$ is called disparity (d) and implies a point displacement in different images. Taking disparity into account, there is the final depth equation:

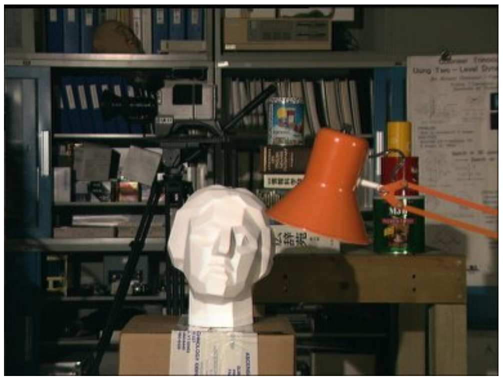


Figure 3.13: Left image, Tsukuba Scharstein and Szeliski (2002)

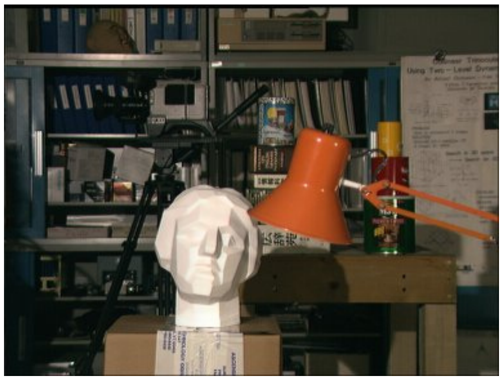


Figure 3.14: Right image, Tsukuba Scharstein and Szeliski (2002)

$$z = f\frac{B}{d} \tag{3.5}$$

Firstly, the coarse depth map is built based on unique points in the SfM output. Next, the depth computation is propagated onto adjacent points. When the depth is computed, it is stored as an image. The example of a stereo pair and a depth map image is shown in Figures 3.13, 3.14, 3.15, 3.16. The first two images (Figures 3.13 and 3.14) are the left and right views, correspondingly. Figure 3.15 is an image with left and right views overplayed; the left image is represented with red channel, and the right one is represented with blue channel. Figure 3.16 is the depth map. In this image, the closer objects look brighter and the farther objects look darker.

The next stage of the MVS algorithm is filtering of the depth maps from outliers. Filtering outliers implies rejecting the points with erroneous depth values and the points which do not form a surface. This is done by segmenting the images by depth discontinuities. The points form a depth segment if they have similar depth and their segment area does not overlap with other segment areas. Then, all the regions with a small number of points, less than 15 for example, are deleted. The points discrepant with all the segment depth values are discarded. Once the depth maps are refined, they are compared against each other and the points in the depth maps that have inconsistent or corrupted depth matches are rejected. Figure 3.17

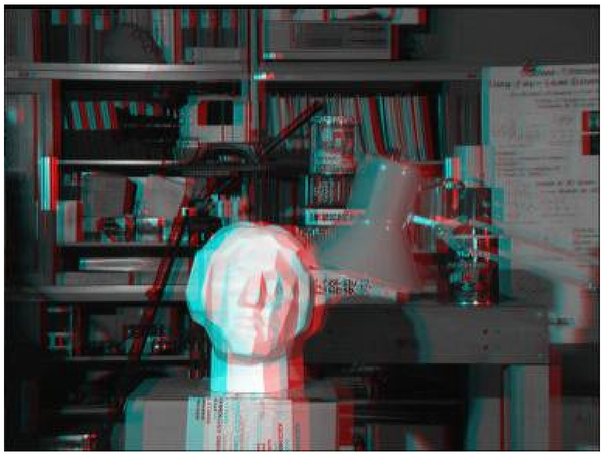


Figure 3.15: Stereoscopic image with the left image coded in red channel and the right image coded in blue channel

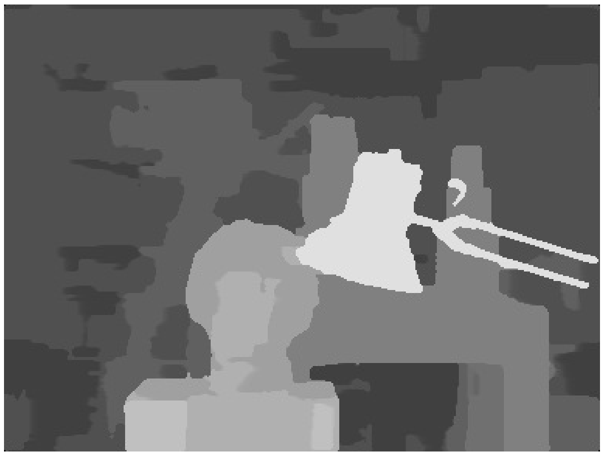


Figure 3.16: Grayscale depth map, Tsukuba Scharstein and Szeliski (2002)

illustrates some resulting cases of depth maps comparison to each other. Depth map for the point X is consistent, whereas the depth maps for the points Y and Z are inconsistent.

The optimal depth values are transformed into 3D points. The non-optimal points might be used for improving the optimal points location. Ultimately, the final output of the MVS algorithm dense point cloud is formed.

The schematic effect of the MVS algorithm on a sparse point cloud is illustrated in Figure 3.18. There are three images of one structure from different angles in the top of the figure. Also, there is a fragment consisting of 3 points on a sparse point

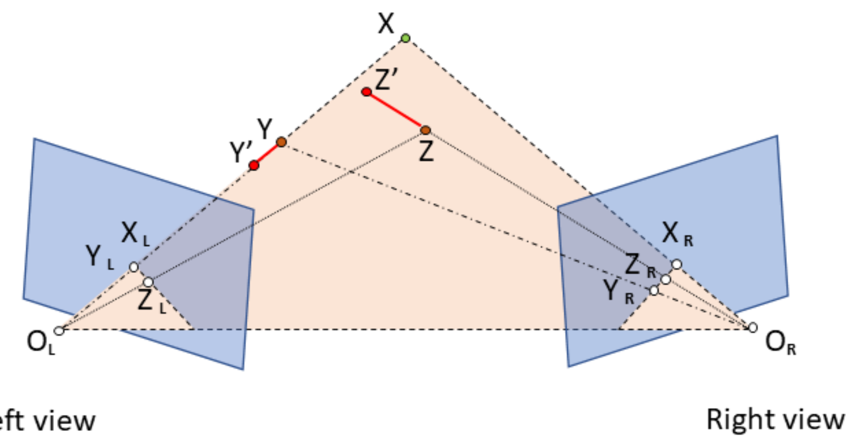


Figure 3.17: Some cases of multiple depth maps comparison

cloud which is shown in the left bottom rectangle; as well, these points projections on the input images are demonstrated. Eventually, the right bottom rectangle illustrates the densified sparse point cloud fragment obtained by employing the MVS algorithm.

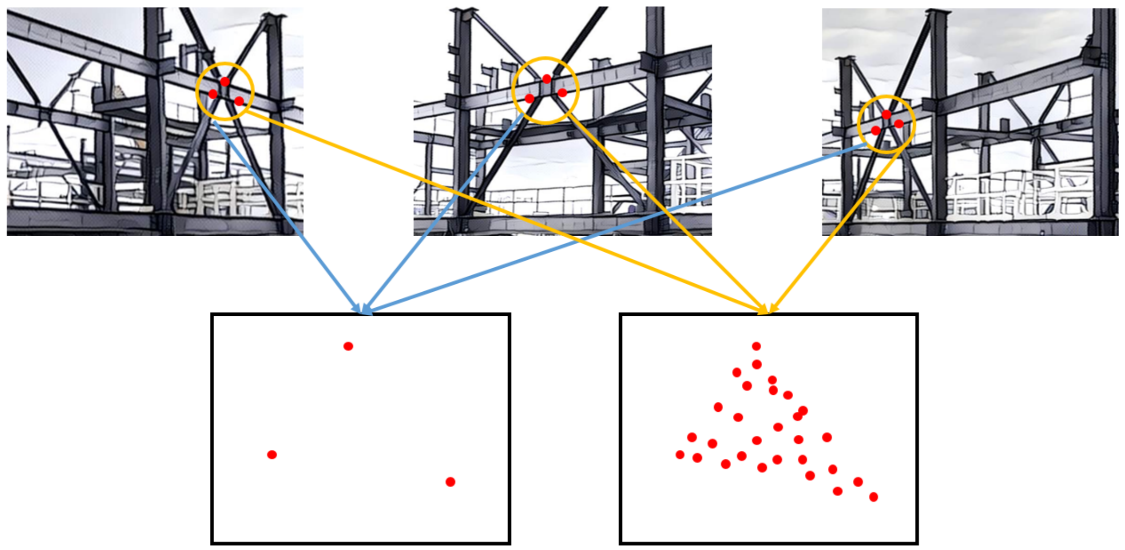


Figure 3.18: MVS densification effect on the fragment of a sparse point cloud

To demonstrate the effect of MVS on an actual point cloud, the already mentioned dataset of the soda can photos and SFM output are used (see subsection 3.2.1). A dense point cloud of 145,150 points is reconstructed, and it is illustrated in Figure 3.19. For contrast, the sparse point cloud of this soda is shown close to the dense point cloud figure.

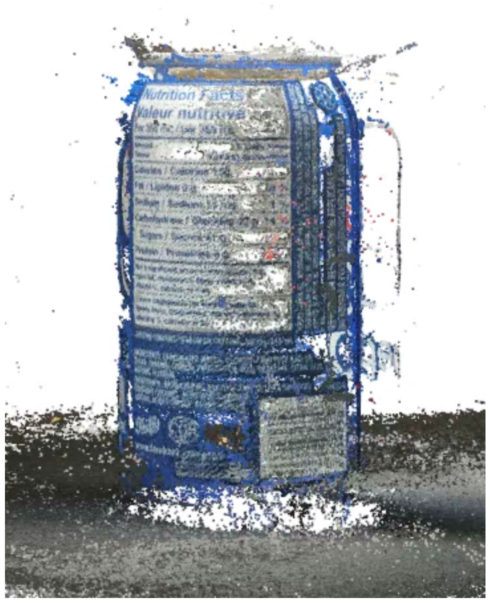
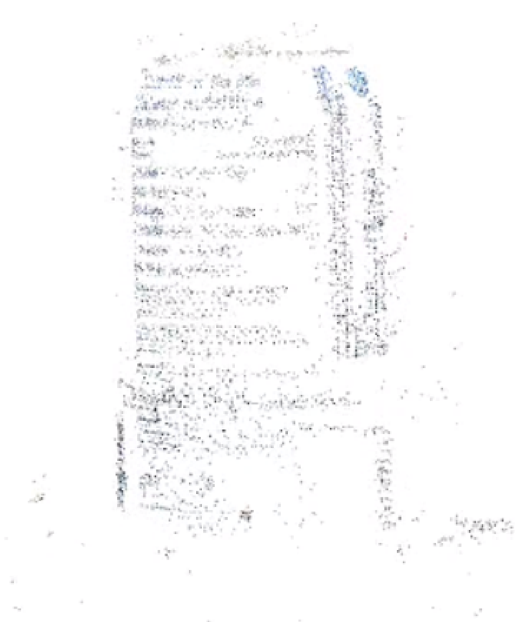


Figure 3.19: Dense point cloud of the soda can



Repeated Figure 3.9: Sparse point cloud of the soda can

3.3 Point Cloud Registration

At the current progress tracking stage, there are a reconstructed as-built point cloud model and a corresponding as-planned 3D CAD model. The next logical step is comparing these models.

There is a list of common difficulties, that should be dealt with for proper models comparison:

- the models are in different format (a point cloud and a 3D model);
- the models are in distinct coordinate systems;
- the models are of different scale.

Considering the stated problems, the initial step is transforming an as-planned model into the same format. Two options are considered for proceeding with this task: converting the as-built model to the CAD model or converting the as-planned model to point cloud. The first option is quite challenging because the points on reconstructed point cloud surfaces are scattered with some tolerance because of the nature of the algorithm as well as these points always have some amount of noise. There a separate research is devoted to this topic. The second option which is converting a CAD model to a point cloud is straightforward. In this scenario, a CAD model is sampled to a point cloud. Thus, considering all the aspects of models format conversion this research work proposes using point cloud as a primary models format.

Having two models in the same format, the point clouds can be aligned and scaled to be the same size. The required transformation problem is commonly referred to as similarity transformation. Similarity transform addresses all the transformations of interest of this thesis work, including rigid transformation (translation and rotation) and uniform scaling.

Rigid transformation is commonly aligned with the original Iterative Closest Point method (ICP, Besl and McKay (1992)). However, similarity transform requires more enhanced ICP modifications, as, for example, described in Pomerleau, Colas, and Siegwart (2015). These extensive methods have been widely used in object recognition, different kinds of inspections, medical imaging and mobile robotics.

One more comparatively new approach for similarity and rigid transformations is Coherent Point Drift (CPD, Myronenko and Song (2010)). This method has been successfully used for medical applications and for geographic change detection (Gadomski (2016)). At the same time, CPD has not been used in the construction industry to the best of the author's knowledge.

Both ICP and CPD approaches are promising for the construction progress tracking pipeline for both computer vision and laser scanners data registration; hence, they are presented in this section.

3.3.1 Iterative Closest Point

Iterative Closest Point (ICP) is an iterative registration algorithm. In computer vision, ICP associates data sets into a common coordinate system by minimizing the alignment error. Herein, the general ICP framework is described according to Besl and McKay (1992); Pomerleau et al. (2015); Pomerleau, Colas, Siegwart, and Magnenat (2013); Pomerleau, Magnenat, Colas, Liu, and Siegwart (2011).

The goal of ICP is to align two point clouds, where the one called reading is to be aligned with the one called reference in the reference coordinate system. This is the same as finding a proper transformation of the reading to the reference. The ICP underlying mathematics is presented in below in accordance with Pomerleau et al. (2015). The optimization function for these point clouds looks as follow:

$$ {}^{A}_{B}\hat{\mathbf{T}} = \underset{T}{\operatorname{argmin}} \left(error\left(T({}^{A}P), {}^{B}\Theta \right) \right) \tag{3.6} $$

where ${}^{A}P$ is the shape of reading point cloud in the coordinate system A, ${}^{B}\Theta$ is the shape of reference point cloud in the coordinate system B, and $T({}^{A}P)$ is the geometrical transformation of the reading point cloud. The goal of this loss function is to find the best estimate of the transformation ${}^{A}_{B}\hat{\mathbf{T}}$ of the P from its original coordinate system A to the reference one B by minimizing the transformation error.

More specifically, the loss function is computed based on pairs of points from the reading and the reference point clouds. Commonly, the points are chosen based on their distance, such as the closest two points from different point clouds forming a pair. This approach does not usually guarantee the best optimal results. Therefore, the transformation results might be improved by using extra information from descriptors and/or features as well as by applying filters.

There are two main characteristics in a point cloud potentially influencing the transformation results. The first one is features. A feature represents point geometric information. The second is descriptors. A descriptor describes point non-spatial information, such as color for example.

Most ICP approaches provide data filters prior to starting algorithm iterations. There are generally two strategies for applying such filters. The first one is reducing computational complexity. This is usually achieved by removing points with weak influence on the optimization function. The second strategy is improving registration by making the optimization function more precise through adding extra information to points. For instance, the points might be complemented with normal or local shape properties.

Knowing the features, descriptors and having different filters, the L1 loss functions is developed as below.

$$error(P, Q) = \sum_{p,q \in M} d(p, q) \tag{3.7}$$

where, $M = match(P, Q) = (p, q) : p \in P, q \in Q$ is the set of matches between the reading and the reference point clouds, and $d(p, q)$ is the distance between points p and q.

In order to enhance this function the weights might be assigned to each match. This potentially decreases the outliers and the noise influence on the final results.

$$error(P, Q) = \sum_{p,q \in M} w(p, q) d(p, q) \tag{3.8}$$

where $W = w(p, q) : \forall (p, q) \in M$.

Having this condition set up, in an ideal case scenario the bast estimated transformation is ${}^{A}_{B}\mathbf{T}$. However, this is not the case in the real world. This is where employing ICP comes in useful. The concept of ICP is minimizing the error for all points, that is why each ICP iteration leads to a better transformation estimate even having bad matches. Thus, ICP consists of a transformation series of the reading point cloud. New matches are iteratively computed with each transformation. This loop is terminated when the error threshold is reached or when the maximum specified number of transformations is reached. This optimization process looks like this:

$$^{i+1}_{\ i}\mathbf{T} \leftarrow \underset{T}{\operatorname{argmin}} \Big(error\big(T(^{i}P), \Theta\big)\Big) \tag{3.9}$$

Finally, the ultimate transformation estimate is a combination of all the iteration transformations:

$$^{A}_{B}\hat{\mathbf{T}} = (\underset{i}{\bigcirc}\, ^{\ \ i}_{i-1}\mathbf{T}) \circ T_{init} \tag{3.10}$$

where $\bigcirc_i\, ^{\ \ i}_{i-1}\mathbf{T} = ... \circ {}^{3}_{2}\mathbf{T} \circ {}^{2}_{1}\mathbf{T}$ is the composition of transformations and T_{init} is an initial transformation.

The algorithm is summarized in a flow chart in Figure 3.20.

3.3.2 Coherent Point Drift

Coherent point drift is an iterative probabilistic method for point cloud registration (Myronenko and Song (2010)). This is an alternative to the ICP approach. The point cloud alignment in this method is considered as a probability estimation problem. The Gaussian mixture model (GMM) centroids of the reading point cloud (as-built model) are fit to the reference point cloud (as-planned model) by maximizing the likelihood. During each iteration the probabilities for all GMMs are computed and the transformation is estimated, which allows the points to move coherently as a group to maintain the point cloud structure. Considering that both point clouds have significantly more good points than outliers, CPD is more invariant to outliers than ICP. The difference of the methods underlying ideas is presented on a simple example in Figures 3.22 and 3.21.

CPD underlying mathematics is presented below according to the original paper by Myronenko and Song (2010).

The principal CPD probability density function is shown in the Equation 3.11.

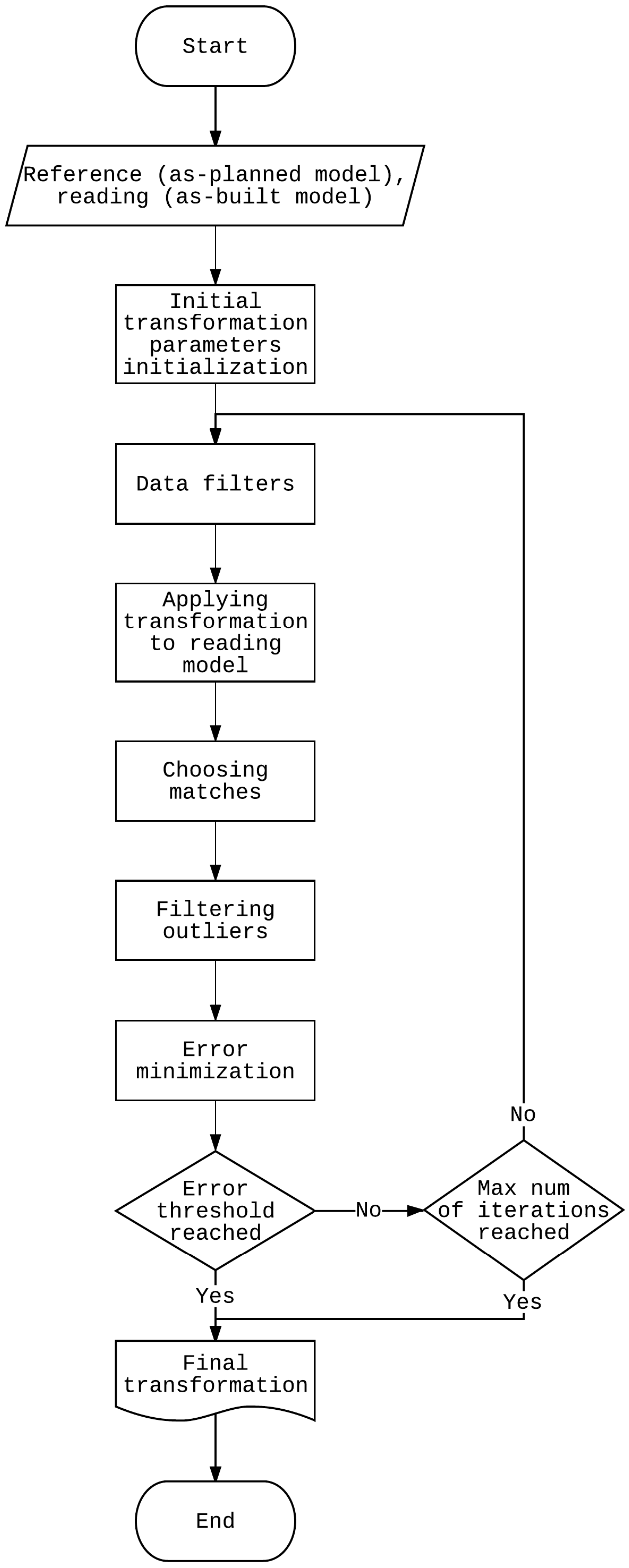

Start
Reference (as-planned model), reading (as-built model)
Initial transformation parameters initialization
Data filters
Applying transformation to reading model
Choosing matches
Filtering outliers
Error minimization
Error threshold reached
No
Max num of iterations reached
No
Yes
Yes
Final transformation
End

Figure 3.20: ICP flowchart

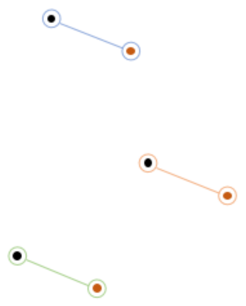


Figure 3.21: Closest points simple concept illustration

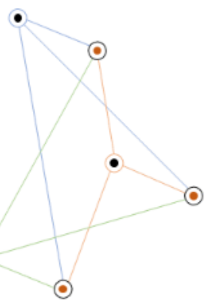


Figure 3.22: Gaussian mixture model simple concept illustration

$$p(x) = \omega\frac{1}{N} + (1-\omega)\sum_{m=1}^{M}\frac{1}{M}\frac{1}{(2\pi\sigma^2)^{\frac{D}{2}}}exp\left(\frac{||x-y_m||^2}{2\sigma^2}\right) \tag{3.11}$$

where

- ω is the weight given to points to account for outliers and noise, such as $0 \leq \omega \leq 1$. The weight is generally distributed uniformly;
- D is the dimensionality of X and Y ($D = 3$ in this particular case with point clouds);
- $X_{N\times D} = (x_1, x_2, ..., x_N)^T$ is the reference (as-planned model) point cloud;
- $Y_{M\times D} = (y_1, y_2, ..., y_M)^T$ is the reading (as-built model) point cloud or the GMM centroids;
- σ^2 is the bandwidth of the GMM.

Next, GMM centroids are characterized by Θ and σ^2. The number and types of parameters of Θ depends on the type of performing transformation (rigid or affine transformation, for example). The objective function Q is defined in Equation 3.12.

$$Q(\Theta,\sigma^2) = \frac{1}{2\sigma^2}\sum_{n=1}^{N}\sum_{m=1}^{M}P^{old}(m|x_n)||x_n - T(y_m,\Theta)||^2 + \frac{N_P D}{2}\log\sigma^2 \tag{3.12}$$

where

$$N_P = \sum_{n=1}^{N} \sum_{m=1}^{M} P^{old} P^{old}(m|x_n) \leq N \tag{3.13}$$

$$P^{old}(m|x_n) = \frac{exp\left(-\frac{1}{2} \left|\left| \frac{x_n - T(y_m, \Theta^{old})}{\sigma^{old}} \right|\right|^2 \right)}{\sum_{k=1}^{M} exp\left(-\frac{1}{2} \left|\left| \frac{x_n - T(y_k, \Theta^{old})}{\sigma^{old}} \right|\right|^2 \right) + c} \tag{3.14}$$

and

$T(y, \Theta)$ - is transformation of the point y with the Θ parameter

$$c = (2\pi\sigma^2)^{\frac{D}{2}} \frac{\omega}{1-\omega} \frac{M}{N} \tag{3.15}$$

Subsequent iterations minimize Q. The iterations are terminated when the error threshold is reached or when the maximum specified number of transformations is reached.

The algorithm is summarized in a flow chart in Figure 3.23.

Nevertheless, although the CPD algorithm looks simpler than ICP, it is really effective. Likewise, CPD might be complemented with special data and outliers filters.

3.4 Progress Identification

The next and final stage after having as-built and as-planned models aligned and in the same scale is progress detection. For this stage, the already built construction object parts should be identified. Based on the available data, one possible effective and computationally inexpensive method is the Hausdorff distance. According to the literature review, this method has not been used for progress tracking of construction objects, although the Hausdorff distance is commonly used in computer vision for point cloud comparison. Hence, the progress identification stage is implemented with the Hausdorff distance.

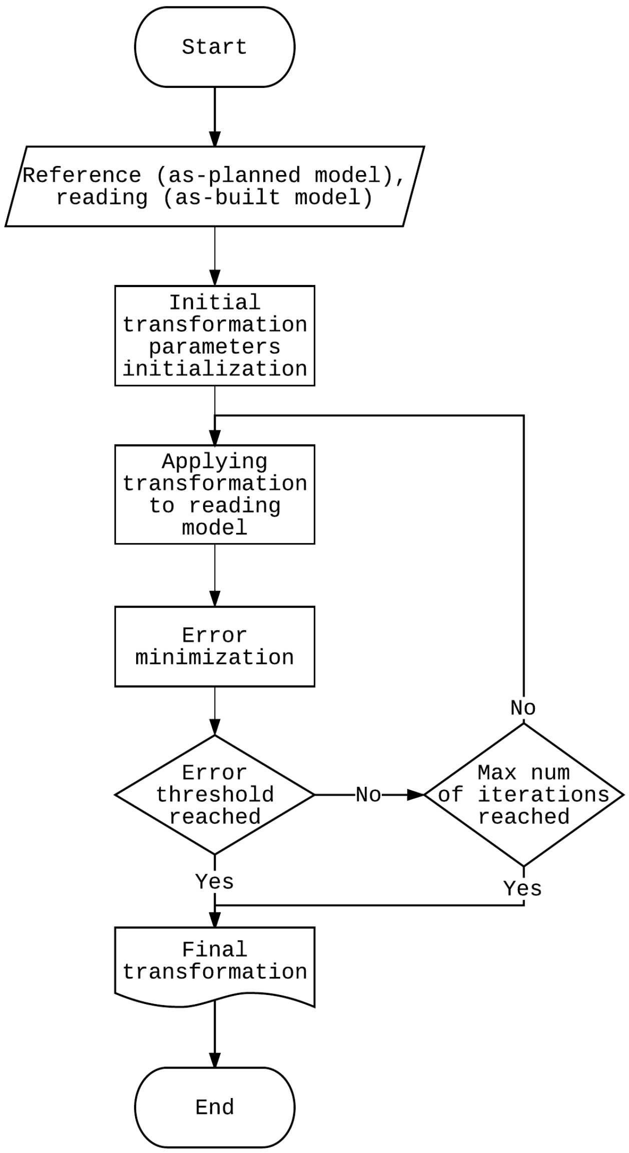

Start
Reference (as-planned model), reading (as-built model)
Initial transformation parameters initialization
Applying transformation to reading model
Error minimization
Error threshold reached
No
Max num of iterations reached
No
Yes
Yes
Final transformation
End

Figure 3.23: CPD flowchart

3.4.1 Hausdorff Distance

The Hausdorff distance is "the maximum of the distances from a point in any of the sets to the nearest point in the other set" (Rote (1991)). Mathematically, given two point sets A and B, the Hausdorff distance is a maximum function (Rucklidge (1996)), defined as:

$$h(A,B) = \max_{a \in A} \min_{b \in B} ||a - b|| \tag{3.16}$$

Thus, each point of A is associated with its closest surrounding point from B, and the most distant of these links is $h(A, B)$. This implies, that each point from A is located within the $h(A, B)$ distance from B (or its nearest neighbor form B).

The general concept of the Hausdorff distance is pretty obvious from the illustration in Figure 3.24. The picture shows three main stages of the computation (from left to right):

- measuring distances from each point of one set of points to each point of another set of points;
- selecting the closest points neighbors;
- choosing the maximum distance among ones selected in the previous step.

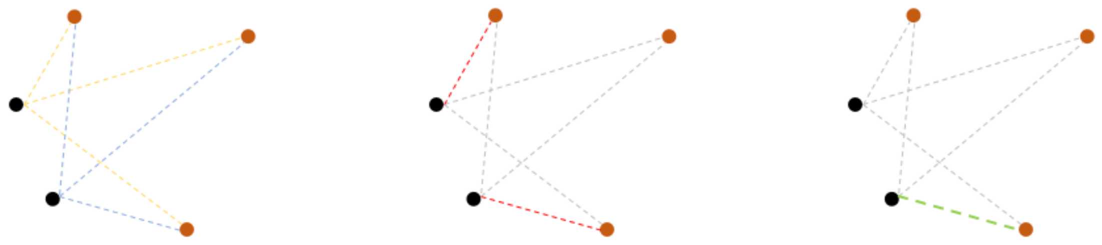


Figure 3.24: Step by step Hausdorff distance computation stages

Thus, the next step after aligning the as-built and the as-planned models is calculating the Hausdorff distance between them. When the distance is calculated the results are visualized in the form of a point cloud with meaningful colors assigned to each point. Figures 3.25 and 3.26 are examples of two 3D models comparison.

Figure 3.25 demonstrates two 3D models and the their Hausdorff distance model visualization. Figure 3.26 shows some details of the Hausdorff distance model visualization. Noticeably, the Hausdorff distance is visualized with a red-green-blue map, thus red is the minimum, green is the medium distance, and blue is the maximum distance.

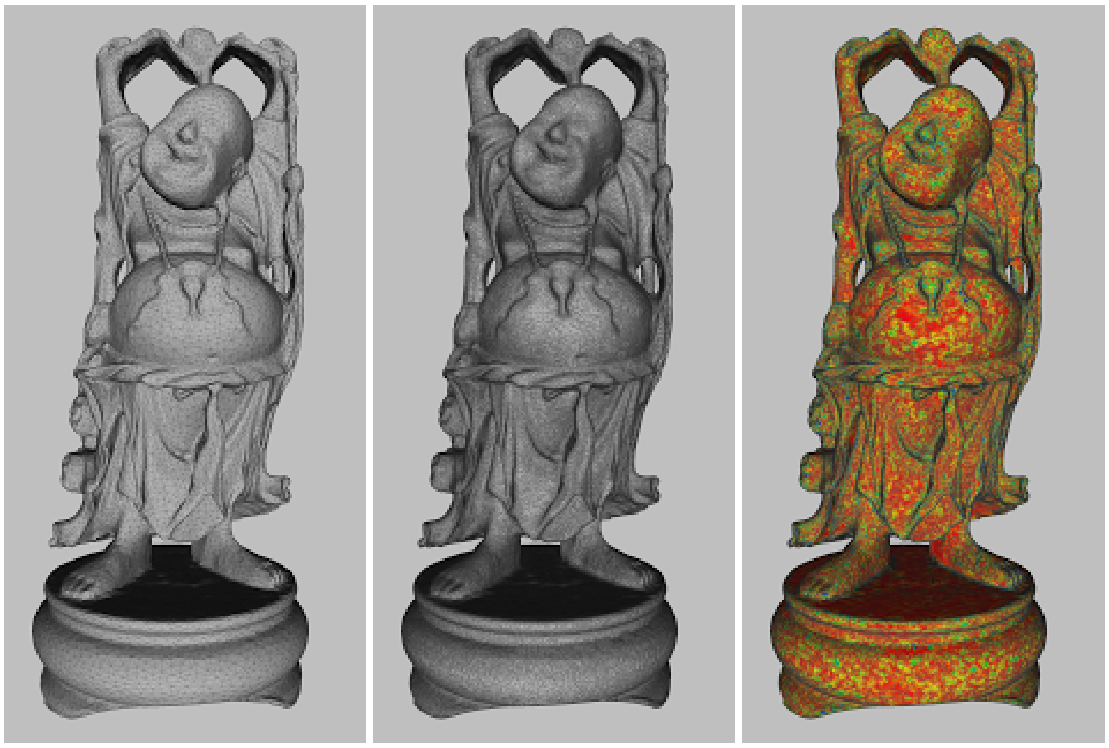


Figure 3.25: Hausdorff distance models (Cignoni (2010))

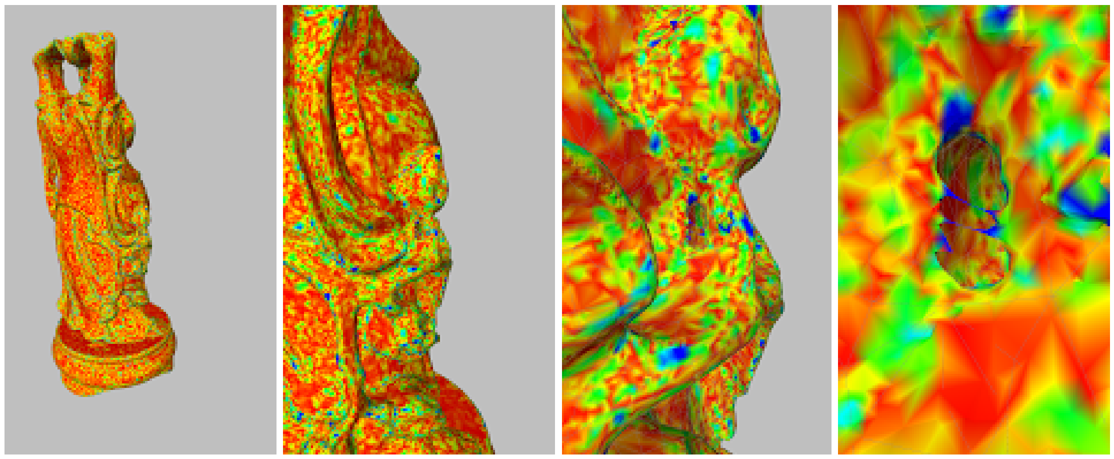


Figure 3.26: Fragments of hausdorff distance model (Cignoni (2010))

Chapter 4

Experimental Study

4.1 Introduction

The object of interest of the case of study is industrial modular construction. The industrial modules are of complex geometry with various tiny elements, which makes them challenging to reconstruct. More specifically, the proposed computer vision reconstruction core consists of an SFM-MVS algorithms pair, where the first algorithm tends to identify distinctive points and the second one densifies the area around unique points. Hence, this method performs exceptionally well for solid surfaces such as civil buildings or groundwork but not necessarily for sophisticated spatial structures. Thus, if the proposed computer vision progress tracking technique demonstrates high performance for modular construction then it is most likely to perform well for other construction sub divisions such as civil, infrastructure and industrial construction. To find the optimal project tracking approach, a case study was performed in partnership with the Ledcor construction company focusing on modular construction. The object of the case study of interest is one module of a modular gas plant construction project in Canada.

This chapter follows up the theoretical concepts introduced in Chapter 3 with practical applications.

4.2 Input Data

Data acquisition was performed on an actual modular construction yard. The photo and video data of a module construction was taken with a rudimentary camera of an iPhone 5S. The photos resolution is 8 MP (3264 x 2448 pixels). The video is captured with a resolution of 2 MP (HD or 1920 x 1080 pixels) with a frequency of up to 30 frames per second. The examples of some acquired data are presented in Figures 4.1 - 4.6.

Some challenges of taking photographs on an actual construction site are as follows:

- a person who is taking pictures should be wearing personal protective equipment;
- a person who is taking pictures should have gone through safety trainings;
- a construction site is usually congested that's why it is sometimes difficult or unsafe to take pictures from all desirable angles;
- there are no workers allowed to be in the photographs because of possible privacy issues; thus, all the photos should be taken outside of regular working hours or during breaks;
- some images around a module data set are usually taken against the sun which might potentially affect 3D reconstruction quality.

4.3 Point Cloud Reconstruction

Point cloud registration is a core of construction project monitoring approach. The quality of reconstructed point clouds significantly affects progress control precision. Simultaneously, the timing of a point cloud reconstruction might potentially be a major progress monitoring limitation. Reconstruction time of more than one day is

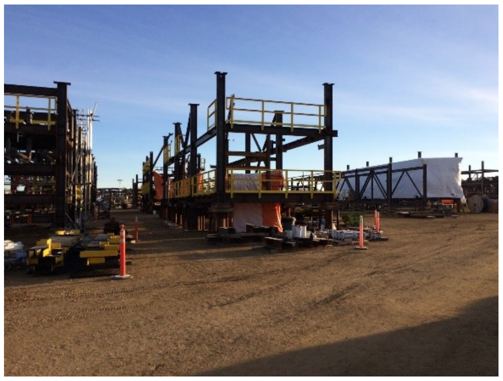


Figure 4.1: Construction module photograph example 1

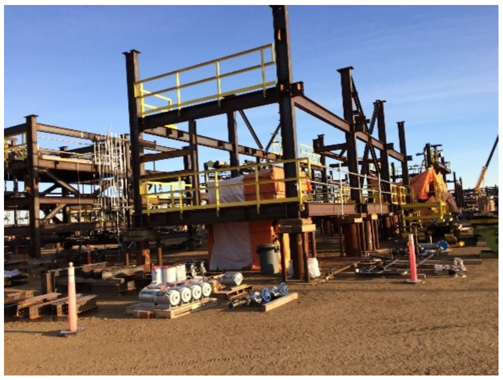


Figure 4.2: Construction module photograph example 2

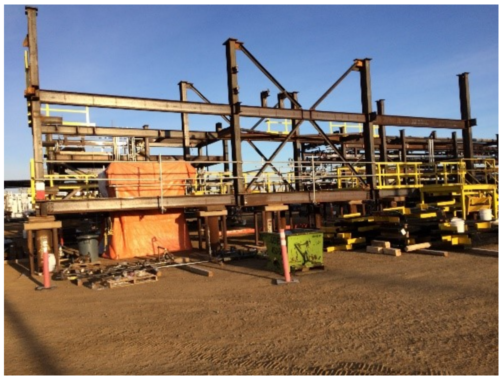


Figure 4.3: Construction module photograph example 3

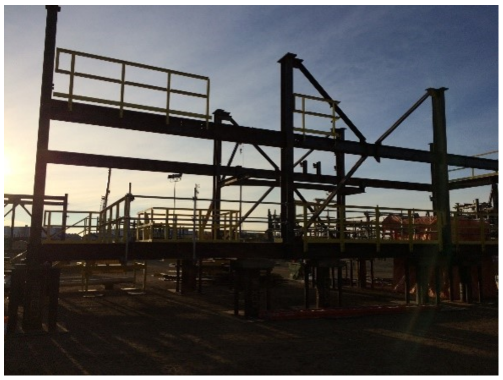


Figure 4.4: Construction module photograph example 4

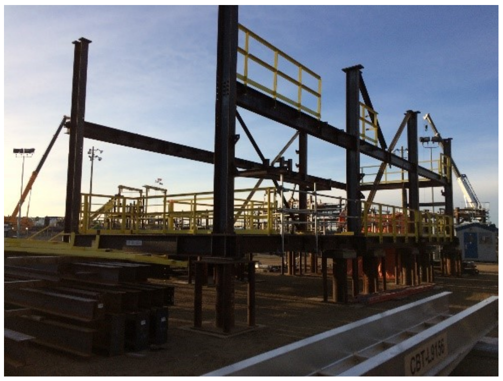


Figure 4.5: Construction module photograph example 5

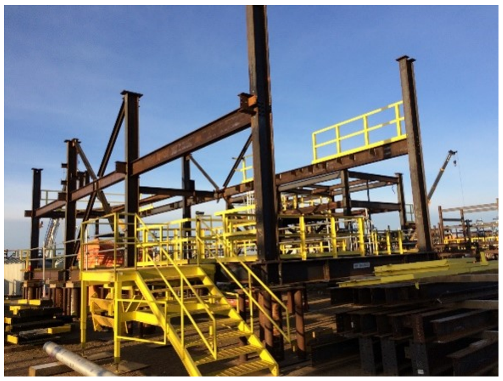


Figure 4.6: Construction module photograph example 6

not reasonable given the current level of high technologies and the degree of people interacting with them. Hence, the point cloud reconstruction experiments were performed focusing on point cloud quality and reconstruction time. Furthermore, there is a limited amount of information available on existing computer vision progress tracking approaches in constriction; as well, no tools were found to be available for progress tracking out of the box. Thus, this section is extensively focuses on experiments.

This section is organized in the way that it starts from the very first experimentation, then it describes the best tested solutions, and finally proceeds to the proposed point cloud registration approach for progress monitoring and discusses encountered challenges.

4.3.1 Initial Experiments

The very first experiments were based on self-written reconstruction code in Matlab on a synthetic data set. This approach required knowing camera parameters before proceeding with actual reconstruction. That is why camera calibration was performed with a Matlab Camera Calibrator for camera parameters estimation.

The initial step of camera calibration is to create a checkerboard with black and white cells of a constant known size. The 10 x 10 cells checkerboard was created with a cells size of 10 x 10 cm. Then, the photos of this board were taken from different angles. Some photos of the checkerboard are presented in Figures 4.7 and 4.8. The Matlab Camera Calibrator detects cells corners for each image and knowing the actual size of these cells it computes camera parameters and camera locations. Figure 4.9 shows the Matlab Camera Calibrator visualization results based on 15 photographs. The histogram plot indicates projection errors for each image as well as the mean error in pixels. The projection error is defined as the error between an actual image and mean image projected to the camera matrix. In this particular example the mean error is less than one pixel, which is likely caused by that the camera's inability to capture more explicit pictures than one

Figure 4.7: Checkerboard for camera estimation photograph example 1

Figure 4.8: Checkerboard for camera estimation photograph example 2

pixel. The other three plots in Figure 4.9 indicate camera locations in reference to the checkerboard.

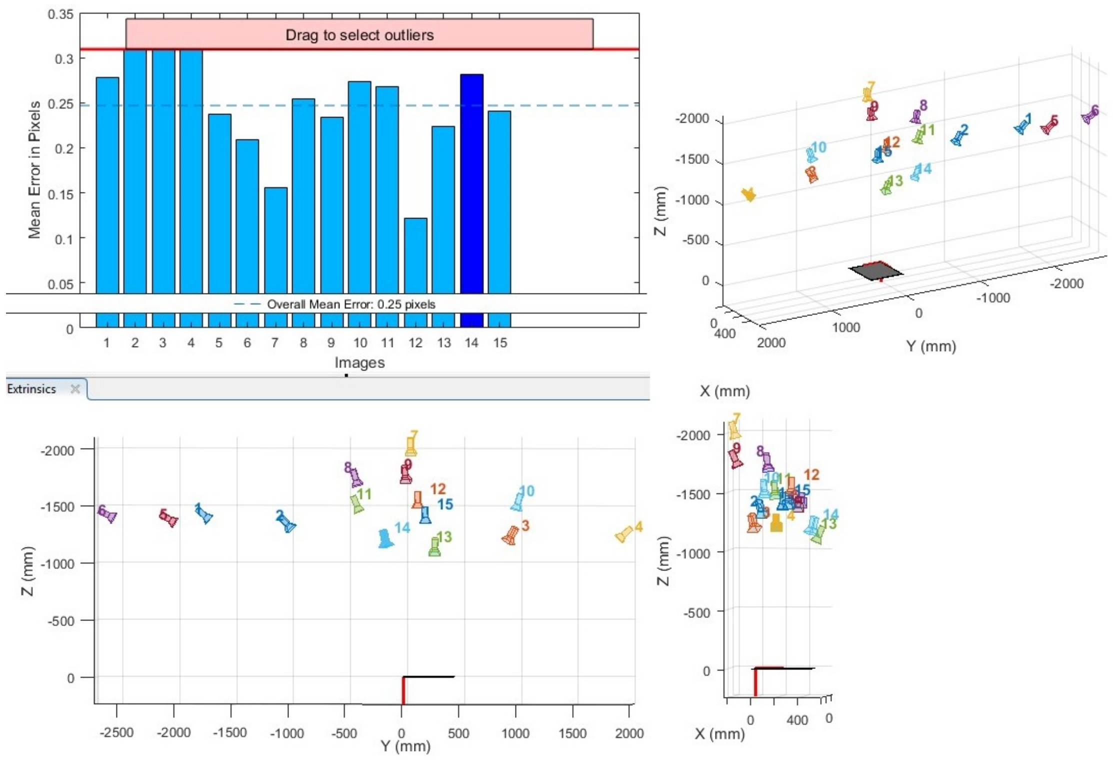


Figure 4.9: Matlab Camera Calibrator results visualization

After having the camera calibrated, the reconstruction of an android puzzle bot was attempted based on artificial data sets. These data sets were chosen for an initial test of the proposed approach, because its reconstruction is supposed to be

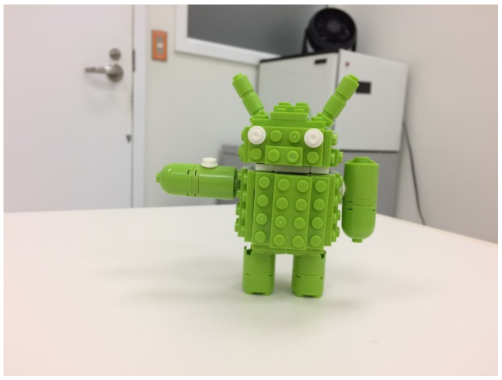


Figure 4.10: Android puzzle bot example 1

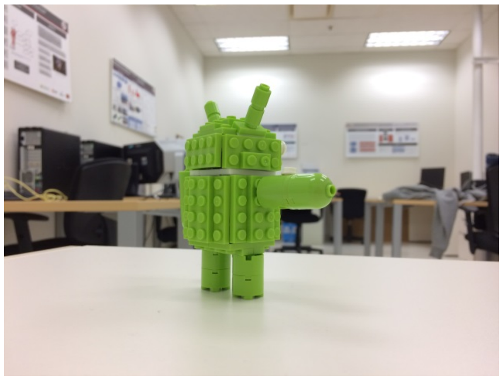


Figure 4.11: Android puzzle bot example example 2

easier and faster than industrial model reconstruction. Thus, the reconstruction of the android puzzle bot was tested on the data sets of different sizes starting from 15 to 40 images. Some examples of these data sets are presented in Figures 4.10 and 4.11.

The reconstruction attempts using self-written code with the Matlab Computer Vision System Toolbox failed. There were very few spatial points detected and the reconstruction took many hours. Hence, this method did not work and the search for a better solution started.

Next, the reconstruction of the same data sets was approached with VisualSFM structure from motion software package (C. Wu, n.d.) proposed by C. Wu (2013). The VisualSFM demonstrated better performance in terms of point cloud quality and reconstruction time. However, the output models were still very sparse and of weak quality. In fact, increasing the number of images did not grow the model density after some point.

After going through more computer vision papers and talking to computer vision researchers I realized that the android puzzle bot dataset is not suitable data for proper testing of SfM algorithm performance. SFM requires the object of interest to be of various textures, which is mentioned as a suggestion to a data set for SfM in subsection 3.2.1. However, the puzzle bot texture is very similar all around the object. This almost uniform structure of the puzzle bot resulted in a small number of distinctive points and consequently in a poor point cloud. In fact, data bases like the puzzle bot data one could be potentially reconstructed with different computer vision algorithms such as Shape From Silhouette. More information on this reconstruction method can be found in Cheung, Baker, and Kanade (2005).

This new realization forced me to test available algorithms with an actual module photo set (the same as shown in Figures 4.1 - 4.6). The first reconstruction was performed based on one hundred 8 MP photographs. The Matlab code implementation was terminated after a day of running. VisualSFM demonstrated acceptable computational speed (less than a day) and point cloud density. The investigation of possible enhancements revealed that the VisualSFM computation time can be improved by enabling the VisualSFM CUDA option. CUDA technology (C. Nvidia (2010)) dramatically increases computer performance by employing computation on a graphics processing unit (GPU) instead of a central processor unit CPU, although the CUDA option can be enabled only if an Nvidia is available for computation. That is why a special computer was purchased for this research. This computer characteristics are Intel i7-7700K, Nvidia GTX Titan X with 12 GB GDDR5X and 16 GB DDR4 2400 MHz. Therefore, computation on 3584 cores of Nvidia GTX Titan X (Nvidia (n.d.)) instead of computation on a processor such as Intel i7-7700K with its 8 threads decreased computation time significantly.

Having an effective computer, the industrial module reconstruction was performed based on different numbers of photographs for determining optimal data set size. One hundred photos reconstruction indicated an adequate point cloud density. Four hundred photos demonstrated better density, but it took considerably more time than the first experiment. A two thousand images VisualSFM test run was forcibly terminated after running for more than one day. One day+ computation time is not considered to be acceptable within the framework of this research. Ultimately, an empirical approach demonstrated that the input of two to four hundred images in 8 MP resolution is optimal for VisualSFM module reconstruction of an industrial module.

In addition, a video around an industrial module was recorded. The video length is about 2.5 minutes. Using the iPhone 5s camera's capturing frequency of up to 30 frames per second, around 4,000 frames were extracted out of this video. The frame extraction was written and performed in Matlab. This photo set has lower resolution images, whereas the transition between the subsequent images is very smooth. Noticeably, the smooth transition has favorable effects on the reconstruction result as it helps to detect more consequent matches, which subsequently densifies a point cloud. Thus, the reconstruction of the same industrial module was performed based on various numbers of frames. Ultimately, the input of four to six hundred video frames produced much denser point cloud than the one based on photos.

The resulting point cloud models are saved in the ply format. The ply models are viewed with the Meshlab software (*Meshlab*, n.d.) presented by Cignoni et al. (2008). The resulting screenshot of the VisualSFM sparse point cloud is shown in Figure 4.12.

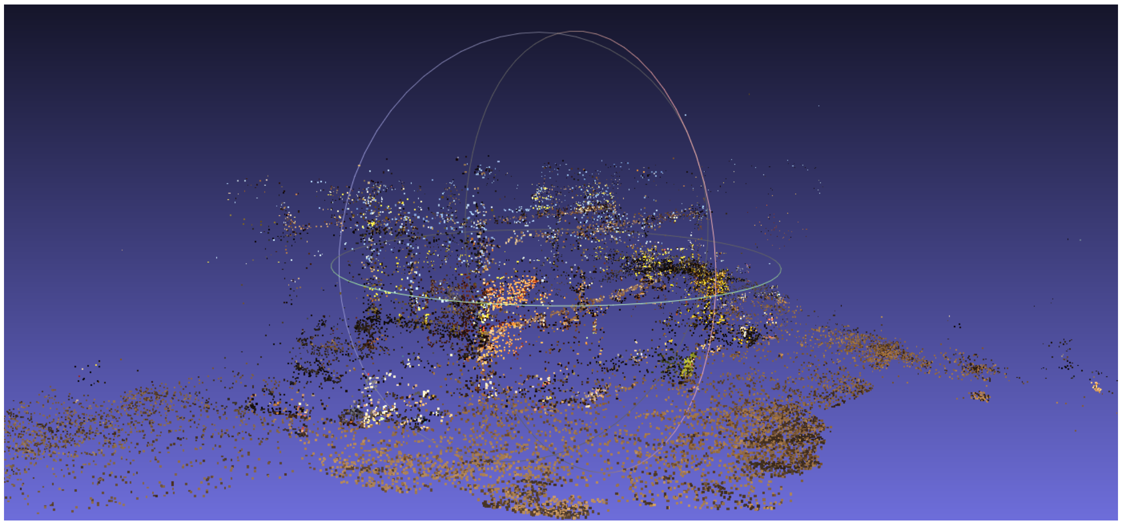


Figure 4.12: Sparse industrial module point cloud produced by VisualSFM

Then the sparse point cloud model was densified with the CMVS multi-view stereo software (Furukawa (n.d.)) proposed by Furukawa and Ponce (2010). This approach radically increased the point cloud density as can be seen in Figure 4.13.

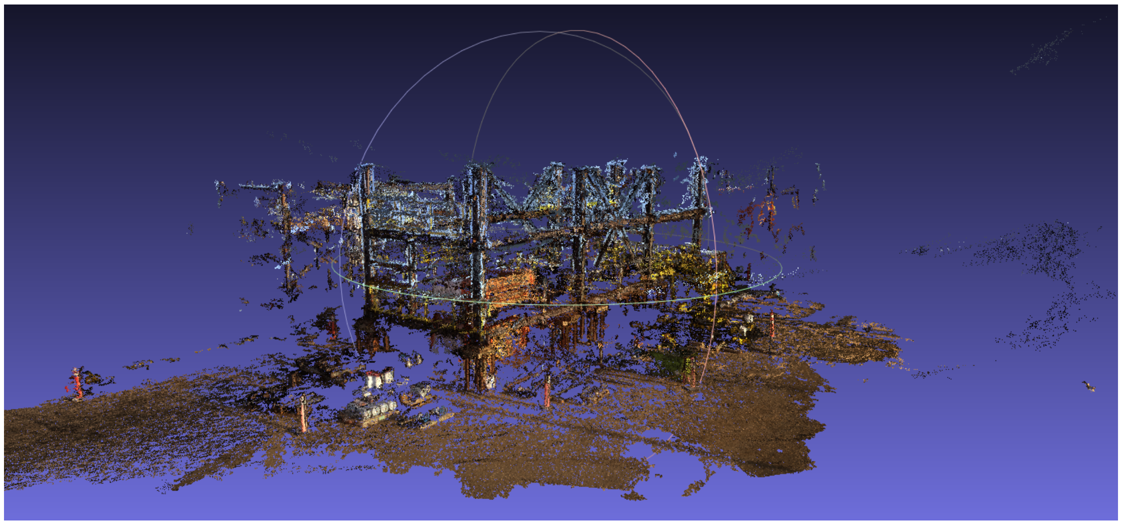


Figure 4.13: Dense industrial module point cloud produced by CMVS based on VisualSFM output

Later, the sparse reconstruction results was improved by using the CMVS software package (Jancosek (n.d.)) presented in Jancosek and Pajdla (2011). This software creates a mesh by covering segments of densified points with rectangular patches taken from the input images. CMVS was tested on previously produced sparse point clouds of the industrial module. Notable, CMVS improved the model visual quality, although its performance indicated that the increasing number of points in the sparse point cloud does not always guarantee better mesh model reconstruction results. Presumably, the reason for this "unexpected" mesh reconstruction depending on the number of images is that the large number of photos increases the projection error, which is likely to add extra noise to the model. This extra noise then increases the number of incorrect clusters. Ultimately, the best performance on the tested data sets was achieved with four to six hundred video frames.

The schematic effect of the CMVS package on a point cloud is illustrated in Figure 4.14. There are three images of one structure from different angles in the top of the figure. Also, there is a fragment consisting of 4 points on a point cloud that is shown in the left bottom rectangle; as well, its points projections on the input images are demonstrated. The right bottom rectangle illustrates the same fragment on a mesh model obtained by employing the CMVS algorithm.

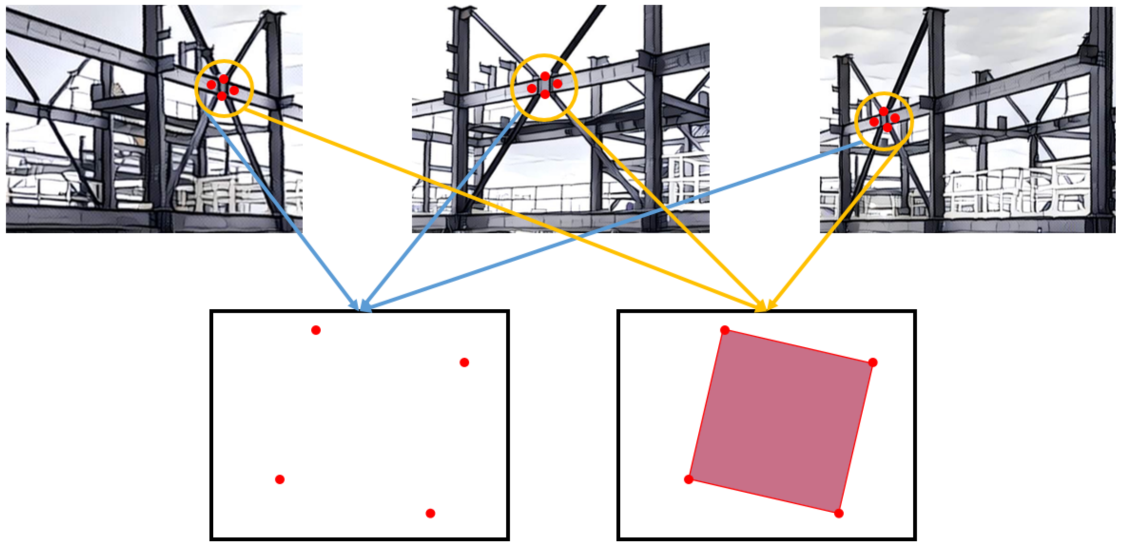


Figure 4.14: CMPMVS effect on a point cloud fragment

The mesh model of the industrial module which was produced by the CMPMVS tool based on the sparse point cloud is presented in Figure 4.15.

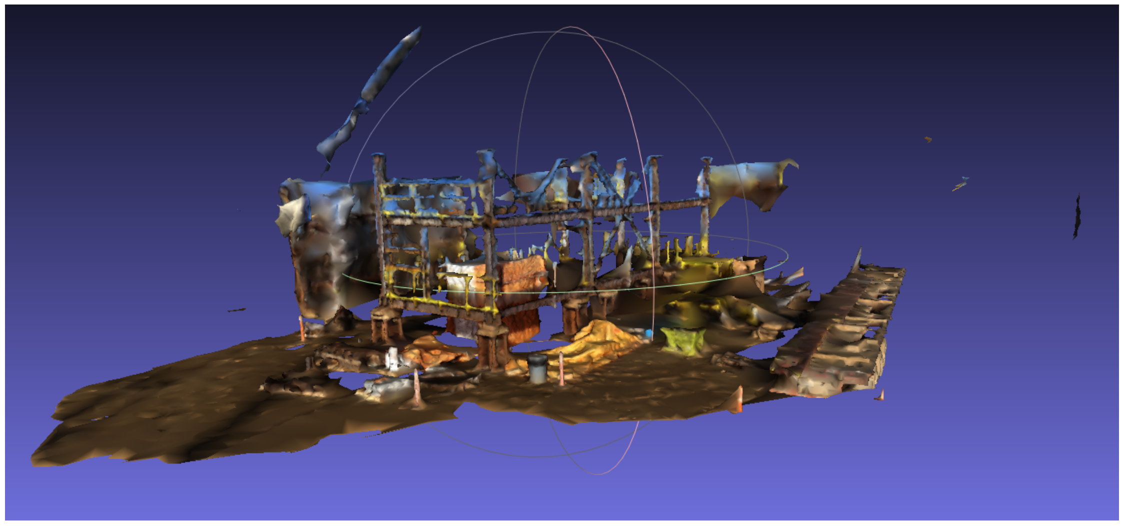


Figure 4.15: Mesh of the industrial module produced by CMPMVS based on VisualSFM output

4.3.2 Experiments with the Newest Software

Although the previous results were quite impressive at first glance, the computation time was still somewhat slow and some important structural parts were missing. The VisualSfM-CMVS and VisualSfM-CMPMVS require at least four hundred video frames, and the computation time is more than four hours. The resulting point clouds size are 1,400,000 and 86,000 points, respectively. Also, Figure 4.16 demonstrates the largest piece which is missing in the reconstructed models and this structural part is marked in red. Figures 4.17 and 4.18 are the screenshots of the CMVS and CMPMVS. The place where the specified piece should have been is marked in red. Having finished VisualSfM-CMVS and VisualSfM-CMPMVS reconstruction, I concluded that the reconstruction results were not good enough as well as not fast enough and I proceeded to looking for a better reconstruction tool.

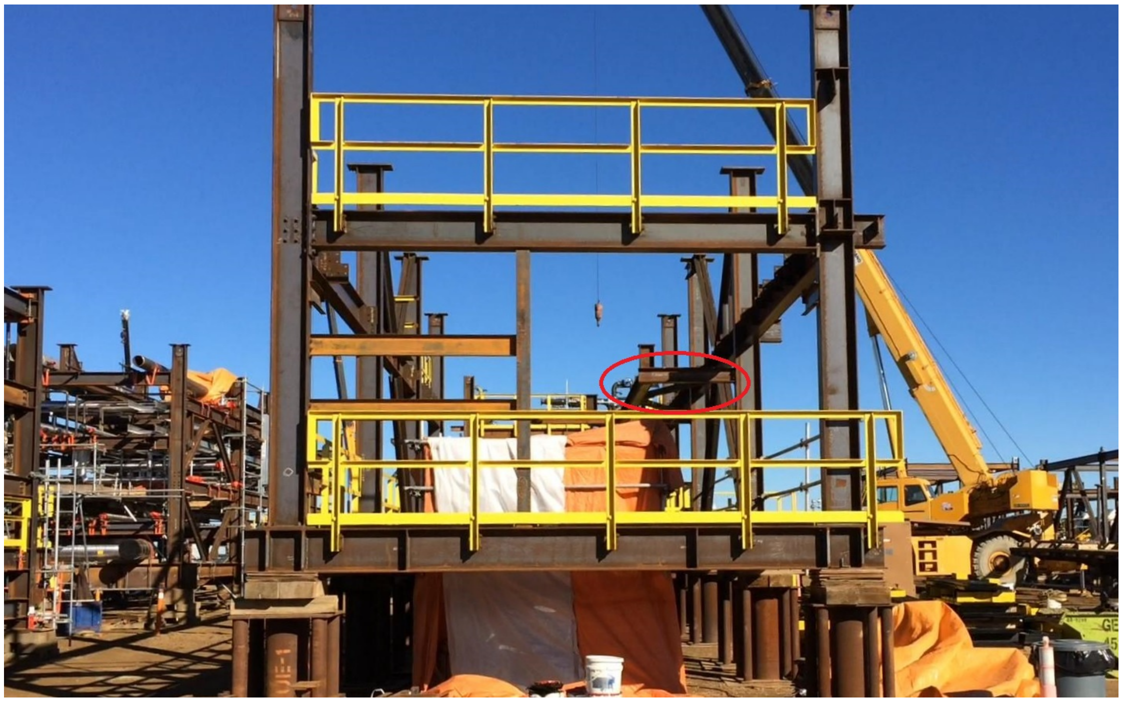


Figure 4.16: Industrial module photograph with the missed in the reconstruction models piece marked in red

Keeping the quality and time issues in focus, I started searching for other top-notch software packages and libraries available for 3D reconstruction. What I came

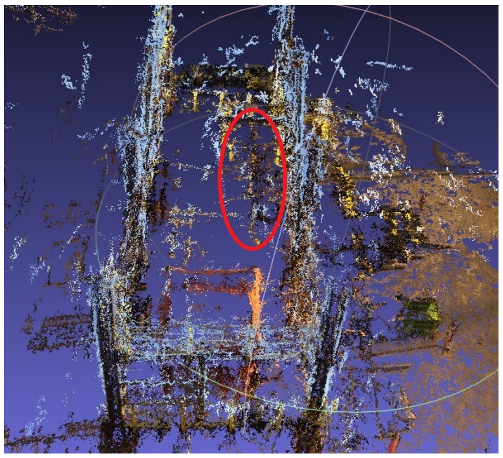


Figure 4.17: CMVS reconstruction model with the missing piece marked in red

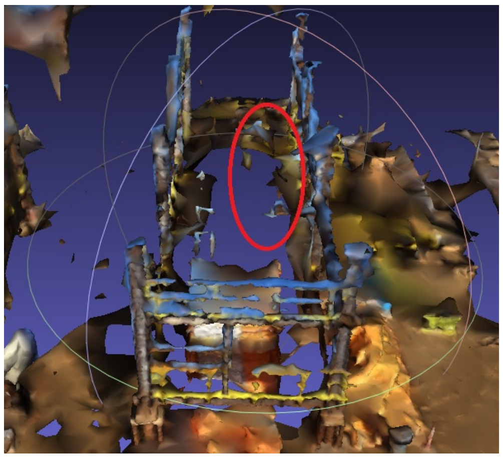


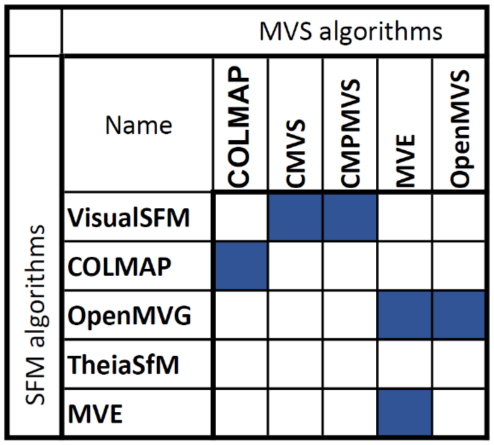
Figure 4.18: CMPMVS reconstruction model with the missing piece marked in red

across were the following products:

- COLMAP (Schnberger (n.d.)) SfM-MVS software presented by Schonberger and Frahm (2016) and Schönberger et al. (2016);
- OpenMVG (Moulon, Monasse, Marlet, and Others (n.d.)) SfM library presented by Moulon, Monasse, Perrot, and Marlet (2016);
- TheiaSfM (Sweeney (n.d.)) SfM library presented by Sweeney, Hollerer, and Turk (2015);
- MVE (Fuhrmann (n.d.)) SfM-MVS library presented by Goesele, Snavely, Curless, Hoppe, and Seitz (2007);
- OpenMVS (*OpenMVS: open Multi-View Stereo reconstruction library* (n.d.)) MVS library presented by Moulon, Monasse, and Marlet (2013).

All the previously mentioned tools such as VisualSFM, CMVS and CMPMVS are free for research purposes; however, they are closed source. In contrast, all the discovered solutions such as COLMAP, OpenMVG, TheiaSFM, MVE, OpenMVS are open source C++ tools.

Considering that this thesis work requires an as-planned model to be a dense point cloud and that the described software proposes SfM or MVS or SfM-MVS reconstruction options, these tools are used as stand alone tools as well as in combinations of different libraries. Figure 4.19 indicates all the combinations of the software packages tested in this work.



		MVS algorithms				
	Name	COLMAP	CMVS	CMPMVS	MVE	OpenMVS
SFM algorithms	VisualSFM		■	■		
	COLMAP	■				
	OpenMVG				■	■
	TheiaSfM					
	MVE				■	

Figure 4.19: Combinations of software tested in this work

Experimenting with different reconstruction techniques demonstrated that the optimal data set for the majority of libraries is of 100 - 200 video frames. Adding more images does not make a big difference in terms of point cloud density whereas it somewhat increases computation time.

TheiaSfM did not produce a proper reconstruction with any of tested data sets. This library considered the tested datasets as combinations of small sets. This issue led to reconstruction of each dataset as separate two or three point clouds. Hence, each separate point cloud was of low density with some parts missing. Many different reconstruction options have been tested, but they did not help. Because of this, the experiments with Theia library stopped at this point. Similarly, the reconstruction of large datasets with VisualSfM takes a lot of time and still lacks quality. Thus, the experiments with CMVS and CMPMVS stopped at this point too.

For all the other libraries, a data set was formed for testing their performance on the actual construction project. This test data set was organized consisting of 138 video frames obtained by uniformly sampling a 2 min 19 sec video at a 1 frame per second sampling rate. The specified dataset was passed to each reconstruction algorithm combination. The reconstruction results are reported herein. All the experiments were conducted on the Intel i7-7700K, Nvidia GTX Titan X, 16 GB DDR4 computer running Ubuntu 16.04 LTS.

Ultimately, the clouds are reconstructed by the following libraries or libraries pairs:

- MVE, where SfM and MVS reconstruction parts are performed fully by MVE;
- OpenMVG-MVE, where SfM is performed by OpenMVG and MVS is performed by MVE;
- OpenMVG-OpenMVS, where SfM is performed by OpenMVG and MVS is performed by OpenMVS;
- COLMAP, where SfM and MVS are performed fully by COLMAP with default reconstruction parameters;
- COLMAP-adjusted, where SfM and MVS are performed fully by COLMAP but the image maximum size (height or width) is set to 800 pixels inside the program.

COLMAP-adjusted was chosen as a “light” version of the COLMAP approach. It is faster than COLMAP but has lower point cloud density.

The numbers of points in each point cloud are presented in Figure 4.20 and the reconstruction time for each cloud is shown in Figure 4.21.

Based on the statistics, one might think that the MVE and OpenMVG algorithms produced the best point clouds in terms of density. However, these point clouds possess the biggest amount of blue noise. The output point clouds screenshot of the presented software packages are demonstrated in Figures 4.22 - 4.26. In fact, the blue noise is caused by the sky in photographs.

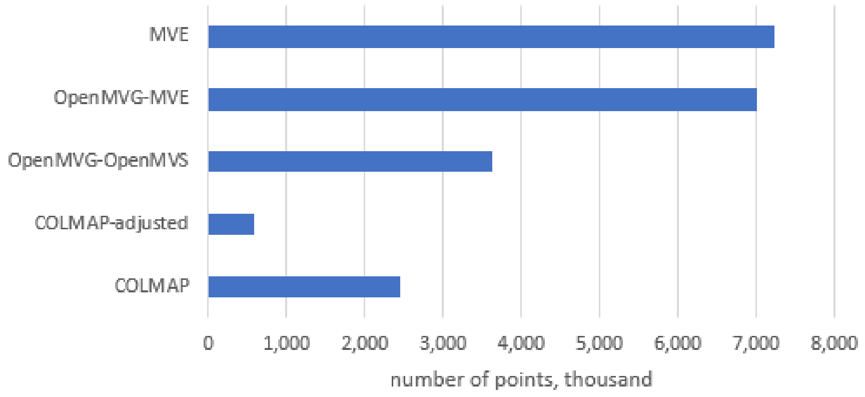


Figure 4.20: SfM-MVS algorithms number of reconstructed points

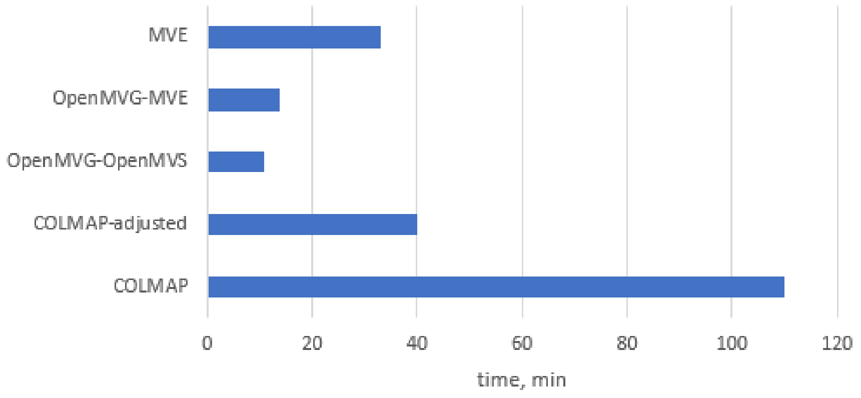


Figure 4.21: SfM-MVS algorithms computational time

In conclusion, the COLMAP and the OpenMVG-OpenMVS reconstructions are able to demonstrate the module silhouette, whereas the rest of the models are completely noisy.

4.3.3 Filtering Noise

Although the noise caused by blue sky should be easy distinguished from the module part of point clouds based on its color, I did not find any out of the box solution for this problem. That is why I wrote the code for separating the sky and the module based on my logical understanding of the red-green-blue (RGB) color space.

There are two lessons I learned while testing different approaches to noise filter-

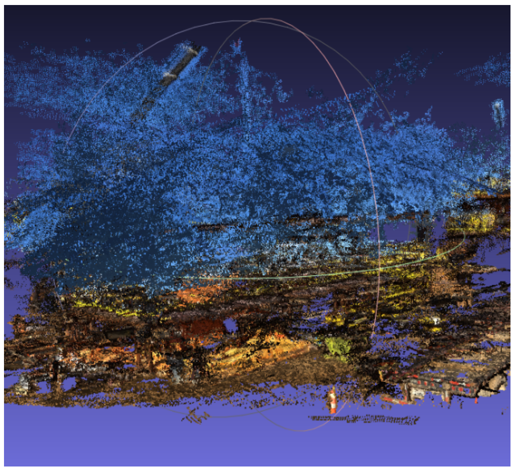


Figure 4.22: MVE reconstructed point cloud

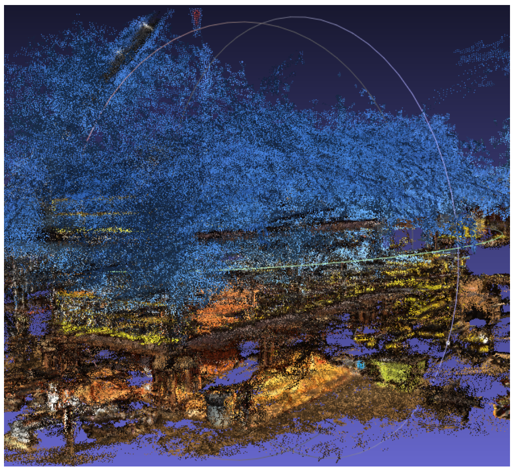


Figure 4.23: OpenMVG-MVE reconstructed point cloud

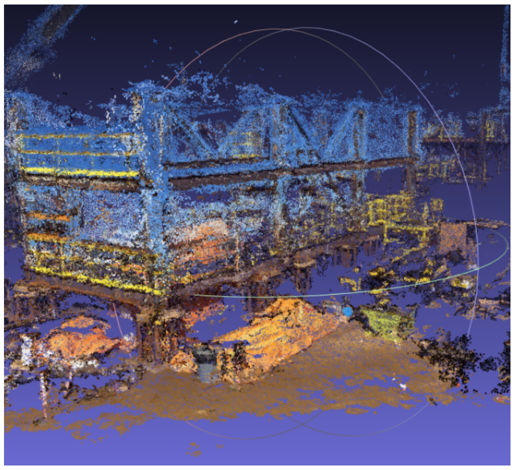


Figure 4.24: OpenMVG-OpenMVS reconstructed point cloud

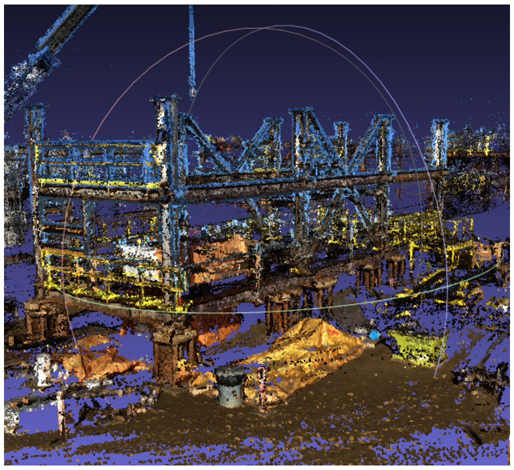


Figure 4.25: COLMAP reconstructed point cloud

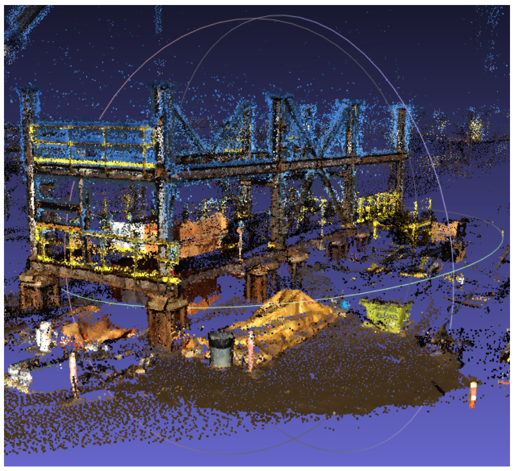


Figure 4.26: COLMAP-adjusted reconstructed point cloud

ing. First, there are no module parts similar to the sky color. Second, filtering the points with maximum value of blue channel and not considering the values of the red and green channel does not guarantee filtering actual blue points. For example, a high value of blue and a high value of red gives a purple color which is not the color of interest at this point.

Here is the best empirical found formula for noise filtering based on the reconstructed point clouds. The value of interest might be called blue concentration and its formula is the following:

$$blue\_concentration = blue - max(red, green) \tag{4.1}$$

where $blue$, red and $green$ are values of corresponding color channels in the scale $[0, 255]$

The concept of this equation is that it calculates concentration of “poor” blue relatively to the rest of the colors. In other words, if the value of blue is high and the values of the red and the green are low, than this color is blue. Thus, blue concentration is calculated for each point in the point cloud and then the values higher than a certain threshold are filtered out. In this work, the blue concentration threshold of 35 is used. This number was obtained experimentally based on the

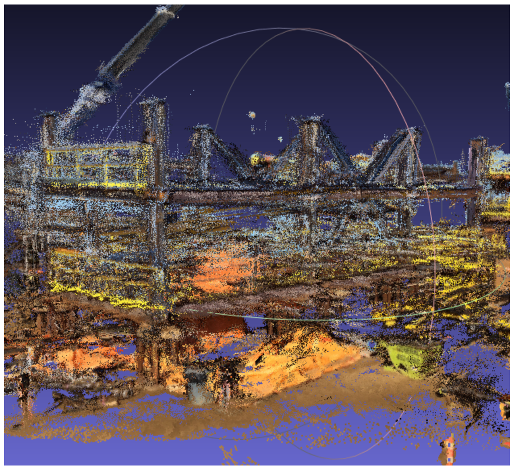


Figure 4.27: MVE filtered point cloud

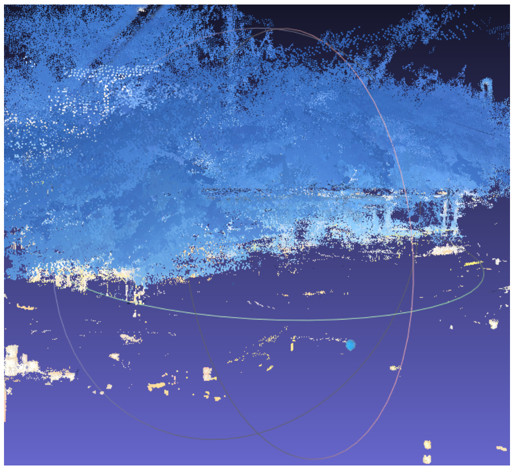


Figure 4.28: MVE noise point cloud

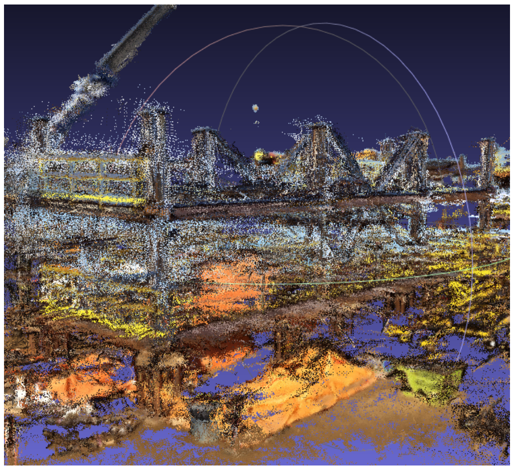


Figure 4.29: OpenMVG-MVE filtered point cloud

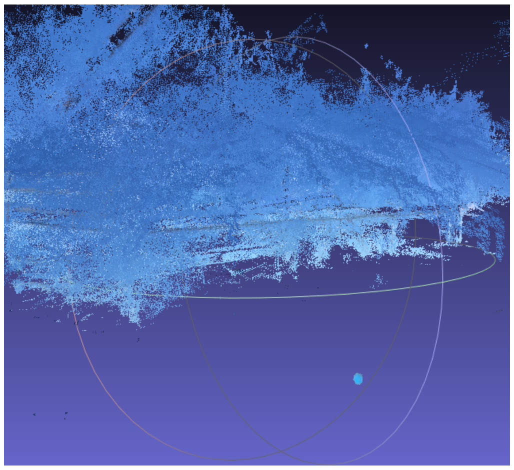


Figure 4.30: OpenMVG-MVE noise point cloud

author's visual perception.

After filtering the MVE and the OpenMVG-MVE, the reconstruction was still left with a considerable amount of nose. The filtered versions of these point clouds along with their noise point clouds are shown in Figures 4.27 - 4.30 for demonstration of the noise filtering algorithm performance. At the same time, the COLMAP, COLMAP-adjusted and the OpenMVG-OpenMVS point clouds were significantly improved. Their filtered versions are show in Figures 4.31 - 4.33.

The filtering process takes less than a minute on 584,000 - 3,632,000 points point clouds, which is why the filtering time has modest effect on the overall progress

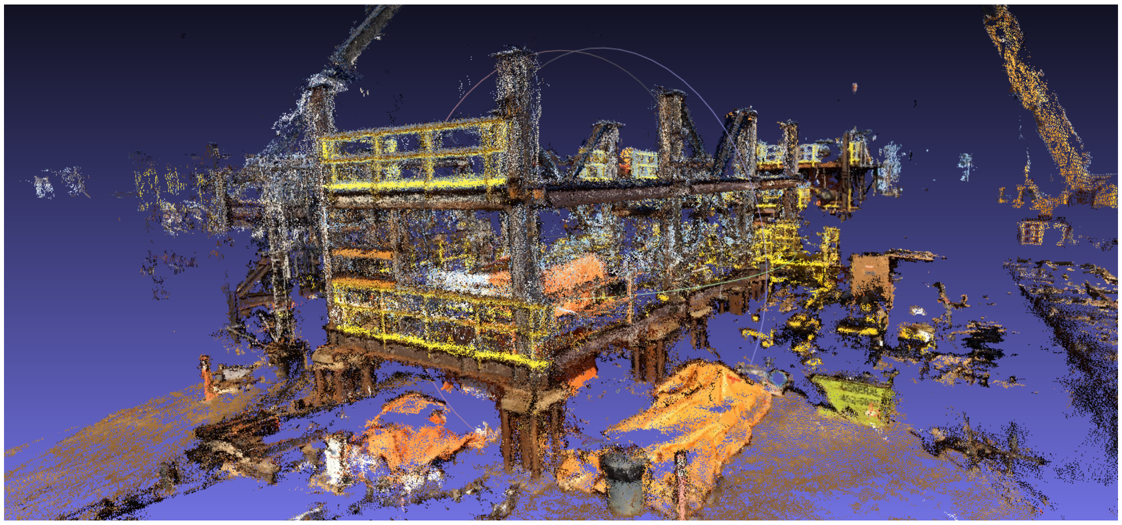


Figure 4.31: OpenMVG-OpenMVS filtered point cloud

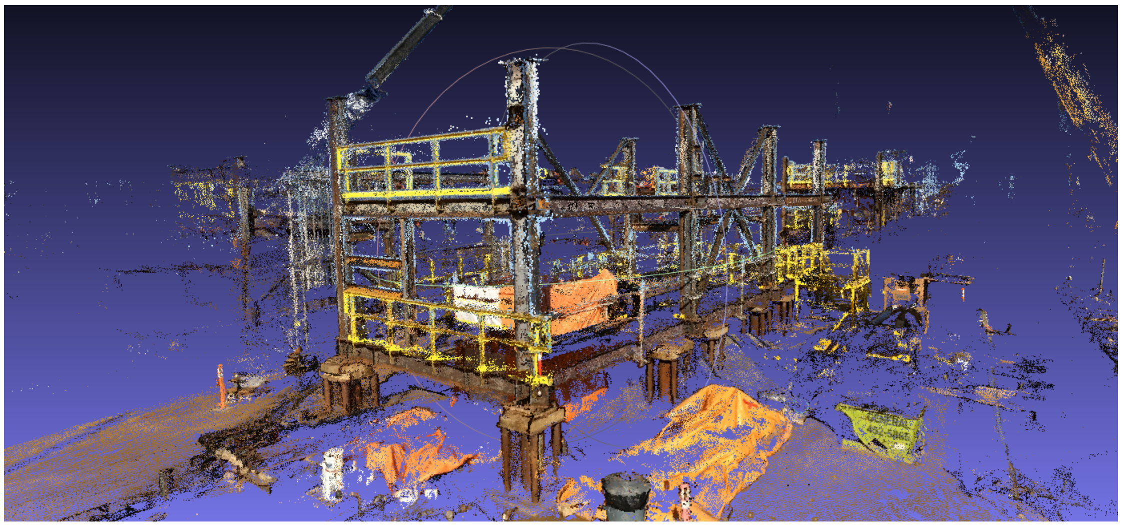


Figure 4.32: COLMAP filtered point cloud

tracking time. Simultaneously, the numbers of points in the clouds are changed substantially. The updated numbers of point as well as the contrasting initial numbers of points are presented in Figure 4.34. Around 20% of the points are removed as outliers in the MVE and OpenMVG-MVE point clouds, and about 10% in the rest of the point clouds.

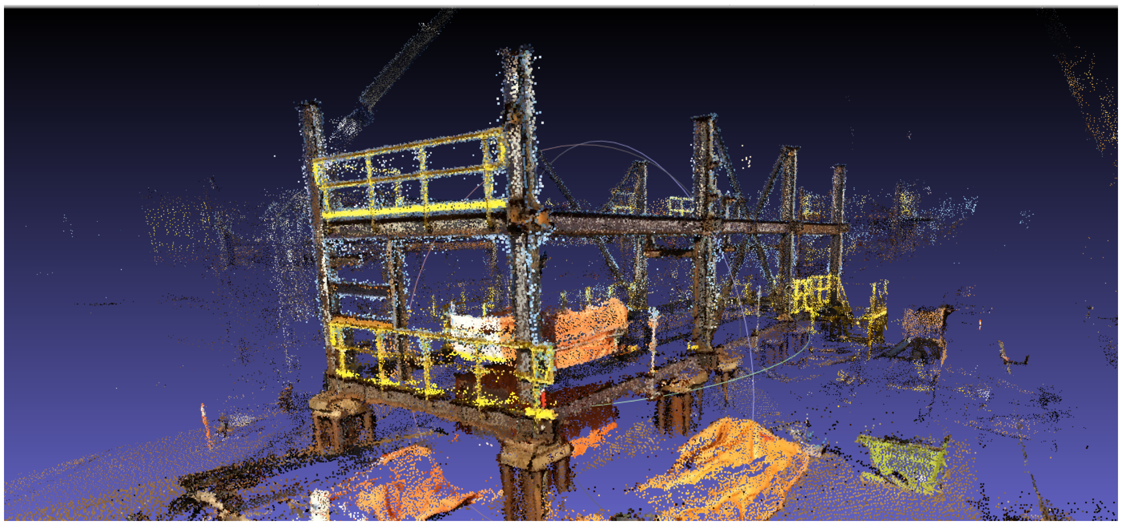


Figure 4.33: COLMAP-adjusted filtered point cloud

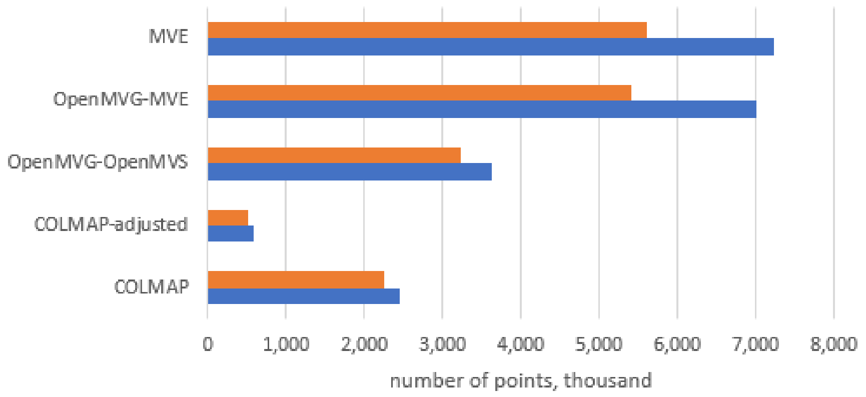


Figure 4.34: SfM-MVS algorithms number of reconstructed points. The orange color implicates the number of points in the filtered point clouds. The blue indicates the numbers of points in the original point clouds

4.3.4 Proposed Reconstruction Software

The MVE and the OpenMVG-MVE filtered point clouds still have too much noise, which is likely to affect progress measurement results. Because of this, the experiments with these libraries were stopped at this point.

Thus, the there are two libraries combinations, COLMAP and OpenMVG-OpenMVS, that produced good quality point cloud models in a reasonable time. In the case of industrial modules reconstruction, COLMAP and OpenMVG-OpenMVS clearly outperformed VisualSFM-CMVS, VisualSFM-CMPMVS, and TheiaSfM and somewhat outperformed the points number forerunners the MVE and the OpenMVG-MVE because these libraries results are excessively noisy. Considering the properties of the reconstructed point clouds, COLMAP makes the most visually precise reconstructions with a minimum amount of noise. At the same time, OpenMVG-OpenMVS is the fastest presented approach to the reconstruction with time of modules reconstruction of 11 min versus 110 min of the standard COLMAP reconstruction, although it creates notable amount of noise. The COLMAP-adjusted approach is located somewhere in between the already mentioned approaches in terms of point cloud quality and time. The COLMAP-adjusted produces less dense point cloud than COLMAP as well as less amount of noise than OpenMVG-OpenMVS whereas its computation time is 40 min.

All three approaches demonstrated similar progress tracking performance, which are discussed later in this chapter. Selecting a particular reconstruction technique depends on the circumstances. When high attention to details is desired, it is recommended to use COLMAP; when fast reconstruction is required, OpenMVG-OpenMVS would be preferable, and the COLMAP-adjusted is the trade-off between attention to details and computation speed. These and all the rest of the approaches' characteristics are presented in Table 4.1.

Table 4.1: Fundamental characteristics of the presented reconstruction software

Properties	COLMAP	COLMAP -adjusted	OpenMVG -OpenMVS
Type	open source	open source	open source
Platform	Windows/Linux	Windows/Linux	Windows/Linux
Compilation required	no for Windows, yes otherwise	no for Windows, yes otherwise	yes
SfM computation time, min	4	4	2.5
MVS computation time, min	106	36	8
Number of points in filtered cloud, thousand	2,258	514	3,237
User interface	yes	yes	no
Command line interface	yes	yes	yes
Documentation	yes	yes	yes

There is not much literature comparing the reconstruction libraries, however Schonberger and Frahm (2016) reported the SFM part of COLMAP to significantly outperform VisualSfM and ThieaSfM in terms of completeness. Likewise, Schöps et al. (n.d.) reported the MVS part of COLMAP substantially surpassing CMPMVS and PMVS (which is based on CMVS) in terms of accuracy and completeness on 2 cm evaluation threshold. These paper confirm the obtained results in this work.

There are also some interesting computation parts which might be accessed through the COLMAP user interface. They are presented in Figures 4.35 - 4.45 which demonstrate different algorithm parts performance on the actual data set. This part follows up the flow presented in Chapter 3.

The first COLMAP step is reconstruction of a sparse point cloud with SfM. The resulting sparse point cloud is shown in Figure 4.35. Also, camera centers with their views and orientations are illustrated around the point cloud in red. Thus, it can be seen where the photos were taken relatively to the object of interest itself.

I chose to demonstrate the COLMAP flow on on one main image and two

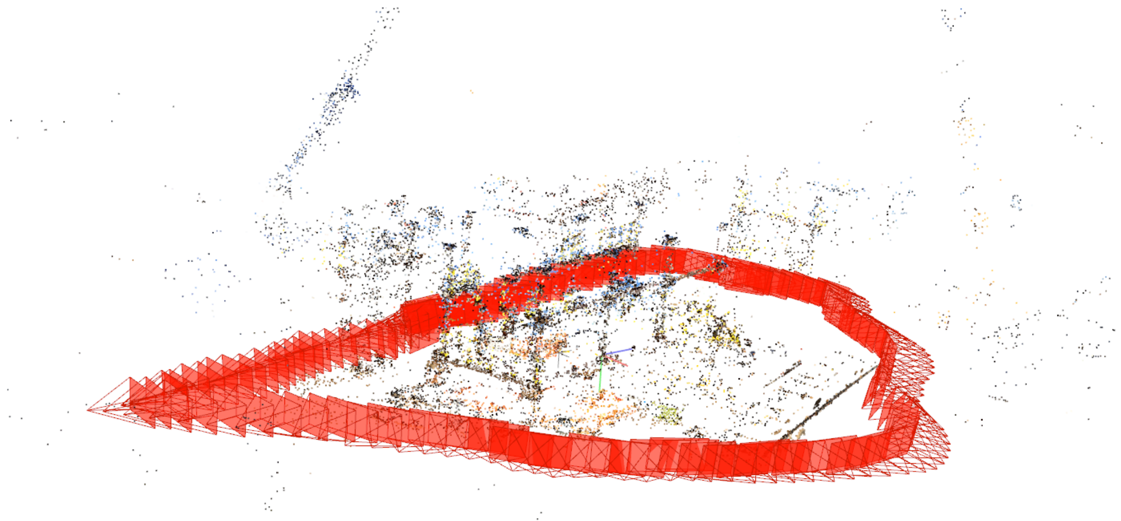


Figure 4.35: COLMAP sparse point cloud reconstruction

axillary ones. The main image was chosen randomly and it is the main one only for presentation purposes, not for reconstruction. The main image with distinctive points computed with SIFT and marked in red is presented in Figure 4.36. This photograph is number 130 in COLMAP. Figure 4.37 demonstrates the number of putative matches of the image 131 with the other images. Image 131 was chosen to be one of the axillary images as it has the maximum matches with the main image. Likewise, image 137 was chosen as the one having a medium number of matches with the main image. Images 131 and 137 are shown in Figures 4.38 and 4.39, correspondingly.

The visualization of putative matches between the main and the axillary images are shown in Figures 4.40 and 4.41.

The next reconstruction step is depth map computation. The examples of depth maps are shown in Figures 4.42 - 4.44. In these images the closer something is in space to the camera the more it is deep blue, whereas light blue indicates further distance, and other colors show great distance of objects from the camera.

Finally, the resulting dense point cloud is shown in Figure 4.45.

Having the as-built point cloud ready allows us to move on to the next section after specifying the challenges faced experimenting with point cloud reconstruction

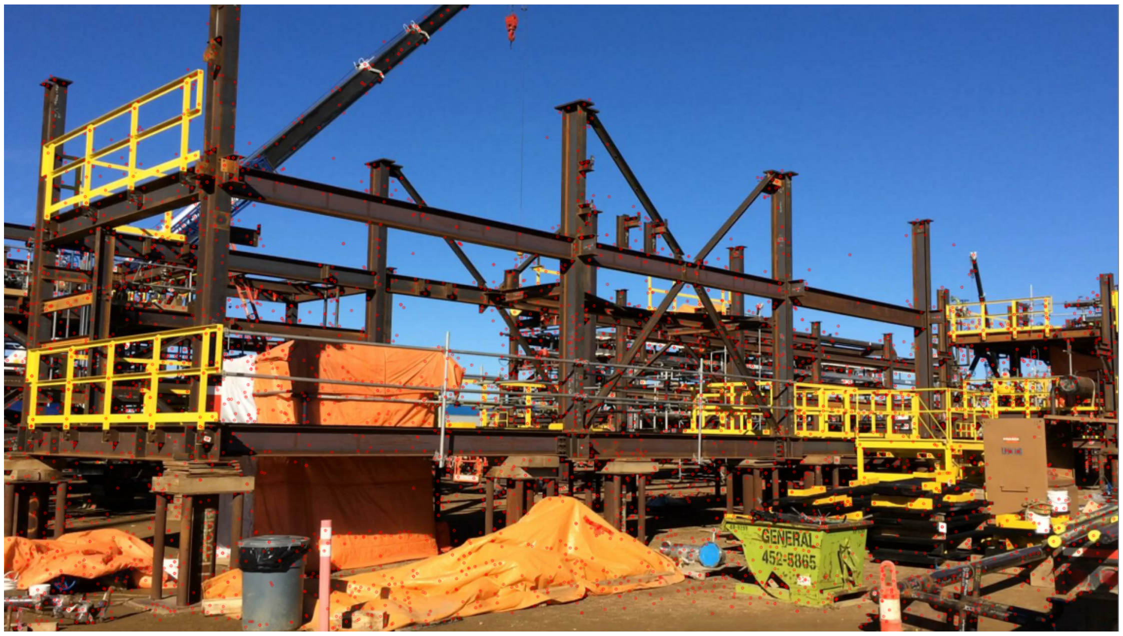
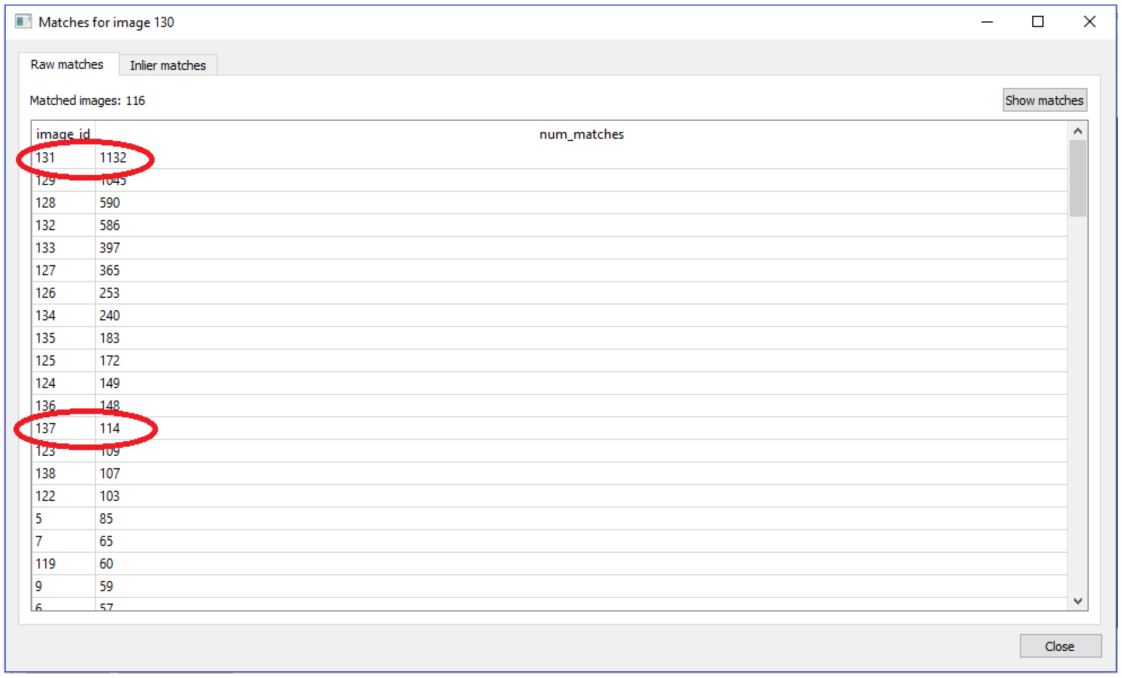


Figure 4.36: COLMAP distinctive points detection in image 130



Matched images: 116

image_id	num_matches
131	1132
129	1045
128	590
132	586
133	397
127	365
126	253
134	240
135	183
125	172
124	149
136	148
137	114
123	109
138	107
122	103
5	85
7	65
119	60
9	59
6	57

Figure 4.37: COLMAP number of matches of image 131 with other images

software. Here is the list of encountered difficulties:

- vision limitations. Not all of the parts can be captured with a hand held camera because of accessibility limitation. An example of such in this work are

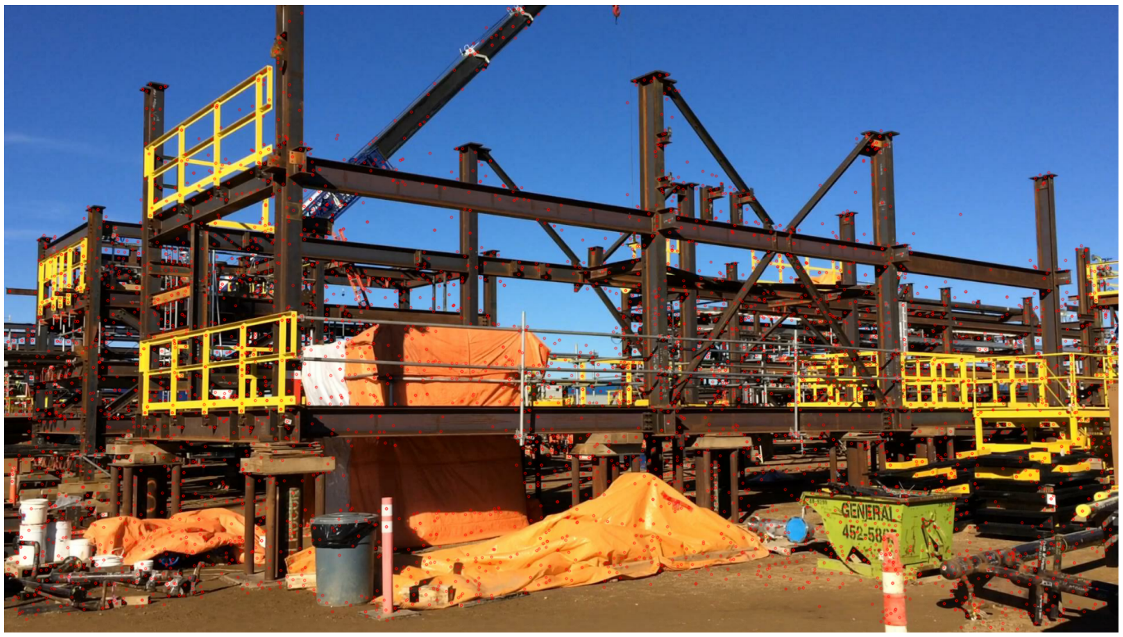


Figure 4.38: COLMAP distinctive points detection in image 131

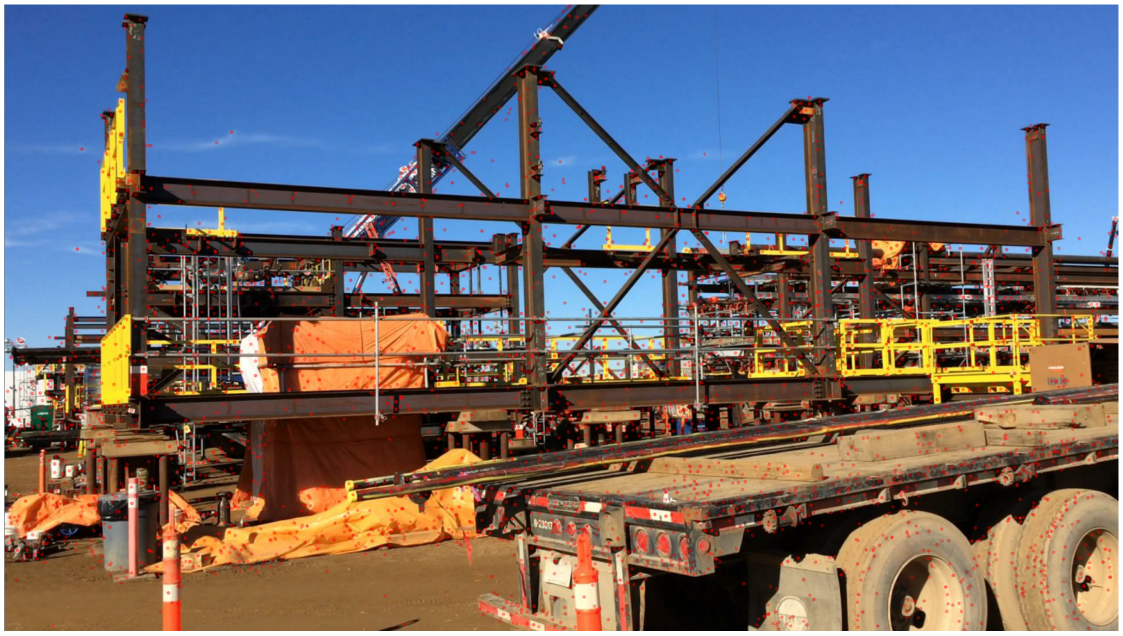


Figure 4.39: COLMAP distinctive points detection in image 137

module bottom or “floor” parts which are almost unseen in the photographs. An example of this issue can be seen in Figure 4.46. However, this figure clearly shows the second level structural part that was missing in the CMVS and CMPMVS reconstruction (referenced in Figures 4.17 and 4.18);

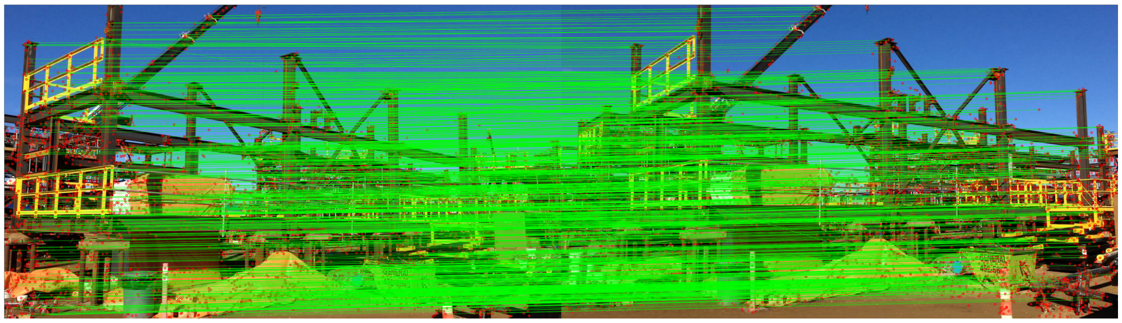


Figure 4.40: COLMAP putative matches between images 130 and 131

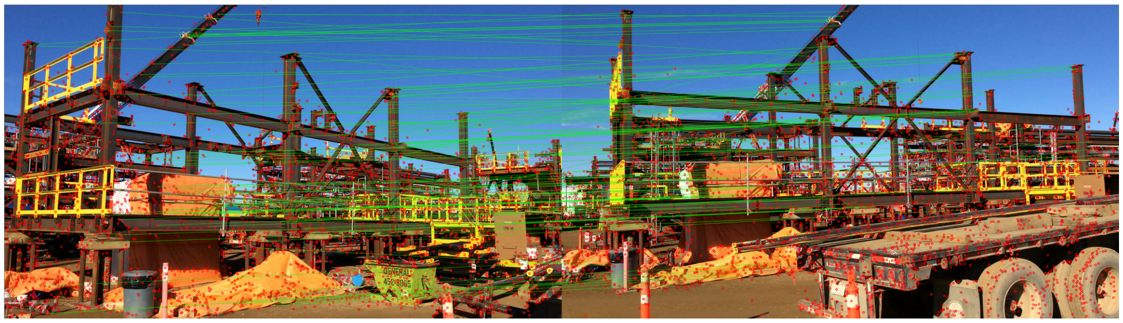


Figure 4.41: COLMAP putative matches between images 130 and 137

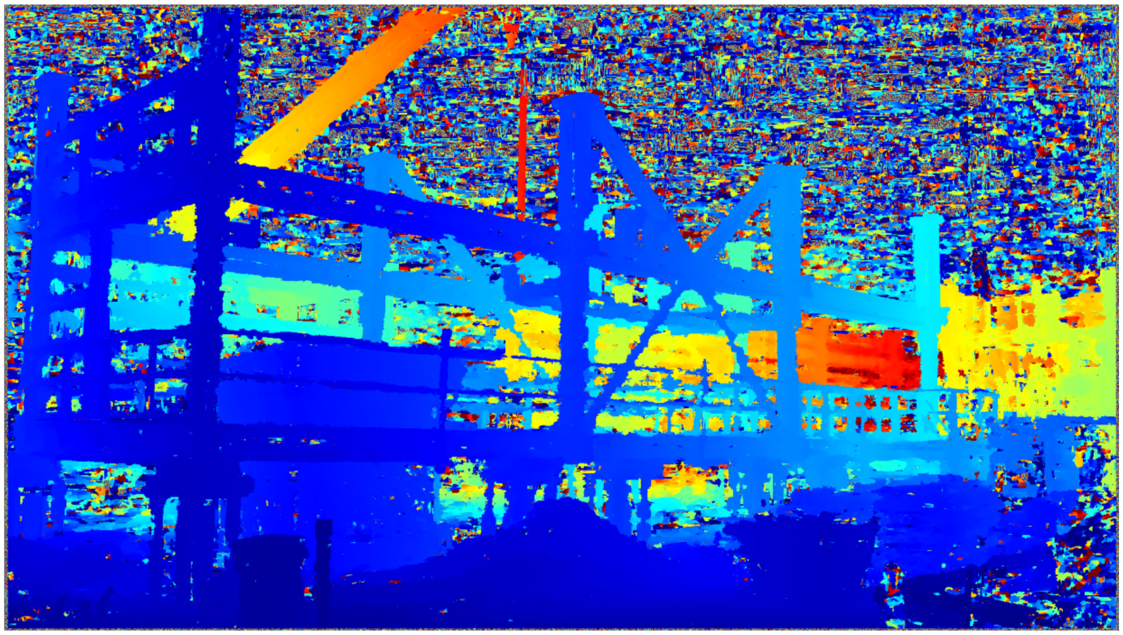


Figure 4.42: COLMAP depth map of image 130

- technical issue of having blue noise in the photographs;
- bad documentation and lack of tutorials. These kinds of issues make library

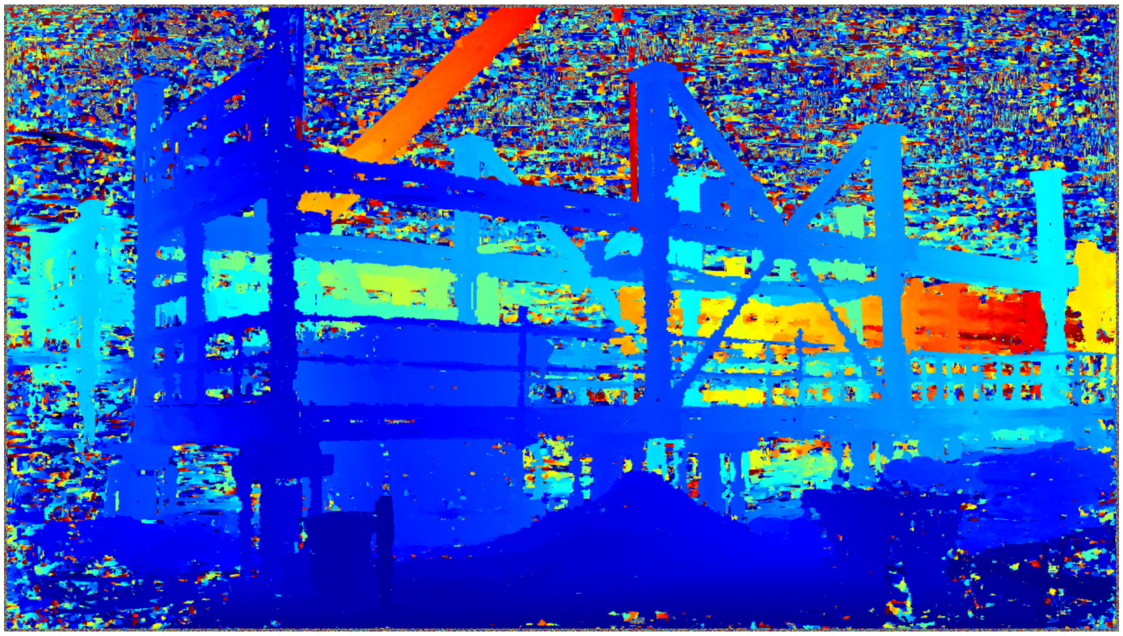


Figure 4.43: COLMAP depth map of image 131

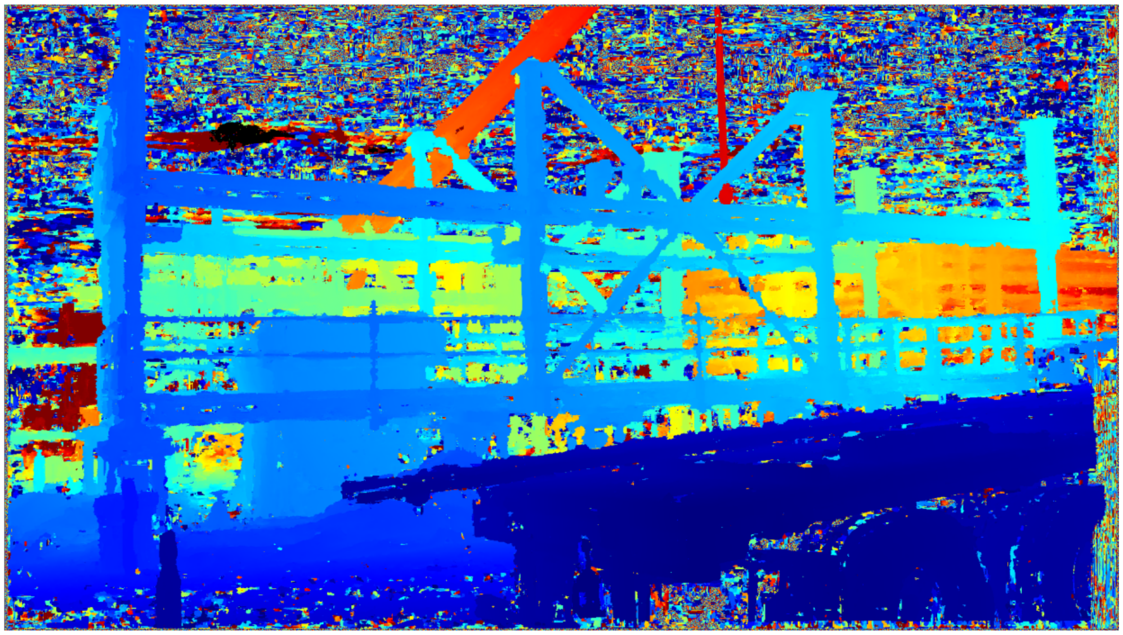


Figure 4.44: COLMAP depth map of image 137

use very difficult or even impossible in some cases;

- all the tested libraries which required compilation have their own dependences, which might have even their own dependences. Occasionally, it is

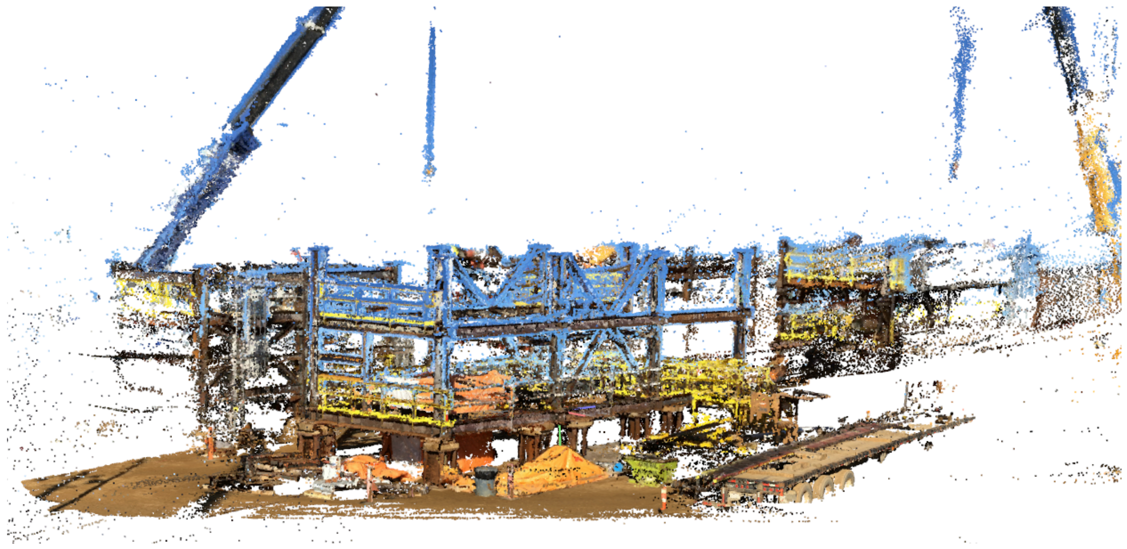


Figure 4.45: COLMAP dense point cloud reconstructed

difficult to find proper dependences for stable work of a library of interest. Furthermore, sometimes dependences linking or setting up environment variables is required. Again, this requires at least a basic understanding of Linux, Cmake and C++;

- dealing with code mistakes, bugs and poor support. Code mistakes and bugs slow down the experimenting process. A combination of code mistakes or/and bugs and the lack of libraries support makes these libraries impossible to use. In fact, there were some such libraries encountered in this work that are unreported, untested and thus did not provide any contribution to this research work. At the same time, some bugs were found in the presented libraries which were subsequently corrected with their authors' support.

4.4 Point Cloud Registration

The first part of point cloud registration is to separate as-built models from the environment. This is done manually in Meshlab. Figure 4.47 demonstrates the number of points in the as-built point clouds without environment in comparison

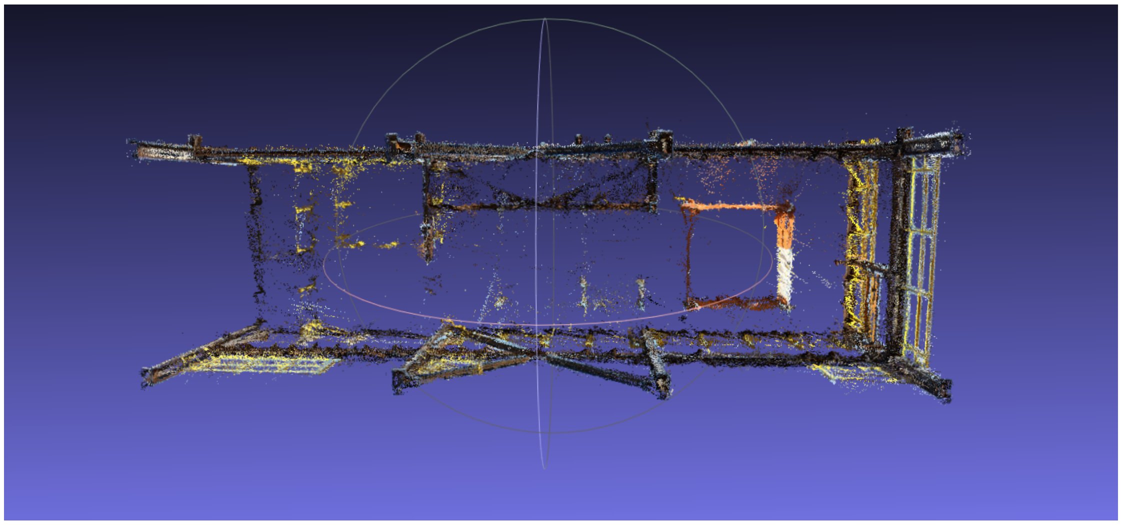


Figure 4.46: Bottom of the COLMAP as-built model

to initial and filtered point clouds. An example of the as-planned model cut from its environment is presented in Figure 4.48.

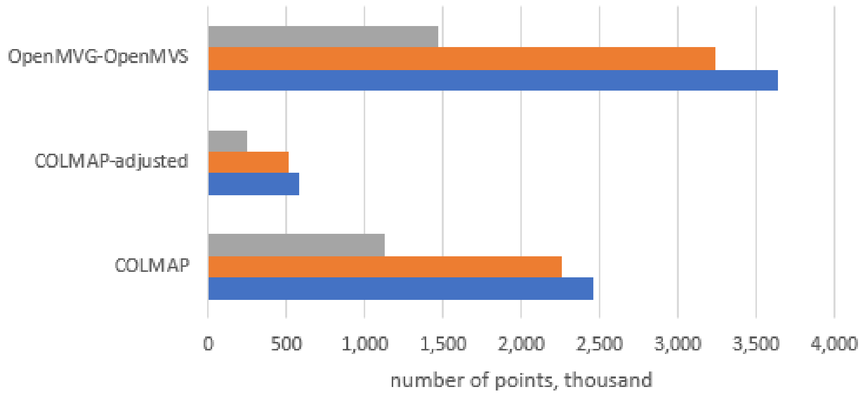


Figure 4.47: SfM-MVS algorithms number of reconstructed points. The gray color denotes the number of points in the as-built module point clouds excluding the environment. The orange color implicates the number of points in the filtered point clouds. The blue indicates the numbers of points in the original point clouds

The as-planned 3D model which represents a completed module is shown in Figure 4.49. Some examples of difference in the as-planned and the as-built point clouds location and scale are indicated in Figures 4.50 and 4.51. The difference in models size is caused by the algorithms' specific approaches to the reconstruction.

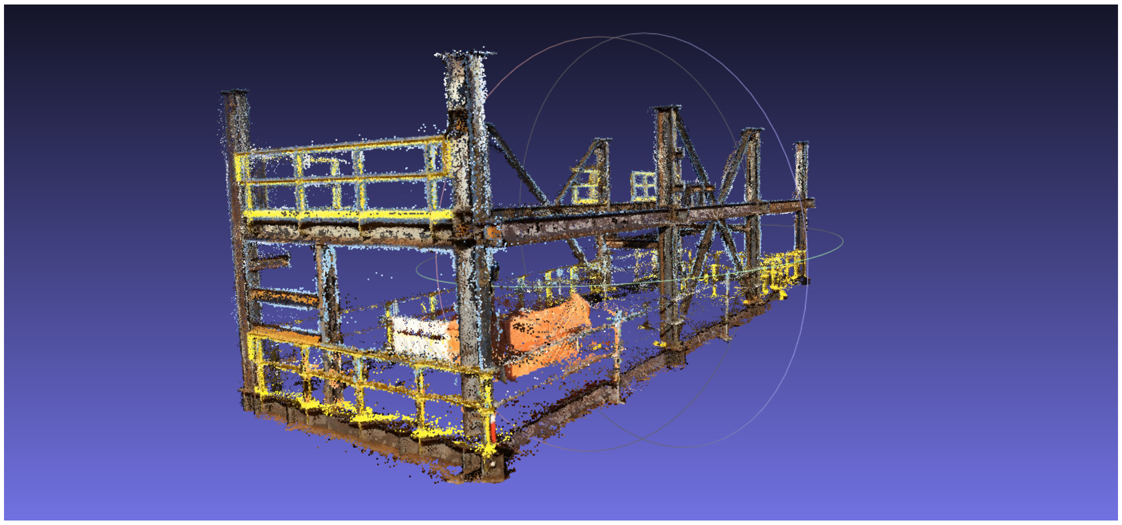


Figure 4.48: COLMAP point cloud cut from its environment

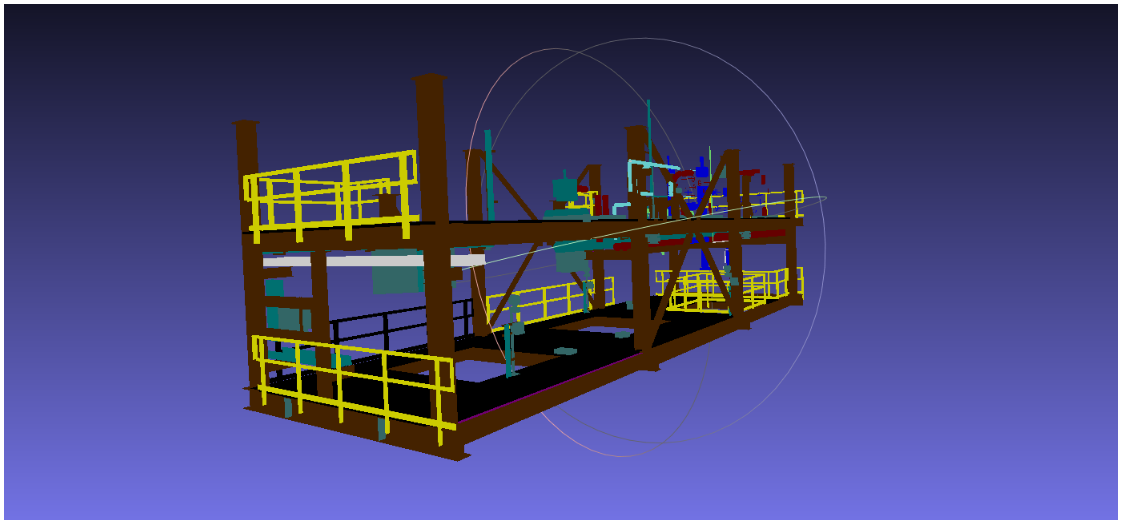


Figure 4.49: As-planed 3D model

Noticeably, the as-planned and as-build models in Figure 4.50 are close to each other, whereas the ones in Figure 4.51 are located fairly far away from one another. A good registration tool should successfully deal with these and other scenarios.

Having the as-planned point clouds ready, registration starts to be specific to the applicable registration approaches. This work proposes using ICP and CPD registration methods and they are described below.

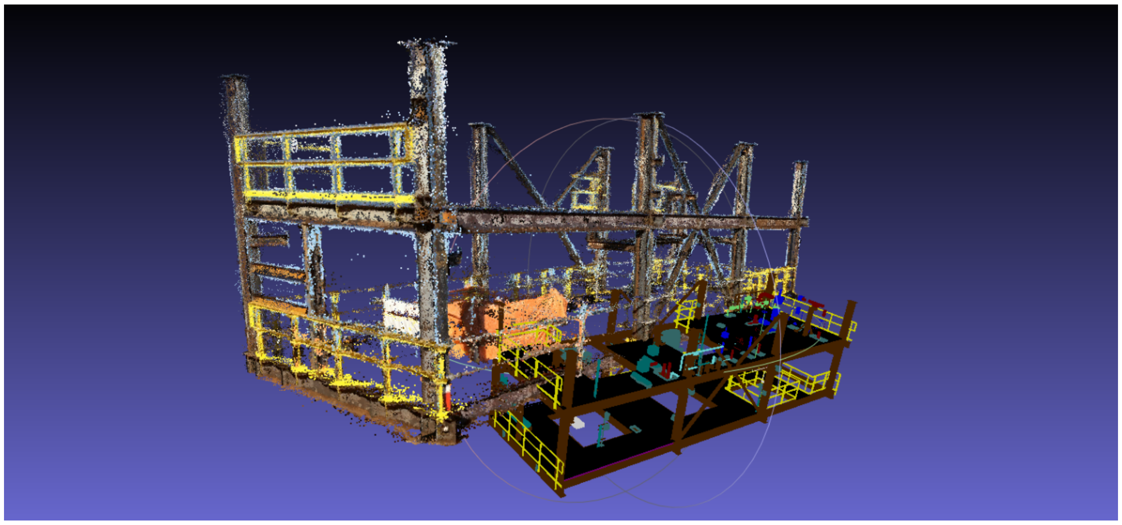


Figure 4.50: As-built COLMAP point cloud (large) and as-planned 3D model (small)

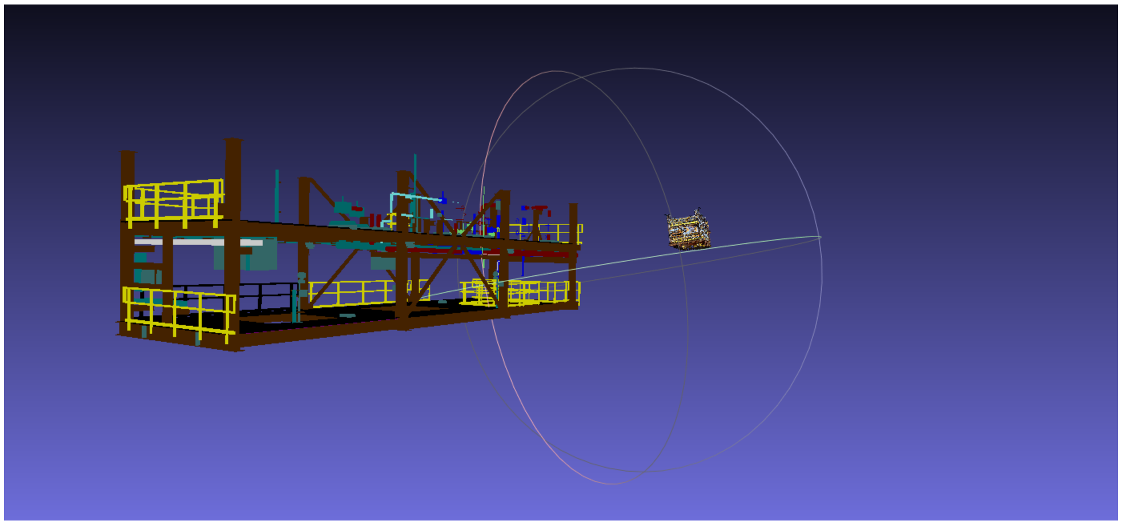


Figure 4.51: As-planned 3D model (large) and as-built OpenMVS-OpenMVG point cloud (small)

4.4.1 Iterative Closest Point

As was stated in Section 3.3, the required transformation for as-planned and as-built point cloud registration is similarity transform. The similarity transform is required because these point clouds are located in different coordinate systems and are of different scales. It was challenging to find an ICP algorithm for not only point clouds alignment but for finding a scale factor as well. There are a large number of ICP implementations available on the Internet; however, almost all of them only account for rigid transformation but not for scale. For example, Matlab has only rigid implementation of ICP.

Only one ICP implementation was found able to keep up with similarity transform, and it is libpointmatcher (Pomerleau (n.d.)) presented by Pomerleau et al. (2011). This library is not only capable of performing similarity transformation but has a number of various filters for transformation results improvements. Initially, the software was designed for use with robotics with computer vision and laser scanning point clouds (Pomerleau et al. (2015)) for such tasks as:

- search and rescue;
- power plant inspection;
- shoreline monitoring;
- autonomous driving.

Experimentally, the best performance of libpointmatcher is achieved when as-built and as-planned point clouds are of about the same size. Thus, it is desirable that the as-planned model is sampled to be of a similar number of points to as-built point cloud. The actual as-planned point cloud sampling is performed with CloudCompare software package (*CloudCompare - 3D point cloud and mesh processing software Open Source Project* (n.d.)) presented in Girardeau-Montaut (2011). Next, testing ICP transformation with different configurations revealed the best libpointmathcer performance on the industrial modules datasets and these configurations are presented in Appendix A Part 1c.

Ultimately, the libpoitmatcher produces the output in form of vtk monocolored reading point cloud transformed to the reference point cloud in vtk format. Additionally, it reports interim iteration point clouds and the final transformation matrix. In the context of this work, the main ICP output is the transformation matrix. The resulting transformation matrices are reported in this work as they allow us to see the difference in transformations proposed by the various methods.

The produced OpenMVG-OpenMVS point clouds ultimate transformation matrix is

$$\begin{bmatrix} -0.0136 & 0.0040 & 0.0077 & 0.1479 \\ 0.0058 & -0.0064 & 0.0136 & -0.0207 \\ 0.0064 & 0.0143 & 0.0039 & -1.1218 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

The COLMAP-adjusted point clouds transformation matrix is

$$\begin{bmatrix} 0.4616 & 0.0503 & -0.1734 & 0.3956 \\ -0.0515 & 0.4929 & 0.00595 & -0.3427 \\ 0.1730 & 0.0125 & 0.4643 & 0.1271 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The COLMAP point clouds transformation matrix is

$$\begin{bmatrix} 0.4658 & 0.0713 & -0.1355 & 0.3794 \\ -0.0713 & 0.4850 & 0.0105 & -0.3505 \\ 0.1356 & 0.0097 & 0.4711 & 0.1278 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The point clouds were aligned correctly only in the case shown in Figure 4.55.

The COLMAP point clouds transformation computation was terminated by the number of iterations which is 80 by default. The transformation likely needed so many iterations because of the large number of points. Large point clouds iterative transformation steps are usually smaller because the difference between the consecutive optimization function values is smaller relatively to the overall optimization function. The COLMAP transformation results are shown in Figure 4.53

OpenMVG-OpenMVS also did not work out correctly, which can be seen in Figure 4.54. The as-built model here transformed in almost a dot.

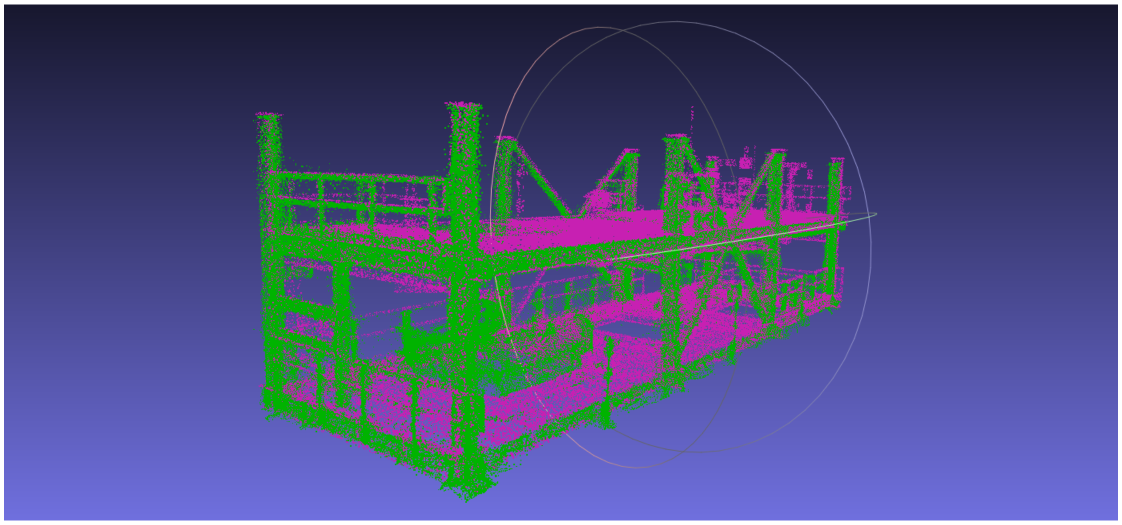


Figure 4.52: COLMAP-adjusted as-planned and as-built point clouds ICP registration. The as-planned model is pink and the as-built model is green

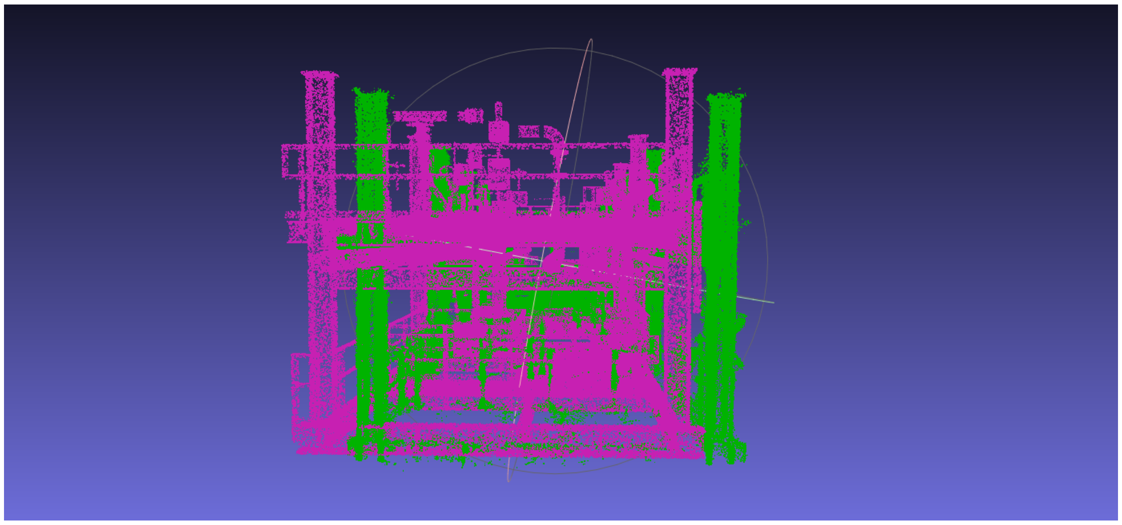


Figure 4.53: COLMAP as-planned and as-built point clouds ICP registration. The as-planned model is pink and the as-built model is green

The experiments with the COLMAP and OpenMVS-OpenMVG ICP transformed models were stopped at this point because of incorrect transformations.

4.4.2 Coherent Drift Point

CPD is a comparatively new approach to point cloud registration. The algorithm was initially introduced for serving medical imaging purposes in Myronenko and

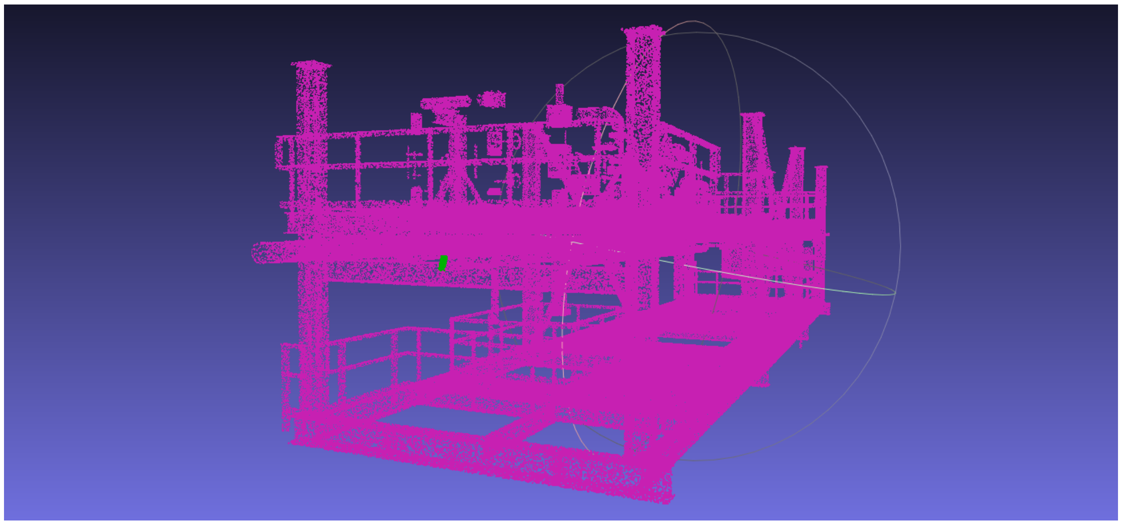


Figure 4.54: OpenMVS-OpenMVG as-planned and as-built point clouds ICP registration. The as-planned model is pink and the as-built model is green (a dot in the front of the green point cloud)

Song (2010). The algorithm proposes options of rigid, similarity and non-rigid registrations. The initial implementation was written in Matlab. Later on the algorithm was further developed for geographic mapping in Gadomski (2016). More specifically, this work is devoted to measuring glacier surface velocities with an airborne LIDAR. Because this work required comparison of vast amounts of points, the CPD implementation was rewritten in C++ for boosting its performance. One more computation speed advancement was achieved by partitioning glacier maps on the segments of 20,000 points and registering each of these pieces separately.

In this thesis work, the CPD implementation (Gadomski, n.d.) proposed in Gadomski (2016) is used. Experimenting with this implementation revealed that the algorithm performs well for a point cloud size of 8,000 - 20,000 points. Calculating transformations on larger point clouds increases the computation time significantly. At the same time, the proposed as-built models consist of 246,000 - 1,500,000 points. Attempts to divide as-planned models onto smaller pieces and computing transformation of each piece do not seem feasible because of the complex geometry of the industrial modules. Testing various approaches to CPD implementation indicated that as-planned and as-built point clouds can be successfully registered based on their sub-sampled models of about 9,000 points size. Because of the algorithm nature discussed in subsection 3.3.2, registration of the downsampled point clouds produces reasonable results. Ultimately, the transformation results are applied to the original point clouds.

Thus, the preparation step before proceeding with registration is subsample as-built and as-planned point clouds. This is done with Matlab sampling functions. The random 9,000 points are sampled from each point cloud and converted in the matrix format of 3 $\times$ number of points size with spreadsheet software. In fact, testing ICP on 9,000 points did not produce meaningful transformation estimates.

The CPD output is a transformation matrix. Additionally, CPD is able to produce transformed point cloud but as the CPD point clouds are downsampled they do not apply to this research. Finally, the estimated transformation is applied to the reading point cloud in Matlab.

In contrast to libpointmatcher, the CPD transformation results are reported in the format of rotation matrix, scale parameter and transformations vector. For example, the OpenMVG-OpenMVS reported results (as-planned to as-built models) are presented below.

OpenMVG-OpenMVS rotation matrix:

$$\begin{bmatrix} 0.9626 & 0.1507 & -0.2249 \\ -0.1467 & 0.9886 & 0.0347 \\ 0.2276 & -0.0004 & 0.9738 \end{bmatrix}$$

OpenMVG-OpenMVS translation vector:

$$\begin{bmatrix} -0.02628 \\ 0.13336 \\ -3.96266 \end{bmatrix}$$

OpenMVG-OpenMVS scale: 0.51548

For the sake of having comparable with the ICP transformation results, CPD estimated transformation is converted to a full transformation matrix.

OpenMVG-OpenMVS full transformation matrix (as-planned to as-built):

$$\begin{bmatrix} 0.4962 & -0.0756 & 0.1173 & -0.0263 \\ 0.0777 & 0.5096 & -0.0002 & 0.1334 \\ -0.1159 & 0.0179 & 0.5019 & -3.9627 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

During the experimentation it was noted, that the downscaling transformation is estimated with the lower scale than it should be almost all the time. The resulting downscaling transformation tolerances are not acceptable for further progress detection. At the same time, the upscaling transformation estimate is just fine. According to the author's opinion, this might be caused by an algorithm specific reaction on industrial model point clouds or an implementation bug. Whatever the problem is, for the sake of this work it was solved by using only the upscaling transformation. In case if the required transformation is downscaling, the upscaling transformation matrix is estimated and inverted. Notable, the inverted upscaling transformation matrix represents downscaling transformation. The experiments indicated good performance of the algorithm with such "adjustment."

OpenMVG-OpenMVS full transformation matrix (as-built to as-planned):

$$\begin{bmatrix} 1.8675 & 0.2924 & -0.4364 & -1.7191 \\ -0.2845 & 1.9178 & 0.0673 & 0.0035 \\ 0.4415 & -0.0008 & 1.8890 & 7.4973 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

COLMAP-adjusted full transformation matrix (as-built to as-planned):

$$\begin{bmatrix} 0.4607 & 0.0472 & -0.1774 & 0.3986 \\ -0.0513 & 0.4932 & -0.0022 & -0.3417 \\ 0.1762 & 0.0204 & 0.4631 & 0.1236 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

COLMAP full transformation matrix (as-built to as-planned):

$$\begin{bmatrix} 0.4620 & 0.0712 & -0.1599 & 0.3991 \\ -0.0714 & 0.4887 & 0.0113 & -0.3416 \\ 0.1598 & 0.0126 & 0.4673 & 0.1351 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

An example of the point clouds registration is shown in Figure 4.55.

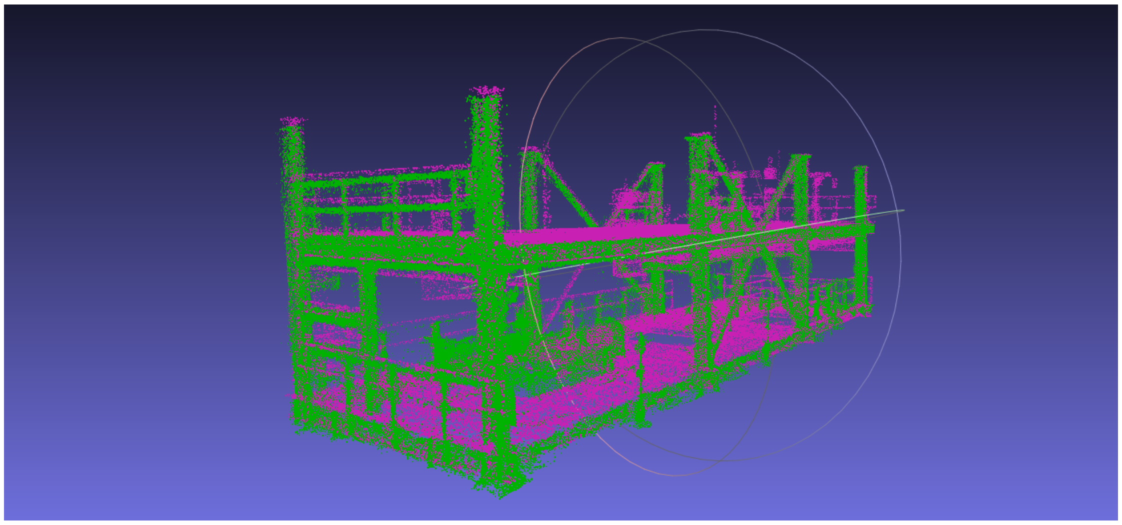


Figure 4.55: COLMAP-adjusted as-planned and as-built point clouds CPD registration. The as-planned model is pink and the as-built model is green

4.4.3 Registration Conclusion

Based on three proposed reconstructions, the CPD and libpointmatcher transformations were estimated for each model. Having the transformation matrices, the as-built point clouds are aligned with the as-planned models in Matlab. Four out of six transformations were successful. The transformations are considered to be successful when they are well aligned visually as it can be seen in Figures 4.52 and 4.55. The counterexamples of a successful transformation are the OpenMVS-OpenMVG ICP transformation (Figure 4.54) and the COLMAP ICP transformation (Figure 4.53). The computation time for each scenario is presented in Figure 4.56. The key properties of the presented approaches are presented in Table 4.2.

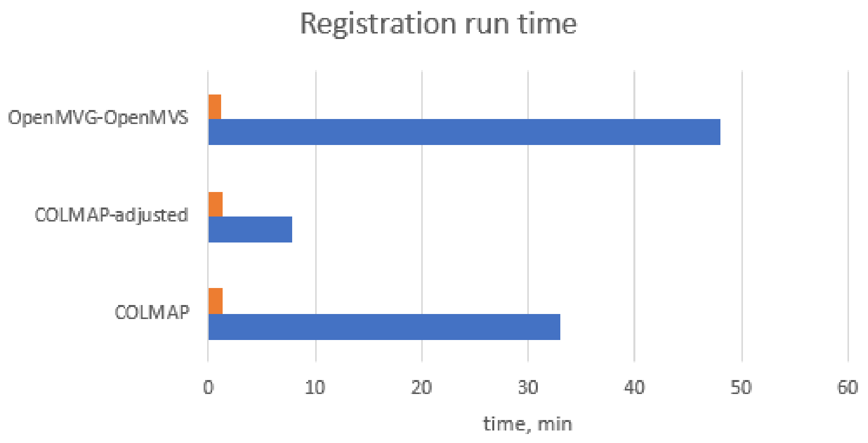


Figure 4.56: Registration run time. The CPD time is orange and the libpointmatcher is blue

Table 4.2: Fundamental characteristics of the presented reconstruction software

Properties	libpointmatcher	CPD
Algorithm	ICP	CPD
Type	open source	open source
Platform	Windows/Linux	Windows/Linux
Compilation required	yes	yes
Input format	vtk	csv
Embedded data filters	open source	open source
Computation time, min	Figure 4.56	
Number of points, thousand	Figure 4.47 (gray)	
User interface	no	no
Command line interface	yes	yes
Documentation	yes	yes

Challenges:

- ICP tendency to converge in local minimum. The OpenMVG-OpenMVS model ICP transformation demonstrated in Figure 4.54 is an example of ICP convergence in a local minimum. This theoretically can be dealt with by applying different data filters. However, the author did not find special algorithm parameters to make libpointmatcher produce correct transformation estimates for the tested scenarios;
- using proper ICP data filters. The COLMAP ICP transformation results shown in Figure 4.53 can also be potentially improved within the default number of iterations by applying more proper data filters.

4.5 Progress Identification

Having as-planned and as-built models aligned and in the same scale, the construction progress can be identified. The proposed approach employs Meshlab for computing the Hausdorff distance. At this stage, either a 3D model or a point cloud can be used as an as-planned model. In terms of this work, the as-planned point cloud is compared to the as-built point cloud. The output of the Hausdorff distance estimation is in the form of processed as-planned model color coded with the Hausdorff distances for each point. This research work proposes to use double color coding, as according to the authors opinion this clearly states the progress status of the as-planned model parts. In this fork the red-black color coding is proposed, where black indicates completed parts and red shows the parts that are to be completed. The Hausdorff distance computation takes less than a minute for all of the presented point clouds. Some key properties of the presented approach are indicated in Table 4.3. The example of the resulting model reported to the end user is shown in Figure 4.57.

Table 4.3: Fundamental characteristics of the presented progress identification software

Properties	Meshlab
Type	open source
Platform	Windows/Linux
Compilation required	no
Input format	ply
Computation time, min	< 1
Number of points, thousand	246 - 1,469
User interface	yes
Command line interface	yes
Documentation	yes

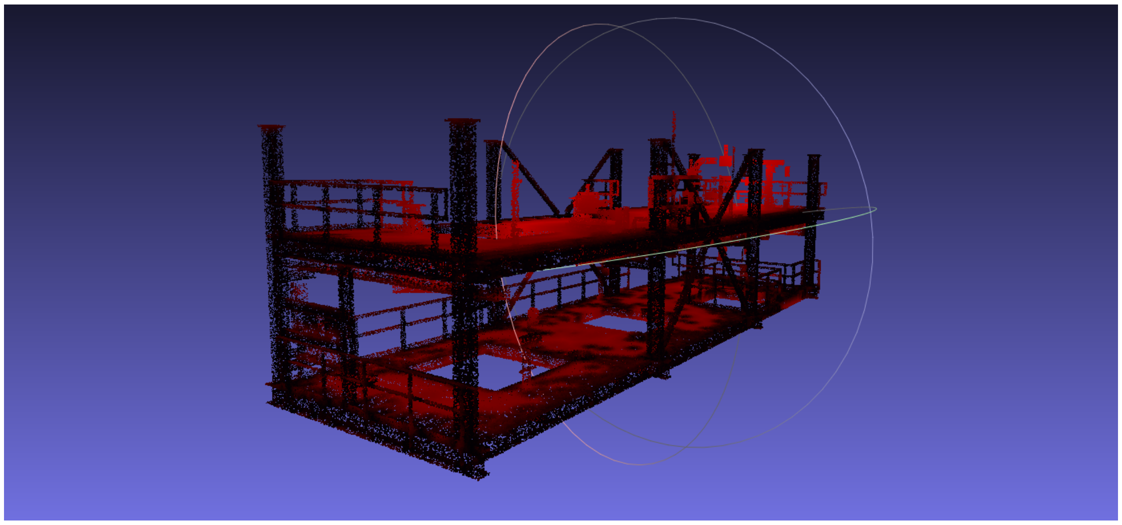


Figure 4.57: COLMAP progress model based on the Hausdorff distance. The black parts are in place and the red ones are not in place

For computation of the final results, the black points are considered to belong to already completed parts of the module, and the red points are considered to belong to unbuilt parts of the module. Thus, the ratio of the black points number to the overall number of points is calculated. This value is the progress completion percentage. The detected progress completion for each reconstruction registration approach is reported in this list:

- COLMAP-adjusted-ICP: 61.05%;
- COLMAP-adjusted-CPD: 61.87%;
- COLMAP-CPD: 63.50%;
- OpenMVS-OpenMVG-CPD: 62.61%.

The obtained results prove that the proposed frameworks are valid, because they individually reported almost the same results based on the same set of photographs. Furthermore, comparing the progress models to the input photographs shows legitimate completed parts identification.

Additionally, the obtained results might be presented visually in the form of RGB red channel histograms. The histograms for each of the models in this section are presented in Figures 4.58 - 4.61. All the histograms are skewed to the right, which means that that there is less work to be done than that which has been completed. This corresponds to the reported progress completion numbers.

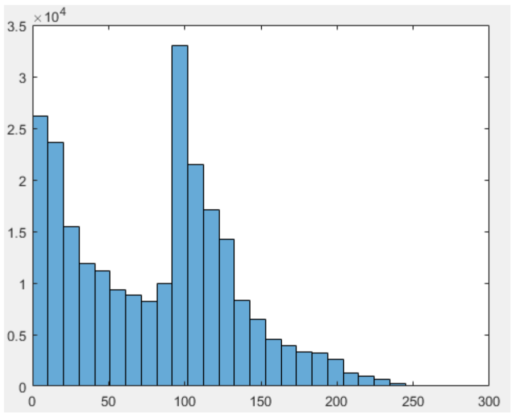


Figure 4.58: COLMAP-adjusted ICP transformed point cloud red channel histogram

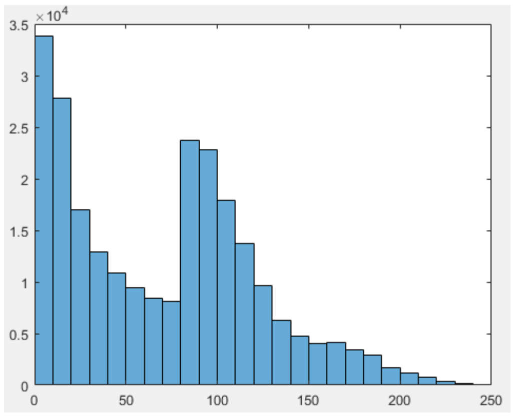


Figure 4.59: COLMAP-adjusted CPD transformed point cloud red channel histogram

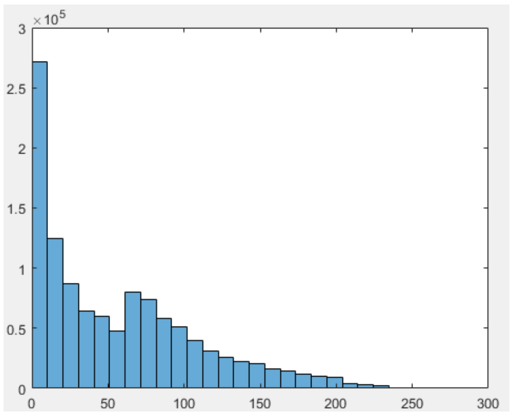


Figure 4.60: COLMAP CPD transformed point cloud red channel histogram

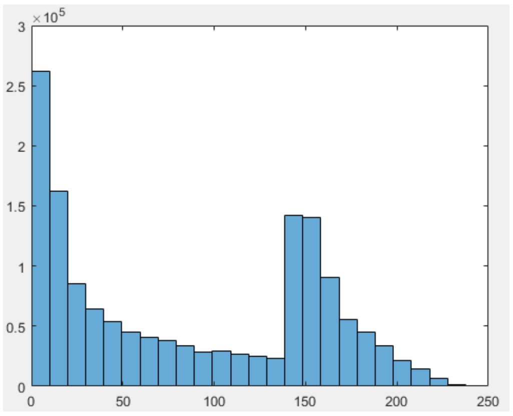


Figure 4.61: OpenMVS-OpenMVG CPD transformed point cloud red channel histogram

4.6 Progress Tracking Summary

The case study was conducted on automated progress tracking of an industrial module by applying top-notch computer vision algorithms. For the sake of different approaches comparability, a 138 photograph set of the industrial module taken by a rudimentary camera was chosen as initial data input. Next, the data set was passed to SfM, MVS, and registration algorithms implemented by different software packages. Thus, the progress tracking was performed based on various combinations of the cutting edge software libraries. The case study allowed finding the methods capable of progress tracking of complex geometry objects as an industrial module for example in congested area. It also helped to compare and test the combinations of different computer vision libraries and select the effective ones. Each software combination which led to the progress identification based on the initial dataset input is considered to be a progress tracking framework. The tested data frameworks along with the algorithms which did not lead to the progress tracking reports are presented in Figure 4.62. A full step-by-step computer vision progress tracking framework configuration and application guide is presented in Appendix A.

Four effective frameworks are proposed in this work. The progress result point clouds produced by these pipelines are very similar visually as well as the progress completion phase identified by all the proposed methods are in a close range (61.05% - 63.50%). These circumstances make the proposed frameworks almost indistinguishable in terms of reported results. However each pipeline has it's own specific properties which are summarized here:

- COLMAP-COLMAP-CPD is the approach which produces the best as-built point clouds in terms of accuracy and completeness. This framework is likely to be someone's choice in cases when the progress tracking of small details is required. However COLMAP-COLMAP-CPD computation time is about a couple of hours which makes it undesirable in cases where fast progress identification is required;

- OpenMVG-OpenMVS-CPD is the fastest progress tracking approach. The computation time of less than a quarter-hour makes it attractive to any construction project. At the same, OpenMVG-OpenMVS-CPD has the highest amount of noise. However, the case study progress identification results show that the OpenMVG-OpenMVS-CPD noise did not have much effect on the finial results;

- COLMAP-COLMAP-adjusted-CPD and COLMAP-COLMAP-adjusted-ICP are equally good intermediary frameworks between COLMAP-COLMAP-CPD and OpenMVG-OpenMVS-CPD. They demonstrated medium computation time as well as medium point completeness and accuracy.

To conclude, all the proposed frameworks demonstrated good performance, and the choice of employing one of them to a real construction project should be made considering their specific properties.

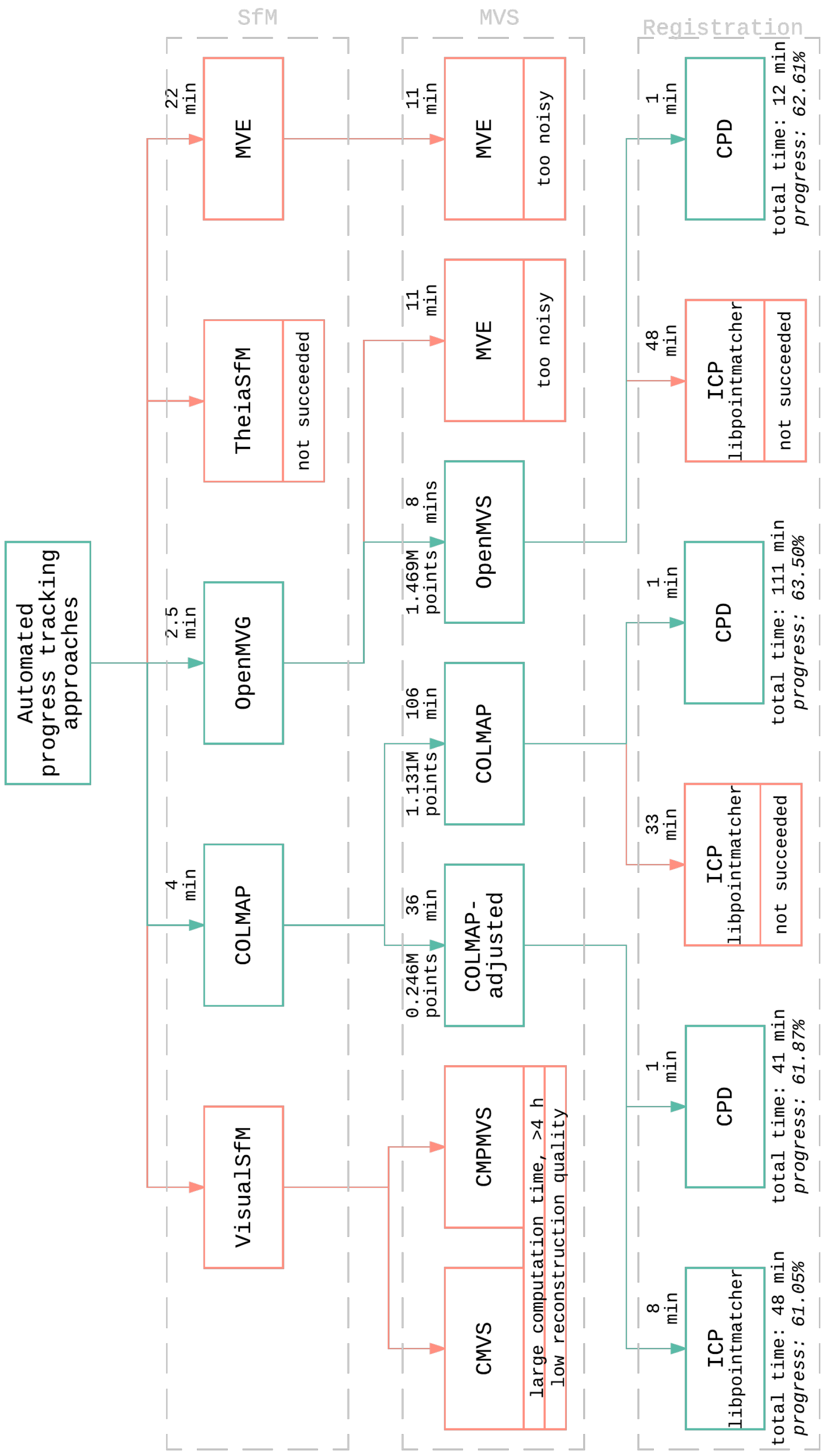


Figure 4.62: Progress tracking experiments schema. The green color demonstrates approaches finished to the final stage, the red color indicates incomplete approaches

Chapter 5

Conclusion

5.1 Proposed Frameworks Real Live Standing

The proposed frameworks for automated progress tracking are not completely new. They follow the logic of the existing progress tracking techniques proposed by Golparvar-Fard et al. (2010) and Golparvar-Fard et al. (2012). Although the existing tools are not available to the public and there are no ways to directly compare the new frameworks to the existing ones, there are reasons to think that the approaches proposed in this research outperform the previous ones:

1. Computation time: the OpenMVG-OpenMVS-CPD resulting time is 12 min on the 138 image dataset versus a few hours on the 160 image dataset reported by Golparvar-Fard et al. (2012), a few hours on unknown dataset size reported by Ahmed et al. (2012) and tree hours on the 75 image dataset reported by Skarlatos and Kiparissi (2012). In fact, SfM timing of the proposed algorithm for the whole industrial module reconstruction is 2.5 - 4 mins on the 138 image dataset versus 10 mins - 7 hrs on the 53 - 242 image datasets for the reconstruction of a columns or a masonry block reported by Golparvar-Fard et al. (2011). In addition, the proposed registration and progress identification techniques are believed to work faster than the existing ones because they are simple and straightforward.

2. Point cloud quality. The proposed pipelines are believed to outperform the existing approaches in:

   (a) SfM:

      i. COLMAP. Schonberger and Frahm (2016) reported COLMAP to improve the state of the art in terms of completeness, robustness, accuracy, and efficiency. COLMAP clearly surpasses such libraries as VisualSfM and Bundeler. Whereas, the approach proposed by Golparvar-Fard et al. (2012) is reported to be of similar or slightly better performance than VisualSfM (Karsch et al. (2014)) and the one proposed by Skarlatos and Kiparissi (2012) is based on Bundler;

      ii. OpenMVG. Moulon et al. (2013) reported OpenMVG to considerably exceed VisualSfM and Bundler in terms of accuracy. Hence, the proposed pipeline is likely to outrun the progress tracking approaches proposed by Golparvar-Fard et al. (2012) and Skarlatos and Kiparissi (2012).

   (b) MVS. The experimental results showed that both COLMAP and OpenMVS exceed CMVS in terms of point cloud quality. One example of this is that a large structural part of a module was missing in CMVS reconstruction (Figure 4.17), whereas the same piece was well reconstructed with COLMAP (Figure 4.46) and OpenMVS. Furthermore, Schöps et al. (n.d.) reported COLMAP outrunning PMVS and CMPMVS in terms of accuracy and completeness. Based on this, the proposed frameworks surpass the progress tracking approaches proposed by Golparvar-Fard et al. (2012) and Skarlatos and Kiparissi (2012), which are based on lagging libraries.

3. Registration automation. Almost all of the existing techniques require some manual steps for point clouds alignment such as manually defining the same points on the as-build and as-planned point clouds (Golparvar-Fard et al.

(2012)) or setting up color-coded targets before taking photos of a construction object (Ahmed et al. (2012)). Only one automated solution was found in the literature review (C. Kim et al. (2011)). The C. Kim et al. (2011) approach to registration is a sophisticated two steps PCA-ICP which is computationally intense. Furthermore, all the subsequent papers of these researchers do not report PCA-ICP registration to be a part of their methods. In contrast to existing approaches, this thesis work presents automated one step ICP as well as a novel one step super fast CPD registration approach.

4. Progress identification. The proposed Hausdorff progress identification technique is very simple. Because the Hausdorff distance demonstrated reliable results of progress identification it is believed to be at least of the same performance as the complicated Bayesian probabilistic model with SVM classier (Golparvar-Fard et al. (2010)) and graph progress identification (Braun et al. (2015)). Hence, the proposed technique advantages is that it is simple and straightforward.

5. Data acquisition. All the existing approaches are tested based on photographs obtained by professional cameras such as Nikon D80 (Golparvar-Fard et al., 2011) or Canon XSi 450D (Ahmed et al. (2012)), or even Bumblebee XB3 stereo vision system (Son and Kim (2010)). In contrast, the proposed in this research frameworks demonstrated efficient results based on the photographs from a rudimentary iPhone 5s.

6. Construction object. Contrasting with all the existing approaches, this research proposed frameworks were tested on an object of complex geometry on a congested construction site.

7. Software type. According to the literature review, the pipelines proposed in this thesis are only free and open source techniques for automated progress tracking. The first advantage of such software is that anyone can download

a product and test, use, check, and/or modify it for their special needs. The second advantage is that open source products are easily accessible for many construction companies and researchers without any considerable investment of money. This in turn makes the product to be more attractive to the public, which might consequently boost automated progress tracking real application and scientific development.

5.2 Contributions

The proposed frameworks for construction progress tracking are believed to outperform existing approaches in terms of progress identification time and quality. The following scientific contributions are achieved in this research:

1. Cutting-edge automated construction progress tracking approaches are improved in terms of time and quality by employing top-notch computer vision algorithms.

2. Algorithms for automatic registration have been incorporated into proposed frameworks. The presented ICP registration is the first ICP implementation proposed for the construction progress monitoring that can be used for similarity transformation on its own. Additionally, this work is the first to propose the novel CPD approach for use in the construction domain.

3. The proposed frameworks are the first comprehensive, free and open source automated construction progress tracking solution.

The industrial contribution of this research is that it provides construction companies with an effective tool for progress tracking. This tool is able to perform progress identification in 10-15 minutes which is almost “instantly” in comparison to existing approaches.

5.3 Limitations and Future Work

The proposed frameworks for automated progress tracking in construction still have limitations. Some of the limitations are caused by the nature of visual data and should be addressed on site, whereas the other limitations may be solved trhough future research endeavors. The following suggestions are made for dealing with the current automated progress tracking framework restraints:

1. Visual object detection: the automated project tracking techniques are only able to detect objects that are visible in photographs. This is also true for construction workers, as humans are generally not able to make a conclusion about the status of an object if they cannot see it. In the case where a portion of an object of interest is hidden, there are two possible solution to detect it:(1) move a camera to a location where the object is visible or (2) integrate a work breakdown structure as a part of progress tracking. Having a WBS as a part of progress tracking framework will allow for the detection of hidden object progress status by their position in WBS. In other words, if work that proceeds installation of the hidden object is complete, then the hidden objects are assumed to be installed.

2. Unnecessary environment reconstruction: the visual approach to construction object reconstruction implicates reconstruction of not only the objects of interest but also the environment around them. Consequently, the environment outside the object of interest must be manually removed. The problem of focusing on construction objects may be addressed by employing a drone. A drone has a much greater range, and is, therefore, more likely to place itself at a more appropriate angle and distance, allowing it to focus only on objects of interest. This advancement will potentially decrease the amount of unwanted environment, the content of noise, as well as improve reconstruction quality.

3. Reconstruction accuracy: the proposed frameworks may potentially be used to verify whether or not proper elements are installed and if they are placed

correctly. The main factors affecting accuracy are camera resolution and distance between the camera and the object of interest. Specifically, the greater the resolution of photograph and the closer the camera to the object of interest, the more precise the reconstruction. Notably, determination of measurement accuracy was not required to conduct the proposed work and is, therefore, beyond the scope of this project.

4. Progress measurement: the proposed frameworks determine progress as a percentage of completed area relative to the overall area to be completed. Notably, percentage-based completion measurements may not be the most representative progress metric. For example, setting up a large element may not represent a large amount of progress, as it may be a simple procedure and may not require a lot of time and equipment. At the same time, erecting a special small pipeline may represent a small area of progress but may account for a considerable amount of progress with respect to labour-hours and time. Furthermore, certain temporary work, such as assembly and disassembly of scaffolding, should also be accounted for. Thus, the proposed frameworks progress identification logic should be further developed. Potentially, it may be improved by utilizing a 4D as-planned model on the planning side and classifying as built model parts with machine learning techniques on the construction side. This development will help to tie as-built construction elements directly with project schedule, subsequently allowing for detailed progress measurement.

Additionally automated progress tracking research can be conducted in the following directions:

- planning and simulation of construction operations associated with industrial modular projects: the current method of planning construction of an industrial modular plant involves experienced people estimating the amount of work from 3D models based on personal experience. This estimation type is subjective, prone to error and time-consuming. However, it is possible to create a tool not only for automated progress tracking but for automated estimation. The first step for this tool development is learning from automated progress tracking results. Then, data are collected and passed to a simulation engine, which could rapidly produce a probabilistic estimate based on actual construction performance data. These results are objective, comparatively precise and rapidly obtainable;

- framework algorithm improvement: this is essential for long-term automated progress tracking development. This development branch should include reporting progress tracking results to the algorithms authors, improving the algorithms performance and sharing the results with the research community. Simultaneously, it is worth examining new computer vision algorithms that may potentially improve the automated progress monitoring framework;

- hardware: the proposed progress tracking frameworks are sensitive to hardware in terms of time. The computer used for this research is equipped with an Nvidia GTX Titan X GPU which is expensive compared to general gaming graphics cards. Thus, someone on a tight budget can test computation timing on, for example, Nvidia GTX 1070 or Nvidia GTX 1060. At the same time, someone with good budget looking for the fastest reconstruction time can try performing calculations on multiple Nvidia GTX 1080, Nvidia GTX 1080 ti, or Nvidia GTX Titan X connected by an SLI bridge.

Appendix A

Computer Vision Progress Tracking Framework Configuration and Application Guide

A.1 Software Installation

This guide assumes that the software installation is performed on a clean install of Ubuntu 16.04 with a CUDA 7.0 or higher capable GPU*.

* GPU CUDA compatibility can be verified at `https://developer.nvidia.com/cuda-gpus` web-page.

1. Download and Install CUDA Toolkit 8.0:

   (a) install the latest Nvidia graphics card driver;

   (b) install CUDA Toolkit.

2. Install COLMAP:

(a) checkout source code:

```
$ git clone https://github.com/colmap/colmap
```

(b) install dependencies from default Ubuntu repositories:

```
$ sudo apt-get install \
                        cmake \
                        build-essential \
                        libboost-all-dev \
                        libeigen3-dev \
                        libsuitesparse-dev \
                        libfreeimage-dev \
                        libgoogle-glog-dev \
                        libgflags-dev \
                        libglew-dev \
                        freeglut3-dev \
                        qt5-default \
                        libxmu-dev \
                        libxi-dev
```

(c) install Eigen3:

i. download the latest version form http://eigen.tuxfamily.org/index.php?title=Main_Page;

ii. compile Eigen3 according to https://eigen.tuxfamily.org/dox/GettingStarted.html.

(d) install Ceres Solver:

```
$ sudo apt-get install libatlas-base-dev libsuitesparse-dev
$ git clone https://ceres-solver.googlesource.com/ceres-solver
$ cd ceres-solver
$ mkdir build
$ cd build
$ cmake ..
$ make -j
$ sudo make install
```

(e) compile COLMAP:

```
$ cd path/to/colmap
$ mkdir build
$ cd build
$ cmake ..
$ make -j4
```

3. Install OpenMVG:

```
$ git clone -b develop https://github.com/openMVG/openMVG.git
$ cd openMVG
$ git submodule init
$ git submodule update
$ mkdir openMVG_Build $$ cd openMVG_Build
$ cmake -DCMAKE_BUILD_TYPE=RELEASE -DOpenMVG_BUILD_TESTS=ON
 -DOpenMVG_BUILD_EXAMPLES=ON . ../openMVG/src/
$ make -j4
```

4. Install OpenMVS:

   (a) install OpenCV:

```
$ sudo apt-get -y install libopencv-dev
```

(b) install CGAL:

```
$ sudo apt-get -y install libcgal-dev libcgal-qt5-dev
```

(c) install VCGLib:

```
$ git clone https://github.com/cdcseacave/VCG.git vcglib
```

(d) compile OpenMVS:

```
$ git clone https://github.com/cdcseacave/openMVS
$ mkdir openMVS_build && cd openMVS_build
$ cmake . ../openMVS -DCMAKE_BUILD_TYPE=Release -DVCG_DIR="$main_p
$ make
```

5. Install CPD:

(a) install jsoncpp:

```
$ sudo apt-get install libjsoncpp-dev
```

(b) compile CPD:

```
$ git clone https://github.com/gadomski/cpd
$ cd cpd
$ git checkout benchmark
$ mkdir build
$ cd build
$ cmake .. -DCMAKE_BUILD_TYPE=Release -DWITH_JSONCPP=ON
$ make
$ sudo make install
$ cd ../examples
$ mkdir build
$ cd build
$ cmake ..
$ make
```

6. Install libpointmatcher:

   (a) install libnabo:

   ```
   $ git clone https://github.com/ethz-asl/libnabo
   $ cd libnabo
   $ mkdir build $$ cd build
   $ cmake ..
   $ make
   $ sudo make install
   ```

   (b) compile libpointmatcher:

   ```
   $ git clone https://github.com/ethz-asl/libpointmatcher
   $ cd libpointmatcher
   $ mkdir build && cd build
   $ cmake ..
   $ make
   $ sudo make install
   ```

7. Install Meshlab:

```
$ sudo apt-get install meshlab
```

8. Install ParaView:

```
$ sudo apt-get install paraview
```

9. Install Matlab.

10. Install CloudCompare:

```
$ sudo snap install cloudcompare
```

A.2 Software Versions

Software versions used at the time of writing this guide are as follows:

```
CUDA ToolKit 8.0
COLMAP 3.1
Cmake 3.5.1
Build-essential 12.1
Libboost-all-dev 1.58
Libsuitesparse-dev 1:4.4.6-1
Libfreeimage-dev 3.17.0
Libgoogle-glog-dev 0.3.4-0.1
Libgflags-dev 2.1.2-3
Libglew-dev 1.13.0-2
Freeglut3-dev 2.8.1-2
Qt5-default 5.5.1
Libxmu-dev 2:1.1.2-2
```

```
Libxi-dev 2:1.7.6-1
Eigen 3.3.4
Ceres Solver 1.13
OpenMVG 1.2
OpenCV 3.2.0
CGAL 4.10
VCGLib 1.0.1
OpenMVS 0.7
Jsoncpp 1.8.1
CPD 0.5.1
Libnabo 1.0.6
Libpointmatcher 1.2.3
MeshLab 2016
Paraview 5.4.1
MATLAB R2017a
Cloudcompare 2.8.1
```

A.3 Software Application

The overall progress tracking pipeline is demonstrated for a reader convenience in Figure A.1. The demonstrated pipeline is imaginary grouped into three components: reconstruction, registration and progress detection. The underlying software application steps are presented below in the same manner.

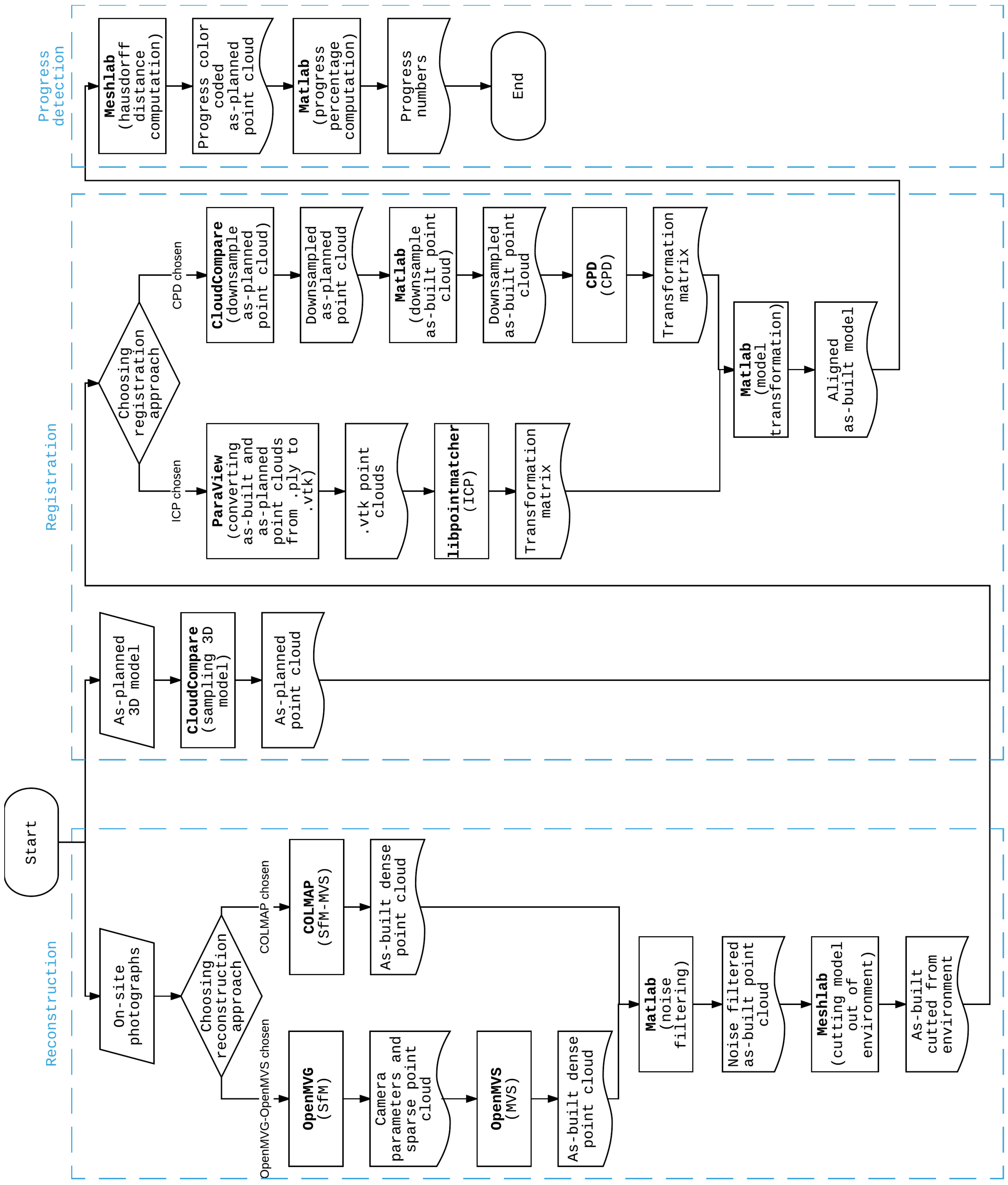

Start
Reconstruction
On-site photographs
Choosing reconstruction approach
COLMAP chosen
COLMAP (SfM-MVS)
As-built dense point cloud
OpenMVG-OpenMVS chosen
OpenMVG (SfM)
Camera parameters and sparse point cloud
OpenMVS (MVS)
As-built dense point cloud
Matlab (noise filtering)
Noise filtered as-built point cloud
Meshlab (cutting model out of environment)
As-built cutted from environment
Registration
As-planned 3D model
CloudCompare (sampling 3D model)
As-planned point cloud
Choosing registration approach
ICP chosen
ParaView (converting as-built and as-planned point clouds from .ply to .vtk)
.vtk point clouds
libpointmatcher (ICP)
Transformation matrix
CPD chosen
CloudCompare (downsample as-planned point cloud)
Downsampled as-planned point cloud
Matlab (downsample as-built point cloud)
Downsampled as-built point cloud
CPD (CPD)
Transformation matrix
Matlab (model transformation)
Aligned as-built model
Progress detection
Meshlab (hausdorff distance computation)
Progress color coded as-planned point cloud
Matlab (progress percentage computation)
Progress numbers
End

Figure A.1: Progress tracking software application schema for the proposed frameworks

The further steps assume that on-site photographs are stored in the following folder structure:

```
/path/to/project/...
        +-- images
        |    +-- image1.jpg
        |    +-- image2.jpg
        |    +-- ...
        |    +-- imageN.jpg
```

A.3.1 Reconstruction

1. OpenMVG-OpenMVS:

   (a) run OpenMVG commands:

```
* run the following commands from the openMVG_Build
linux release directory assuming the project path
$PROJECT_PATH=/path/to/project

$ ./openMVG_main_SfMInit_ImageListing -i $PROJECT_PATH/
images -d $path/to/openMVG/src/openMVG/exif/sensor_
width_database/sensor_width_camera_database.txt -o
$PROJECT_PATH/matches -f 1649

$ ./openMVG_main_ComputeFeatures -i $PROJECT_PATH/
matches/sfm_data.json -o $PROJECT_PATH/matches

$ ./openMVG_main_ComputeMatches -i $PROJECT_PATH/
matches/sfm_data.json -o $PROJECT_PATH/matches -g e

$ ./openMVG_main_GlobalSfM -i $PROJECT_PATH/matches/
sfm_data.json -m $PROJECT_PATH/matches -o
```

```
$PROJECT_PATH/out_Global_Reconstruction/

$ ./openMVG_main_ComputeSfM_DataColor -i $PROJECT_PATH/
out_Global_Reconstruction/sfm_data.bin -o $PROJECT_
PATH/out_Global_Reconstruction/sfm_data_color.ply

$ ./openMVG_main_openMVG2MVE2 -i $PROJECT_PATH/
out_Global_Reconstruction/sfm_data.bin -o $PROJECT_
PATH/out_Global_Reconstruction/

$ ./openMVG_main_openMVG2openMVS -i $PROJECT_PATH/
out_Global_Reconstruction/sfm_data.bin -o
$PROJECT_PATH/out_Global_Reconstruction/openMVS/
scene.mvs

$ ./openMVG_main_openMVG2openMVS -i $PROJECT_PATH/
out_Global_Reconstruction/sfm_data.bin -d $PROJECT_
PATH/out_Global_Reconstruction/openMVS/ -o $PROJECT_
PATH/out_Global_Reconstruction/openMVS/scene.mvs
```

(b) run OpenMVS commands:

```
* run the following commands from the OpenMVS_build bin
directory

$ cd path/to/openMVS_build/bin

$ ./DensifyPointCloud -i $PROJECT_PATH/out_Global_
Reconstruction/openMVS/scene.mvs -w $PROJECT_PATH/
```

```
    out_Global_Reconstruction/openMVS/ -o $PROJECT_PATH/
    out_Global_Reconstruction/openMVS/point-cloud-
    densified-openMVS.ply
```

2. run COLMAP commands:

```
* run the following commands from the COLMAP build
directory assuming the project path is
$PROJECT_PATH=/path/to/project

$ ./src/exe/feature_extractor \
   --database_path $PROJECT_PATH/database.db \
   --image_path $PROJECT_PATH/images

$ ./src/exe/exhaustive_matcher \
   --database_path $PROJECT_PATH/database.db

$ mkdir $PROJECT_PATH/sparse

$ ./src/exe/mapper \
    --database_path $PROJECT_PATH/database.db \
    --image_path $PROJECT_PATH/images \
    --export_path $PROJECT_PATH/sparse

$ mkdir $PROJECT_PATH/dense

$ ./src/exe/image_undistorter \
    --image_path $PROJECT_PATH/images \
    --input_path $PROJECT_PATH/sparse/0 \
    --output_path $PROJECT_PATH/dense \
    --output_type COLMAP \
    --max_image_size 800

$ ./src/exe/dense_stereo \
    --workspace_path $PROJECT_PATH/dense \
    --workspace_format COLMAP \
    --DenseStereo.max_image_size 0 \
    --DenseStereo.geom_consistency true

$ ./src/exe/dense_fuser \
    --workspace_path $PROJECT_PATH/dense \
    --workspace_format COLMAP \
    --input_type geometric \
    --output_path $PROJECT_PATH/dense/point-cloud.ply
```

3. Filtering and Cutting

   (a) run Noise filtering Matlab code:

```
* run the following code on the desired dense reconstructed
model

% Reading as-built point cloud
ptCloud = pcread('your_reconstructed_point_cloud_name.ply');
%pcshow(ptCloud);

% Filtering blue noise
ptCloudRGB = ptCloud.Color;
ptCloudRGB_int = int64(ptCloudRGB);
```

```
sortB = [ptCloudRGB_int(:,3), (ptCloudRGB_int(:,1)+
ptCloudRGB_int(:,2))/2, [1:numel(ptCloudRGB_int(:,1))]' ];
blueS = [  ptCloudRGB_int(:,3) - max( ptCloudRGB_int(:,2),
ptCloudRGB_int(:,1) ) , zeros(numel(ptCloudRGB_int(:,1)), 1) ,
[1:numel(ptCloudRGB_int(:,1))]' ];
% Blue color histogram
figure; histogram(blueS(:,1))
% Setting up threshold
blueC = blueS ( find( blueS(:,1) > 35 ),: );

% Forming blue outlier point cloud
ptOUT = pointCloud(ptCloud.Location(blueC(:,3),:));
ptOUT.Color = ptCloud.Color( blueC(:,3),: );
%pcshow(ptOUT);

% Forming filtered inlier point cloud
blueOK = sortB;
blueOK(blueC(:,3),3) = -5;
blueOK = blueOK ( find( blueOK(:,3) >= 0 ),: );
ptIN = pointCloud(ptCloud.Location(blueOK(:,3),:));
ptIN.Color = ptCloud.Color( blueOK(:,3),: );
%pcshow(ptIN);

% Saving inlier and outlier point clouds
pcwrite(ptIN,'your_filtered_point_cloud_name.ply')
pcwrite(ptOUT,'your_noise_point_cloud_name.ply')
```

(b) Cutting model out of environment (Meshlab):

i. launch MeshLab;

ii. click Import Mesh from the toolbar and select the filtered point cloud .ply file;

iii. remove unwanted points from the point cloud:

A. click the Select Faces in a rectangular region icon and select points on the point cloud which are desired to be removed;

B. press the Delete key.

iv. save the file as point-cloud-clean.ply in ASCII format.

A.3.2 Registration

For the following steps, ensure an as-planned 3D mesh model availability. For the purpose of this tutorial, the complete 3D model will be referred as as-planned.ply.

1. run libpointmatcher (ICP) transformation:

   (a) sample the as-planned model:

      i. run CloudCompare from the corresponding directory:

```
$ ./cloudcompare.CloudCompare
```

      ii. select File →Open, select point-cloud-clean.ply, hit Apply;

      iii. record the number of points in Properties;

      iv. select File →Open, select as-planned.ply, hit Apply;

      v. select the Sample points on a mesh icon from above, enter the recored number of points from the previous step, under Points Number;

      vi. save the resulting Mesh.sampled entity as-planned-pt.ply in ASCII format.

   (b) convert point clouds from .ply files to .vtk:

      i. open the as-planned-pt.ply file in MeshLab click Export Mesh;

ii. uncheck Binary encoding, click OK;

iii. perform the following steps for both as-planned-pt.ply and point-cloud-clean.ply files:

A. open file in ParaView;

B. click Apply;

C. File →Save Data;

D. save as .vtk and click OK.

(c) create a configuration file for libpointmatcher called icp_cfg.yaml:

```
readingDataPointsFilters:
  - RandomSamplingDataPointsFilter



referenceDataPointsFilters:
  - SamplingSurfaceNormalDataPointsFilter:
      knn: 10

matcher:
  KDTreeMatcher:
    knn: 1

outlierFilters:
  - TrimmedDistOutlierFilter:
      ratio: 0.75

errorMinimizer:
  PointToPointSimilarityErrorMinimizer
```

```
transformationCheckers:
  - CounterTransformationChecker:
      maxIterationCount: 80
  - DifferentialTransformationChecker:
      minDiffRotErr: 0.001
      minDiffTransErr: 0.001
      smoothLength: 4

inspector:
 VTKFileInspector:
     baseFileName: pointmatcher-run1
     dumpPerfOnExit: 0
     dumpStats: 0
     dumpIterationInfo: 0
     dumpDataLinks: 1
     dumpReading: 1
     dumpReference: 1
```

(d) run libpointmatcher:

   i. run the following command:

```
$ path/to/libpointmatcher/build/examples/pmicp -v --config
icp_cfg.yaml as-planned-pt.vtk point-cloud-clean.vtk
```

   ii. record the resulting transformation matrix;

   iii. visually check overlap of the resulting transformed .vtk models in ParaView.

(e) perform as-built model transformation based on the libpointmatcher resulting transformation matrix in Matlab:

```
clearvars;
pt_target = pcread('as-planned-pt.ply');
pt = pcread('point-cloud-clean.ply');

T_full = [*libpointmatcher resulting transformation
matrix*]

pt_transformed_Location_full = T_full * cat( 1,
pt.Location', ones ( size( pt.Location, 1 ),1 )' );
pt_transformed_full = pointCloud( pt_transformed_
Location_full(1:3, : )' );

% Visual comparison of the as-planned
% and transformed as-built models
pcshowpair(pt_target,pt_transformed_full);

% Assigning original colors and saving
% the transformed point cloud
pt_transformed_full.Color = pt.Color;
%pcshow(pt_transformed_full);
pcwrite(pt_transformed_full,'point-cloud-clean_ICP-
transformed.ply');
```

2. run CPD transformation:

   (a) sample the as-planned model in CloudCompare:

      i. follow step 1a and create a full point cloud model if it is not done yet;

      ii. follow step 1a and create a downsampled 9000 points cloud;

      iii. save the resulting point cloud as as-planned-pt-9K.ply.

(b) sample the as-built model in Matlab:

```
pt = pcread('point-cloud-clean.ply');
pcshow(pt);

pt_target = pcread('as-planned-pt-9K.ply');
pcshow(pt_target);

pt_downsample = pcdownsample( pt,'random',round(
pt_target.Count/pt.Count,3) )
pt_downsample.Count
pcshow(pt_downsample);

pcwrite(pt_downsample,'point-cloud-clean-9K.ply');
```

(c) converting the as-built and as-planned models in matrix format suitable for CPD with LibreOffice Calc:

i. open a .ply file in LibreOffice Calc;

ii. on the Import Dialog, under Separator Options, check Space click OK;

iii. delete rows 1-17;

iv. delete columns D-J;

v. select File →Save As →Edit Filter Settings;

vi. change the Field delimiter and Text delimiter fields to a single space;

vii. click OK.

(d) run CPD commands:

```
$ cd path/to/cpd/examples/build
$ ./cpd-rigid point-cloud-clean-9K.csv as-planned-
pt-9K.csv
```

(e) record the obtained rotation matrix, the translation vector and the scale;

(f) transform the as-built point cloud according to the output recorded in the previous step in Matlab:

```
clearvars;

pt_target = pcread('as-planned-pt.ply');
%pcshow(pt_target);
pt = pcread('point-cloud-clean.ply');
%pcshow(pt;

scale = *CPD resulting scale*
T_rigid_rotation = [*CPD resulting rotation
matrix*]
T_rigid_translation = [*CPD resulting translation
vector*]'

% Inverse transformation
T_full = cat( 1, cat( 2, scale*T_rigid_rotation',
T_rigid_translation), [0 0 0 1] )
T_full = inv (T_full)
pt_transformed_Location_full = T_full * cat( 1,
pt.Location', ones ( size( pt.Location, 1 ),1 )' );
pt_transformed_full = pointCloud( pt_transformed_
Location_full(1:3, : )' );
```

```
% Visual comparison of the as-planned
% and transformed as-built models
pcshowpair(pt_target,pt_transformed_full);

% Assigning original colors and saving
% the transformed point cloud
pt_transformed_full.Color = pt.Color;
%pcshow(pt_transformed_full);
pcwrite(pt_transformed_full,'point-cloud-clean_CPD-
transformed.ply');
```

A.3.3 Progress identification

1. Compute Hausdorff distance in Meshlab:
   (a) open as-planned-pt.ply and either point-cloud-clean_ICP-transformed.ply or point-cloud-clean_CPD-transformed.ply, depending on the chosen registration method;
   (b) click Show Layer Dialog;
   (c) select the as-planned-pt layer;
   (d) select Filters →Sampling →Hausdorff Distance →Apply;
   (e) view only the as-planned-pt layer;
   (f) select Quality Mapper;
   (g) in the Quality Mapper Dialog, under Preset Ramps, select Red Scale;
   (h) Click Apply (at this stage color distribution might be corrected if required);
   (i) select Export Mesh As and save the point cloud as final.ply (this is a resulting visual progress tracking model).

2. Compute project completion percentage in Matlab:

```
pt = pcread('final.ply');

% Red color distribution histogram
histogram(pt.Color(:,1),25)

% Choosing threshold
threshhold = *chosen THRESHOLD*

% Progress percentage calculation
pt_progress = sum(pt.Color(:,1)<threshhold)/length(
pt.Color(:,1))
```

References

Ahmed, M., Haas, C., & Haas, R. (2012). Using digital photogrammetry for pipeworks progress tracking 1 1 this paper is one of a selection of papers in this special issue on construction engineering and management. *Canadian Journal of Civil Engineering*, *39*(9), 1062–1071.

Bae, H., Golparvar-Fard, M., & White, J. (2014). Rapid image-based localization using clustered 3d point cloud models with geo-location data for aec/fm mobile augmented reality applications. In *Computing in civil and building engineering (2014)* (pp. 841–849).

Bae, H., Walker, M., White, J., Pan, Y., Sun, Y., & Golparvar-Fard, M. (2016). Fast and scalable structure-from-motion based localization for high-precision mobile augmented reality systems. *mUX: The Journal of Mobile User Experience*, *5*(1), 1–21.

Baeza-Yates, R., Ribeiro-Neto, B., et al. (1999). *Modern information retrieval* (Vol. 463). ACM press New York.

Bailer, C., Finckh, M., & Lensch, H. (2012). Scale robust multi view stereo. *Computer Vision–ECCV 2012*, 398–411.

Beder, C., & Steffen, R. (2006). Determining an initial image pair for fixing the scale of a 3d reconstruction from an image sequence. In *Joint pattern recognition symposium* (pp. 657–666).

Beneder, C., Fuechsel, F., Krause, T., Kuhn, A., & Mueller, M. (2008). The role of 3d fusion imaging in sentinel lymphadenectomy for vulvar cancer. *Gynecologic oncology*, *109*(1), 76–80.

Besl, P. J., & McKay, N. D. (1992). Method for registration of 3-d shapes. In *Robotics-dl tentative* (pp. 586–606).

Beucher, S., et al. (1992). The watershed transformation applied to image segmentation. *SCANNING MICROSCOPY-SUPPLEMENT-*, 299–299.

Biswas, H. K., Bosché, F., & Suna, M. (2015). Planning for scanning using building information models: a novel approach with occlusion handling. In *Isarc. proceedings of the international symposium on automation and robotics in construction* (Vol. 32, p. 1).

Bosché, F. (2010). Automated recognition of 3d cad model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction. *Advanced engineering informatics*, *24*(1), 107–118.

Bosché, F. (2012). Plane-based registration of construction laser scans with 3d/4d building models. *Advanced Engineering Informatics*, *26*(1), 90–102.

Bosché, F., Ahmed, M., Turkan, Y., Haas, C. T., & Haas, R. (2015). The value of integrating scan-to-bim and scan-vs-bim techniques for construction monitoring using laser scanning and bim: The case of cylindrical mep components. *Automation in Construction*, *49*, 201–213.

Bosché, F., & Biotteau, B. (2015). Laser scanning and the continuous wavelet transform for flatness control. In *Isarc. proceedings of the international symposium on automation and robotics in construction* (Vol. 32, p. 1).

Bosché, F., & Guenet, E. (2014). Automating surface flatness control using terrestrial laser scanning and building information models. *Automation in construction*, *44*, 212–226.

Bosche, F., & Haas, C. (2008). Automated retrieval of 3d cad model objects in construction range images. *Automation in Construction*, *17*(4), 499–512.

Bosche, F. N., & Haas, C. T. (2008). Automated retrieval of project three-dimensional cad objects in range point clouds to support automated dimensional qa/qc. *Journal of Information Technology in Construction (ITcon)*, *13*(6), 71–85.

Braun, A., Tuttas, S., Borrmann, A., & Stilla, U. (2015). A concept for automated construction progress monitoring using bim-based geometric constraints and photogrammetric point clouds. *Journal of Information Technology in Construction (ITcon)*, *20*(5), 68–79.

Brutto, M. L., & Meli, P. (2012). Computer vision tools for 3d modelling in archaeology. *International Journal of Heritage in the Digital Era*, *1*(1_suppl), 1–6.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*(6), 679–698.

Cheung, K.-M., Baker, S., & Kanade, T. (2005). Shape-from-silhouette across time part i: Theory and algorithms. *International Journal of Computer Vision*, *62*(3), 221–247.

Cignoni, P. (2010). *Measuring the difference between two meshes.* Retrieved from http://meshlabstuff.blogspot.com/2010/01/measuring-difference-between-two-meshes.html (Accessed on 20/06/2017)

Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., & Ranzuglia, G. (2008). Meshlab: an open-source mesh processing tool. In *Eurographics italian chapter conference* (Vol. 2008, pp. 129–136).

Cloudcompare - 3d point cloud and mesh processing software open source project. (n.d.). Retrieved from http://www.danielgm.net/cc/ (Accessed on 06/04/2017)

Czerniawski, T., Nahangi, M., Walbridge, S., & Haas, C. T. (2015). Automated dimensional compliance assessment with incomplete point cloud.

Das, S., & Meher, S. (2013). Automatic extraction of height and stride parameters for human recognition. In *Engineering and systems (sces), 2013 students conference on* (pp. 1–6).

Dimitrov, A., & Golparvar-Fard, M. (2014). Vision-based material recognition for automated monitoring of construction progress and generating building information modeling from unordered site image collections. *Advanced Engineering*

Informatics, *28*(1), 37–49.

Ellenberg, A., Branco, L., Krick, A., Bartoli, I., & Kontsos, A. (2014). Use of unmanned aerial vehicle for quantitative infrastructure evaluation. *Journal of Infrastructure Systems*, *21*(3), 04014054.

El-Omari, S., & Moselhi, O. (2011). Integrating automated data acquisition technologies for progress reporting of construction projects. *Automation in construction*, *20*(6), 699–705.

Eschmann, C., & Wundsam, T. (2017). Web-based georeferenced 3d inspection and monitoring of bridges with unmanned aircraft systems. *Journal of Surveying Engineering*, 04017003.

Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395.

Fonstad, M. A., Dietrich, J. T., Courville, B. C., Jensen, J. L., & Carbonneau, P. E. (2013). Topographic structure from motion: a new development in photogrammetric measurement. *Earth Surface Processes and Landforms*, *38*(4), 421–430.

Fuhrmann, S. (n.d.). *Mve.* Retrieved from https://github.com/simonfuhrmann/mve (Accessed on 20/06/2017)

Furukawa, Y. (n.d.). *Cmpmvs - multi-view reconstruction software.* Retrieved from https://www.di.ens.fr/cmvs/ (Accessed on 11/11/2016)

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

Furukawa, Y., & Ponce, J. (2010). Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence*, *32*(8), 1362–1376.

Gadomski, P. (n.d.). *cpd.* Retrieved from https://github.com/gadomski/cpd (Accessed on 06/04/2017)

Gadomski, P. (2016). Measuring glacier surface velocities with lidar: A comparison of three-dimensional change detection methods.

Gibelli, D., Obertová, Z., Ritz-Timme, S., Gabriel, P., Arent, T., Ratnayake, M., ... Cattaneo, C. (2016). The identification of living persons on images: A literature review. *Legal Medicine*, *19*, 52–60.

Girardeau-Montaut, D. (2011). Cloudcompare-open source project. *OpenSource Project*.

Goesele, M., Snavely, N., Curless, B., Hoppe, H., & Seitz, S. M. (2007). Multi-view stereo for community photo collections. In *Computer vision, 2007. iccv 2007. ieee 11th international conference on* (pp. 1–8).

Golparvar-Fard, M., Bohn, J., Teizer, J., Savarese, S., & Peña-Mora, F. (2011). Evaluation of image-based modeling and laser scanning accuracy for emerging automated performance monitoring techniques. *Automation in Construction*, *20*(8), 1143–1155.

Golparvar-Fard, M., Peña-Mora, F., & Savarese, S. (2009). D4ar–a 4-dimensional augmented reality model for automating construction progress monitoring data collection, processing and communication. *Journal of information technology in construction*, *14*(13), 129–153.

Golparvar-Fard, M., Peña-Mora, F., & Savarese, S. (2012). Automated progress monitoring using unordered daily construction photographs and ifc-based building information models. *Journal of Computing in Civil Engineering*, *29*(1), 04014025.

Golparvar-Fard, M., Savarese, S., & Peña-Mora, F. (2010). Automated model-based recognition of progress using daily construction photographs and ifc-based 4d models. In *Construction research congress 2010: Innovation for reshaping construction practice* (pp. 51–60).

Ham, Y., Han, K. K., Lin, J. J., & Golparvar-Fard, M. (2016). Visual monitoring of civil infrastructure systems via camera-equipped unmanned aerial vehicles (uavs): a review of related works. *Visualization in Engineering*, *4*(1), 1.

Hamledari, H., & McCabe, B. (2016). Automated visual recognition of indoor project-related objects: Challenges and solutions. In *Construction research congress 2016* (pp. 2573–2582).

Han, K. K., Cline, D., & Golparvar-Fard, M. (2015). Formalized knowledge of construction sequencing for visual monitoring of work-in-progress via incomplete point clouds and low-lod 4d bims. *Advanced Engineering Informatics*, *29*(4), 889–901.

Han, K. K., & Golparvar-Fard, M. (n.d.). Bim-assisted structure-from-motion for analyzing and visualizing construction progress deviations through daily site images and bim. In *Computing in civil engineering 2015* (pp. 596–603).

Han, K. K., & Golparvar-Fard, M. (2014). Automated monitoring of operation-level construction progress using 4d bim and daily site photologs. In *Construction research congress 2014: Construction in a global network* (pp. 1033–1042).

Han, K. K., Muthukumar, B., & Golparvar-Fard, M. (n.d.). Enhanced appearance-based material classification for the monitoring of operation-level construction progress through the removal of occlusions. In *Construction research congress 2016* (pp. 879–889).

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

Hauswiesner, S., Straka, M., & Reitmayr, G. (2011). Free viewpoint virtual try-on with commodity depth cameras. In *Proceedings of the 10th international conference on virtual reality continuum and its applications in industry* (pp. 23–30).

Huang, X., Xu, H., Wu, L., Dai, H., Yao, L., & Han, F. (2016). A data fusion detection method for fish freshness based on computer vision and near-infrared spectroscopy. *Analytical Methods*, *8*(14), 2929–2935.

Ibrahim, Y., Lukins, T. C., Zhang, X., Trucco, E., & Kaka, A. (2009). Towards automated progress assessment of workpackage components in construction projects using computer vision. *Advanced Engineering Informatics*, *23*(1), 93–103.

Jackman, P., Sun, D.-W., & Allen, P. (2011). Recent advances in the use of computer vision technology in the quality assessment of fresh meats. *Trends in Food Science & Technology*, *22*(4), 185–197.

Jain, T. (2011). Computer vision: An enterprise of industrial automation and integration. *Advances in Modeling, Optimization and Computing*, 1015.

Jancosek, M. (n.d.). *Cmpmvs - multi-view reconstruction software.* Retrieved from http://ptak.felk.cvut.cz/sfmservice/websfm.pl?menu=cmpmvs (Accessed on 19/01/2017)

Jancosek, M., & Pajdla, T. (2011). Multi-view reconstruction preserving weakly-supported surfaces. In *Computer vision and pattern recognition (cvpr), 2011 ieee conference on* (pp. 3121–3128).

Jin, X.-H., & Le, Y. (2014). Monitoring construction projects using information technologies. In *Proceedings of the 17th international symposium on advancement of construction management and real estate* (pp. 1011–1020).

Kale, P. V., & Sharma, S. D. (2012). A review of securing home using video surveillance. *International Journal of Science and Research (IJSR)*, *3*.

Karsch, K., Golparvar-Fard, M., & Forsyth, D. (2014). Constructaide: analyzing and visualizing construction sites through photographs and building models. *ACM Transactions on Graphics (TOG)*, *33*(6), 176.

Kim, C., Lee, J., Cho, M., & Kim, C. (2011). Fully automated registration of 3d cad model with point cloud from construction site. In *28th international symposium on automation and robotics in construction* (pp. 917–922).

Kim, C., Son, H., & Kim, C. (2013). Automated construction progress measurement using a 4d building information model and 3d data. *Automation in Construction*, *31*, 75–82.

Kim, J., & Chi, S. (2017). Adaptive detector and tracker on construction sites using functional integration and online learning. *Journal of Computing in Civil Engineering*, 04017026.

Ko, J., & Ho, Y.-S. (2016). 3d point cloud generation using structure from motion with multiple view images. In *The korean institute of smart media fall conference* (pp. 91–92).

Kolivand, H., Tomi, B., Zamri, N., & Sunar, M. S. (2015). Virtual surgery, applications and limitations. In *Medical imaging technology* (pp. 169–195). Springer.

Kropp, C., Koch, C., & König, M. (2014). Drywall state detection in image data for automatic indoor progress monitoring. In *Computing in civil and building engineering (2014)* (pp. 347–354).

Leung, S.-w., Mak, S., & Lee, B. L. (2008). Using a real-time integrated communication system to monitor the progress and quality of construction works. *Automation in construction*, *17*(6), 749–757.

Lin, J. J., & Golparvar-Fard, M. (2016). Web-based 4d visual production models for decentralized work tracking and information communication on construction sites. In *Construction research congress 2016* (pp. 1731–1741).

Lin, J. J., Han, K. K., & Golparvar-Fard, M. (2015). A framework for model-driven acquisition and analytics of visual data using uavs for automated construction progress monitoring. In *Computing in civil engineering 2015* (pp. 156–164).

Linder, W. (2009). *Digital photogrammetry.* Springer.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91–110.

Marr, W. A. (2007). Why monitor performance? In *7th fmgm 2007: Field measurements in geomechanics* (pp. 1–27).

Meshlab. (n.d.). Retrieved from http://www.meshlab.net/,note={Accessedon10/07/2017}

Miziński, B., & Niedzielski, T. (2017). Fully-automated estimation of snow depth in near real time with the use of unmanned aerial vehicles without utilizing ground control points. *Cold Regions Science and Technology*, *138*, 63–72.

Moulon, P., Monasse, P., & Marlet, R. (2013). Global fusion of relative motions for robust, accurate and scalable structure from motion. In *Proceedings of the ieee international conference on computer vision* (pp. 3248–3255).

Moulon, P., Monasse, P., Marlet, R., & Others. (n.d.). *Openmvg (open multiple view geometry).* Retrieved from https://github.com/openMVG/openMVG (Accessed on 22/06/2017)

Moulon, P., Monasse, P., Perrot, R., & Marlet, R. (2016). Openmvg: Open multiple view geometry. In *International workshop on reproducible research in pattern recognition* (pp. 60–74).

Myronenko, A., & Song, X. (2010). Point set registration: Coherent point drift. *IEEE transactions on pattern analysis and machine intelligence*, *32*(12), 2262–2275.

Nahangi, M., Czerniawski, T., Yeung, J., Haas, C. T., Walbridge, S., & West, J. (2015). An image-based frameworks for automated discrepancy quantification and realignment of industrial assemblies.

Nahangi, M., & Haas, C. T. (2014). Automated 3d compliance checking in pipe spool fabrication. *Advanced Engineering Informatics*, *28*(4), 360–369.

Nkanza, N. (2005). Image registration and its application to computer vision: mosaicing and independant motion detection.

Nvidia. (n.d.). *Nvidia titan x.* Retrieved from https://www.nvidia.com/en-us/geforce/products/10series/titan-x-pascal/ (Accessed on 10/07/2017)

Nvidia, C. (2010). *Programming guide.*

Openmvs: open multi-view stereo reconstruction library. (n.d.). Retrieved from https://github.com/cdcseacave/openMVS (Accessed on 22/06/2017)

Özyeşil, O., Voroninski, V., Basri, R., & Singer, A. (2017). A survey of structure from motion*. *Acta Numerica*, *26*, 305–364.

Patrick, R., & Bourbakis, N. (2009). Surveillance systems for smart homes: A comparative survey. In *Tools with artificial intelligence, 2009. ictai'09. 21st international conference on* (pp. 248–252).

Perez-Perez, Y., Golparvar-Fard, M., & El-Rayes, K. (n.d.). Semantic and geometric labeling for enhanced 3d point cloud segmentation. In *Construction research congress 2016* (pp. 2542–2552).

Pomerleau, F. (n.d.). *libpointmatcher.* Retrieved from https://github.com/ethz-asl/libpointmatcher (Accessed on 06/04/2017)

Pomerleau, F., Colas, F., & Siegwart, R. (2015). A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics*, *4*(1), 1–104.

Pomerleau, F., Colas, F., Siegwart, R., & Magnenat, S. (2013). Comparing icp variants on real-world data sets. *Autonomous Robots*.

Pomerleau, F., Magnenat, S., Colas, F., Liu, M., & Siegwart, R. (2011). Tracking a depth camera: Parameter exploration for fast icp. In *Intelligent robots and systems (iros), 2011 ieee/rsj international conference on* (pp. 3824–3829).

Rashidi, A., Brilakis, I., & Vela, P. (2014). Generating absolute-scale point cloud data of built infrastructure scenes using a monocular camera setting. *Journal of Computing in Civil Engineering*, *29*(6), 04014089.

Rebolj, D., Babič, N. Č., Magdič, A., Podbreznik, P., & Pšunder, M. (2008). Automated construction activity monitoring system. *Advanced engineering informatics*, *22*(4), 493–503.

Remondino, F., Del Pizzo, S., Kersten, T., & Troisi, S. (2012). Low-cost and open-source solutions for automated image orientation–a critical overview. *Progress in cultural heritage preservation*, 40–54.

Rote, G. (1991). Computing the minimum hausdorff distance between two point sets on a line under translation. *Information Processing Letters*, *38*(3), 123–127.

Rothermel, M., Wenzel, K., Fritsch, D., & Haala, N. (2012). Sure: Photogrammetric surface reconstruction from imagery. In *Proceedings lc3d workshop, berlin* (Vol. 8).

Rucklidge, W. (1996). *Efficient visual recognition using the hausdorff distance* (Vol. 1173). Springer Berlin.

Sacks, R., Navon, R., Brodetskaia, I., & Shapira, A. (2005). Feasibility of automated monitoring of lifting equipment in support of project control. *Journal of* *construction engineering and management*, *131*(5), 604–614.

Saldaña, E., Siche, R., Luján, M., & Quevedo, R. (2013). Review: computer vision applied to the inspection and quality control of fruits and vegetables. *Brazilian Journal of Food Technology*, *16*(4), 254–272.

Sansoni, G., Trebeschi, M., & Docchio, F. (2009). State-of-the-art and applications of 3d imaging sensors in industry, cultural heritage, medicine, and criminal investigation. *Sensors*, *9*(1), 568–601.

Scharstein, D., & Szeliski, R. (2002). *Middlebury stereo evaluation-version 2.*

Schonberger, J. L., & Frahm, J.-M. (2016). Structure-from-motion revisited. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4104–4113).

Schönberger, J. L., Zheng, E., Frahm, J.-M., & Pollefeys, M. (2016). Pixelwise view selection for unstructured multi-view stereo. In *European conference on computer vision* (pp. 501–518).

Schöps, T., Schönberger, J. L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., & Geiger, A. (n.d.). A multi-view stereo benchmark with high-resolution images and multi-camera videos.

Schnberger, J. (n.d.). *Colmap.* Retrieved from https://github.com/colmap/colmap (Accessed on 31/05/2017)

Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., & Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer vision and pattern recognition, 2006 ieee computer society conference on* (Vol. 1, pp. 519–528).

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

Siebert, S., & Teizer, J. (2014). Mobile 3d mapping for surveying earthwork projects using an unmanned aerial vehicle (uav) system. *Automation in Construction*, *41*, 1–14.

Skarlatos, D., & Kiparissi, S. (2012). Comparison of laser scanning, photogrammetry and sfm-mvs pipeline applied in structures and artificial surfaces. *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences*, *3*, 299–304.

Snavely, N., Seitz, S. M., & Szeliski, R. (2006a). Photo tourism: exploring photo collections in 3d. In *Acm transactions on graphics (tog)* (Vol. 25, pp. 835–846).

Snavely, N., Seitz, S. M., & Szeliski, R. (2006b). Photo tourism: exploring photo collections in 3d. In *Acm transactions on graphics (tog)* (Vol. 25, pp. 835–846).

Son, H., & Kim, C. (2010). 3d structural component recognition and modeling method using color and 3d data for construction progress monitoring. *Automation in Construction*, *19*(7), 844–854.

Son, H., Kim, C., Hwang, N., Kim, C., & Kang, Y. (2014). Classification of major construction materials in construction environments using ensemble classifiers. *Advanced Engineering Informatics*, *28*(1), 1–10.

Son, H., Kim, C., & Kim, C. (2011). Automated color model–based concrete detection in construction-site images by using machine learning algorithms. *Journal of Computing in Civil Engineering*, *26*(3), 421–433.

Son, H., Kim, C., & Kwon Cho, Y. (2017). Automated schedule updates using as-built data and a 4d building information model. *Journal of Management in Engineering*, 04017012.

Sonka, M., Hlavac, V., & Boyle, R. (2008). *Image processing, analysis, and machine vision.* Thomson Learning.

Sweeney, C. (n.d.). *Theiasfm.* Retrieved from https://github.com/sweeneychris/TheiaSfM (Accessed on 20/06/2017)

Sweeney, C., Hollerer, T., & Turk, M. (2015). Theia: A fast and scalable structure-from-motion library. In *Proceedings of the 23rd acm international conference on multimedia* (pp. 693–696).

Tsoulkas, V., Kostopoulos, D., & Leventakis, G. (2014). Real time asset monitoring and risk management of critical infrastructures. In *Vulnerability, uncertainty, and risk: Quantification, mitigation, and management* (pp. 746–751).

Turkan, Y., Bosche, F., Haas, C. T., & Haas, R. (2012). Automated progress tracking using 4d schedule and 3d sensing technologies. *Automation in Construction*, *22*, 414–421.

Turkan, Y., Bosché, F., Haas, C. T., & Haas, R. (2012). Toward automated earned value tracking using 3d imaging tools. *Journal of construction engineering and management*, *139*(4), 423–433.

Valero, E., & Bosché, F. (2016). Automatic surface flatness control using terrestrial laser scanning data and the 2d continuous wavelet transform. In *Isarc. proceedings of the international symposium on automation and robotics in construction* (Vol. 33, p. 1).

Wu, C. (n.d.). *Visualsfm : A visual structure from motion system.* Retrieved from http://ccwu.me/vsfm/index.html (Accessed on 11/11/2016)

Wu, C. (2013). Towards linear-time incremental structure from motion. In *3dtv-conference, 2013 international conference on* (pp. 127–134).

Wu, Y., Kim, H., Kim, C., & Han, S. H. (2009). Object recognition in construction-site images using 3d cad-based filtering. *Journal of Computing in Civil Engineering*, *24*(1), 56–64.

Yan, Y., Guldur, B., & Hajjar, J. F. (n.d.). Automated structural modelling of bridges from laser scanning. In *Structures congress 2017* (pp. 457–468).

Yang, J., Arif, O., Vela, P. A., Teizer, J., & Shi, Z. (2010). Tracking multiple workers on construction sites using video cameras. *Advanced Engineering Informatics*, *24*(4), 428–434.

Yu, X., Wang, J., Kays, R., Jansen, P. A., Wang, T., & Huang, T. (2013). Automated identification of animal species in camera trap images. *EURASIP Journal on Image and Video Processing*, *2013*(1), 52.

Yuan, C., Li, S., & Cai, H. (2016). Vision-based excavator detection and tracking using hybrid kinematic shapes and key nodes. *Journal of Computing in Civil Engineering*, *31*(1), 04016038.

Zhang, B., Huang, W., Li, J., Zhao, C., Fan, S., Wu, J., & Liu, C. (2014). Principles, developments and applications of computer vision for external quality inspection of fruits and vegetables: A review. *Food Research International*, *62*, 326–343.

Zhang, X., Bakis, N., Lukins, T. C., Ibrahim, Y. M., Wu, S., Kagioglou, M., ... Trucco, E. (2009a). Automating progress measurement of construction projects. *Automation in Construction*, *18*(3), 294–301.

Zhang, X., Bakis, N., Lukins, T. C., Ibrahim, Y. M., Wu, S., Kagioglou, M., ... Trucco, E. (2009b). Automating progress measurement of construction projects. *Automation in Construction*, *18*(3), 294–301.

Zhou, Z., Gong, J., & Guo, M. (2015). Image-based 3d reconstruction for posthurricane residential building damage assessment. *Journal of Computing in Civil Engineering*, *30*(2), 04015015.

Zhu, Z., Ren, X., & Chen, Z. (2016). Visual tracking of construction jobsite workforce and equipment with particle filtering. *Journal of Computing in Civil Engineering*, *30*(6), 04016023.

Zollmann, S., Hoppe, C., Kluckner, S., Poglitsch, C., Bischof, H., & Reitmayr, G. (2014). Augmented reality for construction site monitoring and documentation. *Proceedings of the IEEE*, *102*(2), 137–154.

Zubair, A. M., Zaimi, M., Majid, A., & Mushairry, M. (2006). A systematic approach for monitoring and evaluating the construction project progress.