PHOTOGRAMMETRY FOR 3D RECONSTRUCTION IN SOLIDWORKS AND ITS APPLICATIONS IN INDUSTRY

A Thesis

Submitted to the Faculty

of

Purdue University

by

Nikhil S. Potabatti

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Mechanical Engineering

August 2019

Purdue University

Indianapolis, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF COMMITTEE APPROVAL

I dedicate this thesis to my family. There has been constant, support from my parents in every situation, while the sister has been an inspiration to me as she is managing her family and her Career. I have always admired my father for his wisdom.

ACKNOWLEDGMENTS

First of all, I want to thank my graduate advisor Dr. Hazim El-Mounayri for his constant support during my graduate study, research and internship tenures. His motivation has pushed me to achieve more and complete my goals. For this thesis, he had shown immense enthusiasm and anticipation. During both the research and writing of this thesis, Dr. El- Mounayri's guidance was invaluable.

Also, I am grateful to my other committee advisors: Dr. Sohel Anwar and Dr. Jie Chan for their encouragement and insights.

During my internship at Dassault Systemes, my manager Mr. Shyam Venugopal had introduced me to Mr. Mark Rushton and his photogrammetry project idea. My Manager had introduced me several other people at Dassault Systemes for getting their help during programming and SOLIDWORKS API related issues.

Mr. Mark Rushton who had an idea about creating a small prototype of a big object using photogrammetry tool. During our meeting, we had discussed a tool inside SOLIDWORKS which can be used for quality inspection or reverse engineering using open source code photogrammetry. So thank you much Mark for your ideas and constant endless support.

With this platform, I would like to thank great people from Dassault Systemes : Mr. Suchit Jain, Mr. Scott Stanley, Mr. Ditmars Veinbachs, Mr. Nitish Jain.

This platform gives me an opportunity to say thanks to Mr. Derek Miller from Digital lab office at IUPUI library for helping me with scanners. Jeff Lee Rogers from "Advanced Visualization Laboratory at Indiana University" helped me for creating and examining case studies for my proposed workflow. Hence I am very grateful to him and his team.

Last but not least, I would like to thank my parents, sister and family for supporting me financially, mentally and spiritually my whole life.

PREFACE

This thesis work is an outcome of progress in the Photogrammetry field over the years with new ideas. The idea of this thesis project was given by Mr. Mark Rushton who is Senior Product Portfolio Manager at Dassault Systemes - SOLIDWORKS at Waltham Campus, for which I am incredibly grateful. During my internship, my managers allowed me to continue my thesis work, to whom I say thank you. The technological resources and information which were needed to finish this thesis project were provided by Dassault Systemes and its employees.

This thesis project has primarily carried out as an individual project, but I have a lot of selfless lending hand from my internship colleagues at Dassault Systemes and IUPUI. For that, I reciprocate my gratitude.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Figure Page

Figure Page

SYMBOLS

Ψ	Tilt Angle
θ	Step Angle

ABBREVIATIONS

3D	3 Dimensional
CAD	Computer Aided Design
LIDAR	Light Detection and Ranging
API	Application Program Interface
SFM	Structure from Motion
AI	Artificial Intelligence
CPU	Central Processing Unit
GPU	Graphical Processing Unit
POI	Point of Interest
DSLR	Digital Single Lens Reflex
UAV	Unmanned Aerial Vehicles
VR	Virtual Reality
ROI	Return on Investment
MP	Megapixels
CMOS	Complementary Metal Oxide Semiconductor
JPEG	Joint Photographic Experts Group (image file format)
OS	Operating System
GUI	Graphical User Interface
CLI	Command Line
JSON	JavaScript Object Notation
SIFT	Scale-invariant feature transform
PNG	Portable Network Graphics
TIFF	Tagged Image File Format
MB	Megabytes

fps	frames per second
mm	millimeters
MP	MegaPixels

GLOSSARY

Model	devise a representation, especially a mathematical one, of (a phenomenon or system)
Automate	convert (a process or facility) to mostly automatic operation.
Silhouette	the shape and layout of something visible if used a lighter like diffused light.
Topographic	related to the organized adjustment of the physical features of an area.
Perspective	the art of drawing solid objects on a two-dimensional surface to give the right impression of their height, width, depth and position concerning each other when viewed from a particular point.
SOLIDWORKS	SolidWorks is a solid modeling computer-aided design and computer-aided engineering computer program that runs on Microsoft Windows.
GitHub	GitHub is a web-based hosting service for version control using Git. It is mostly used for computer code. It offers all of the distributed version control and source code management functionality of Git also adding its features.
Spatial	associated to occupying space
Geomorphology	the study of the physical features of the surface of the earth and their relation to its geological structures.
Megapixels	a unit of graphic resolution equivalent to one million or (strictly) 1,048,576 (220) pixels.
RAW image	an unprocessed photograph captured with a digital camera.

DLL	Dynamic Link Library(.dll) file contains a library of functions and other information that can be accessed by a Windows program.
Cookie	A small text file (up to 4KB) created by a website that is stored in the user's computer either temporarily for that session only
Frame Rate	the frequency at which frames in a television picture, film, or video sequence are displayed.
Shutter Speed	the time for which a shutter is open at a given setting.

ABSTRACT

Potabatti, Nikhil S. M.S.M.E., Purdue University, August 2019. Photogrammetry for 3D Reconstruction in Solidworks and its Applications in Industry. Major Professor: Dr. Hazim El-Mounayri..

Close range, image based photogrammetry and LIDAR laser scanning technique are commonly utilized methodologies to snap real objects.3D models of already existing model or parts can be reconstructed by laser scanning and photogrammetry. These 3D models can be useful in applications like quality inspection, reverse engineering.

With these techniques, they have their merits and limitations. Though laser scanners have higher accuracy, they require higher initial investment. Close-range photogrammetry is known for its simplicity, versatility and effective detection of complex surfaces and 3D measurement of parts. But photogrammetry techniques can be initiated with comparatively much lower initial cost with acceptable accuracy.

Currently, many industries are using photogrammetry for reverse engineering, quality inspection purposes. But, for photogrammetric object reconstruction, they are using different softwares. Industrial researchers are using commercial/open source codes for reconstruction and another stand-alone software for reverse engineering and mesh deviation analysis.

So the problem statement here for this thesis is **to integrate Photogrammetry, reverse engineering and deviation analysis to make one state-of-the-art workflow**.

The objectives of this thesis are as follows:

1. Comparative study between available source codes and identify suitable and stable code for integration; understand the photogrammetry methodology of that particular code.

2. To create a taskpane add-in using API for Integration of selected photogrammetry methodology and facilitate methodology with parameters.

3. To demonstrate the photogrammetric workflow followed by a reverse engineering case studies to showcase the potential of integration.

4. Parametric study for number of images vs accuracy

5. Comparison of Scan results, photogrammetry results with actual CAD data

In this thesis, only open source code photogrammetry tools have been studied. Photogrammetric results obtained in the form of point cloud from these tools were compared to ideal point cloud from laser scanning tool. This was done using Cloud-Compare function. Once the best possible code for integrated is identified, its methodology was injected in SOLIDWORKS CAD tool. SOLIDWORKS tool provides functions like mesh cleanup(Pre-processing), reverse engineering and mesh-CAD deviation analysis. After integration has been done with API programming using CSharp, this workflow was tested with case studies for accuracy of results with actual 3D models. These CAD models were firstly converted to surface mesh and compared with mesh comparison function.

The case studies presented in this report shows that scan mesh and photogrammetry mesh have relative accuracy of within 2 mm for a object of size (150 to 300 mm). The parametric study presented suggests that higher number of images increases accuracy of the reconstructed model. After comparison of CAD data, photogrammetry and scan results it can be inferred that photogrammetry can replace laser scanners if allowed tolerances are little higher. Hence this thesis, successfully presents reverse engineering function through photogrammetry in an integrated environment.

1. INTRODUCTION

1.1 Overview

The Science of calculating the geometrical data in terms of size, shape and position of objects by reviewing images/photographs captured using electronic medias like digital cameras or mobile phones with image capturing capabilities. The fundamental objective of photogrammetry methodology is to connect geometrical data of image plane and 3D object in space while imaging. After correctly establishing this relationship, the corresponding geometrical data and its physical quantities of that object could be gathered from those images. Photogrammetric method could be a game-changer in particular cases where objects those need to be measured are complex or hard to access, moving objects or deformations and their surface and/or contour data is required.

According to Slater, [1] there are limitations to the human eye. In terms of logistics, out of current computer vision/visualization devices, photogrammetry seems to be best promising method. Photogrammetric tools extracts accurate geometric properties and configurations of objects from overlapping pictures and automates to generate 3D content by triangulation method of common visible points from different photographs. Triangulation is a mathematical method for calculating intersections and lines in order to extract 3D coordinates. After complete calculations, one is left with close digital model obtained from the photographs [1]. To obtain necessary data about 3 space coordinate system at least 2 projections are necessary which can be derived from photographs. Rays(or lines of sight) are drawn from the position of the camera to point of interest (POI) and the mathematical equations are used for production [2]. From 2 photographs of the same objects, its original true dimension could be calculated to build 3D model [3]. Process involving multiple photographs requires high-end CPU and GPU with effective reconstruction software. If positions and directional locations of camera are known, then rays can be mathematically met to get *xyz* coordinates of the POI (see Figure 1.1) [2].

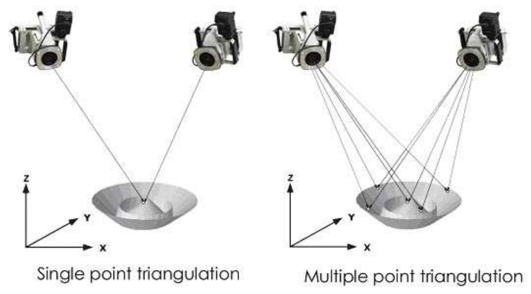


Fig. 1.1. Single point and Multipoint Triangulation[3]

A bundle triangulation happens when numerical convergences are simultaneously calculated for all spread out photographs. Multiple images generate many rays or lines of sight. The estimated *xyz* coordinates are located in local coordinate system. As the principal use of post-processing software tools is to create 3D model. But, these softwares are not able to handle scale factors and transformations [2].

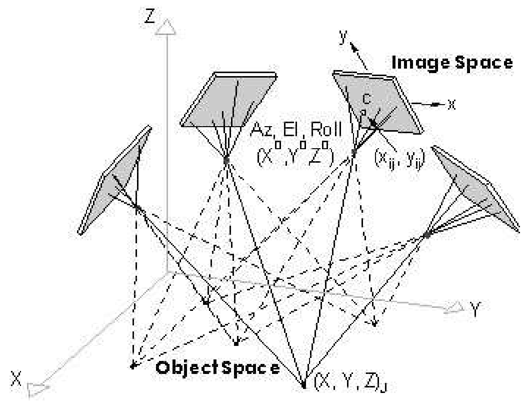


Fig. 1.2. Multiple POI's from multiple camera images[3]

Photogrammetry is a part of image-based modelling category, a group which consists exotic technologies like shape from shading, shape from silhouette and shape from texture [4].

Previously custom special instruments for photogrammetric measurements were needed for those whoever planning a 3D reconstruction project. Due to more significant degree of automation capability, standard computing equipments are used. Moreover, expert level skills are not necessary in order to carry out project work and processing. Look at flowchart suggesting reduction in total project time after evolution of digital cameras [2].

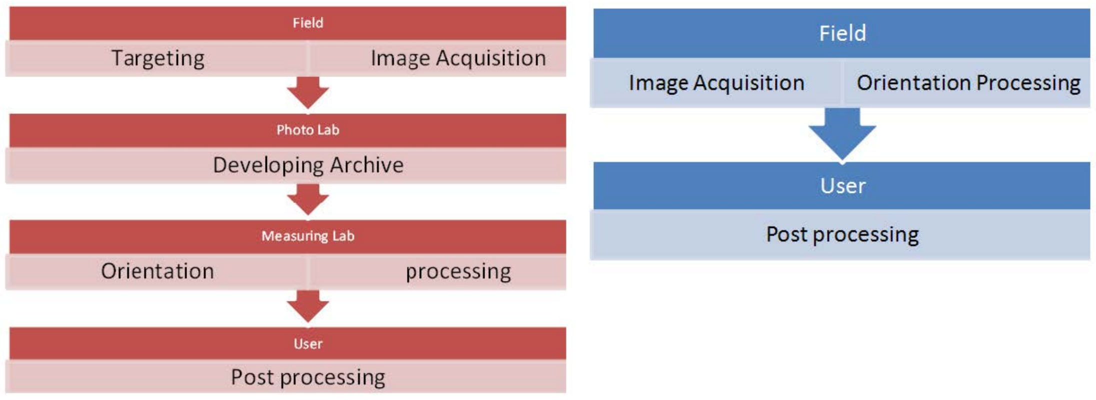


Fig. 1.3. `Workflow comparison between Analog (Left) and Digital (right) Photogrammetry system[3]`

1.1.1 History

Photogrammetry term comes from 3 Greek words; *phos* or *phot*, gamma and metrein; which means light, drawn and measure [5]. The concept of photogrammetry dates back in the late 1400s when the principles involved in geometrical analysis was studied by the Renaissance painters. Projective geometry was the next significant development which includes mathematical foundation of photogrammetry. After first successful testing powered-flight from Wright brothers in early 20th Century, concept of aerial/terrestrial photogrammetry has risen exponentially for topographic mapping which can be obtained by photographs. These photographs were captured from cameras installed on flying aircraft. Hence, the intensive research further conducted in this field, the developed photogrammetry methodology, tools and terms utilized are highly influenced by aerial photogrammetry [6].

With the exponential growth of cameras, electro-optical recorders and computers in the 1990s, use of the non-metric cameras were allowed by the analytical photogrammetric methods. Thus, we could find various applications of close-range photogrammetry in industrial quality inspection, medical image reconstruction and archaeological documentation. Also, there has been dramatic decrease in computer' cost with greater speed of Internet which made the use of complex 3D models available worldwide [7].

1.1.2 Computer Vision Development

Simultaneously, during computer evolution period, computer scientists have researched a lot about in the region perspective projection for computer vision . As we know the artificial intelligence (AI), they tend to adopt various formulations and much evolved mathematical models in perspective geometry, image algebra and differential geometry [6].

Electronic image sensors have replaced films; thus there are other expressions for photogrammetry that have been used to identify this extraction of 3D spatial data from images. There are different names, which are primarily a matter of personal choice of researchers of a unique application, include digital photogrammetry, videogrammetry, geomatics, videometrics, and computer vision. Here the commonly known term photogrammetry is adopted and occasionally replaced with some other terms [6].

1.1.3 Classification of Photogrammetry

Photogrammetric projects are generally conducted using camera devices like mobile phones, Digital Single Lens Reflex(DSLR) cameras or drones/unmanned aerial vehicles (UAV).

Quality of images varies as resolution of cameras are different. Thus these devices may require to record more photographs [8]. Photogrammetry techniques can be categorized as shown in the Fig. 1.4 [9]

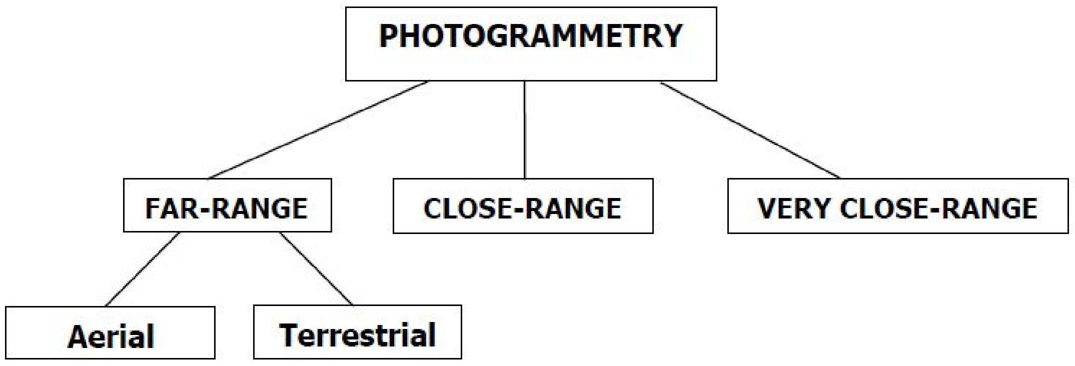


Fig. 1.4. `Classification of Photogrammetry methods[10]`

Far-range Photogrammetry: is applied to reconstruct the shape of building, to create topographical maps etc. [9] Range: (areas of m^2). Aerial photogrammetry is conducted using UAVs to which a camera is attached for capturing images while its in air [8].

Close-range and very close-range Photogrammetry : generally used for industrial applications where non-contact measurements has to be done. Range: could vary between (1 cm^2 to 1 mm^2) [9]. For Close-range photogrammetry, photographs are captured within 300 m of test object using DSLR cameras/ cellphone [8].

1.1.4 Characteristics of Close-range Photogrammetry

According to Jing [10], following are the characteristic of close-range photogrammetry.

- High Measurement Accuracy
  - Using Single camera measuring system, relative accuracy up to 5 μm can be achieved. For high configuration, it could be 5 μm/m [10].
- High Efficiency
  - Within concise period, feature information is extracted from tens of thousands of data points [10]
- Stable Performance
  - Test objects have unaffected high precision measurements quality when used in adverse environment like high temperature, hazardous environment, high pressure and electro-magnetic environment [10].
- Non-contact Measurement
  - So without touching or doing any damage to object high accuracy measurements can be done [10].

1.1.5 Photogrammetry Vs Laser Scanners

According to koelman [4], author has given comparative rationale using following points. Though there has been more recent developments in Laser scanning field than photogrammetry, the latter isn't obsolete. These two methods have differences in terms of measurement method as well as derived characteristics [4].

- Taking order of magnitude into consideration, number of measured points are typically in few hundreds while for laser scanning it's in millions.
- These number of measured points, which also known as point cloud, result into reliable surface marching automation with laser scanning. This means accuracy of photogrammetry is undoubtedly not as good as laser scanners. But

over the period of the development, computational cost can decrease; hence photogrammetry could be a cost-effective alternative to laser scanners [11].

- Comparing in terms of value, one has to make significant investment for laser scanners, but photogrammetric equipments are available at affordable prices.
- Test objects which are in focus view of laser-scanners can only be detected. So, if one has to scan an object which is too large for laser scanners to cover all of its surface features; additional scanners have to be installed in order to catch all features. This leads to cumbersome task of merging the results from various scanners into single scan mesh with uniform coordinate system. Photogrammetry provides that extra edge for final object which is created using photographs from different positions.
- With photogrammetry, common points between different photographs must be marked so that final object will have uniquely estimated points. If they are not recognizable naturally, object can be marked with color or stickers. Laser scanners do not need this prerequisite.
- For photogrammetry process, there is no limit on dimensions or complexity of test object, but many laser scanners are limited to tens or hundred meters.
- Comparing transport and installation, camera equipments are much more comfortable than laser scanners in that sense. This plays pivotal role while working in harsh surrounding of shipyard industry and obviously in confined places. Photogrammetry is applicable in constrained spaces like car interiors [12]. On the other hand, sufficient light is needed to take photographs, which is not always the case [4]. Photogrammetry is insensitive against shocks and vibrations [12]. Also, it is unaffected in broad range of temperatures. Also, photogrammetric system weighs near about 10 *kg*.

- Generally, scanned mesh obtained by laser scanners contain noises which has to be post-processed for its removal. There are inaccuracies in photogrammetric measurements, but these are taken care of in least square process which help to estimate 3D coordinates [4].
- From taking photographs and inserting into algorithm for object reconstruction can be done by less skilled person [11].

When analyzing a dense point cloud generated by laser scanning, to human eye might seem surfaces and features created in it. One might think conversion of that point cloud to workable CAD model is straightforward but indeed its a challenging work [13]. In [14], mixed answers are given to primary question Laser scanning or photogrammetry? Conclusion made in this literature was about importance of human interaction in both methodologies for measurements and modelling.

In [14], two methods were compared quantitatively. For case study of church as an object, photogrammetry and post-processing took 10 hours while just getting point cloud model from laser scanning took around 7-8 hours. (With laser scanning they haven't post-processed point cloud for surface creation). Author has compared accuracy of both of them. According to Koelman [4], as per the qualitative ranking, photogrammetry wins the battle by 23 to 17 points of laser scanning.

Based on comparative research, critical advantages of photogrammetry are about equipment' investment, its compatibility and flexibility and most importantly its ability to handle objects of any size and any complexity [4].

Table 1.1.
Laser Scanner Vs Photogrammetry]

Parameters	**Laser Scanner**	**Photogrammetry**
Cost	Significant	Affordable
Accuracy	High	High
Automation	High	Low
Field of View	Limited	No limit
Complexity and Dimensions	Limited	No limit
Transportation	Limited	Flexible
Post-processing	More	Less
Efficiency	Low	High
Feature Recognition	Not guaranteed	Included in the Process
Equipment	Large and Vulnerable	Small, not vulnerable

1.2 Objective of Thesis

During my internship at Dassault Systmes - SOLIDWORKS Boston campus, Mr. Mark Rushton had an idea of creating a small 3D printed prototype of a building which is modelled using photogrammetry technique. As he wanted to gift a prototype of some architecture on special days. So this thesis project turned into reality after we discussed currently available commercial codes and open source codes in the market.

Image reconstruction for 3D modelling as introduced before has immediate positive impact on realistic 3D render model, close-range photogrammetry. Currently, in the market state-of-the-art following commercial softwares are available : PhotoModeler, Pix4D, Agisoft Photoscan, DroneDeploy, Autodesk 123D catch etc. There are some free open source codes available like Alicevision Meshroom, MicMac, Colmap, Regard3D etc.

These tools helps user to get object reconstruction from images but does not support post-processing on these results. Viz. Mesh cleanup, reverse CAD modeling and deviation analysis. So, I recognized that there is a gap in reverse engineering tools from photogrammetry which don't have this state-of the-art work-flow built-in in one software. For this purpose SOLIDWORKS has been chosen for creating workflow.

Hence, the prime objective of this thesis research is to create a photogrammetry work-flow inside SOLIDWORKS software which could be used in the industry as cost-effective alternative to expensive laser scanners in the market. This thesis research's other objective is to study and understand already existing open source codes. And later integrate them in SOLIDWORKS using API capabilities using C# programming language. Third objective of this thesis is to validate this photogrammetric work-flow with case studies.

The following mentioned points are my significant contributions in this work:

1. Comparative study between open source codes and identify suitable and stable code; understand the photogrammetry methodology of that particular code.

2. To create a SOLIDWORKS add-in using its API for Integration of selected photogrammetry methodology and inject methodology with parameters.

3. To demonstrate the photogrammetric workflow followed by a reverse engineering case study to showcase the potential of integration.

1.3 Thesis Outline

In this thesis, chapters are outlined as follows:

- Chapter 2: In this chapter, there is a brief overview on various applications related to close-range photogrammetry in manufacturing, aerospace and mechanical industry. In addition to that, there is a discussion pertaining measuring procedure and test setup to get good results.

- Chapter 3: This chapter describes available open source photogrammetry tools and their tutorial results. Furthermore, there are explanations about their selection for integration.
- Chapter 4:Once photogrammetry have been finalized, next step is about studying and understanding algorithm for photogrammetry. After that, integration methods using GitHub libraries explained.
- Chapter 5: This chapter presents a way of add-in creation using SOLIDWORKS API and integration of photogrammetric algorithm. This includes discussion related to user interface.
- Chapter 6: This chapter demonstrates how to use photogrammetric tool for reverse engineering purposes. Along with that, this method is validated by comparing with original CAD files.

In the last chapter, conclusions and future work prospect has been mentioned.

2. LITERATURE REVIEW

2.1 Applications of Photogrammetry

For many decades, Photogrammetry has been used in areal surveying and architecture for object reconstruction. Other than these fields, this method can be applied in areas where it s desired to get the spatial shape of an existing object.

Close-range photogrammetry has been used in the field of topographical mapping, historical studies, archaeological explorations, and lately geomorphological research. In Kidson and manton [15], authors have used photogrammetry to survey geomorphology, coastal studies, stability of slopes and river channels.

Photogrammetry has been applied in Museums for creating a digital copy of dinosaur bones, historical architecture, or minor artifacts. For the production of a famous Hollywood movie, The Matrix, photogrammetry had been used [1].

2.1.1 Photogrammetry used for Virtual Reality (VR)

The current hot topic around the world is VR. This topic has inspired many researchers to create algorithms for effective object reconstruction. For facility-management application in the construction industry, these VR model can play an important role. Also, for teleoperating which means interaction with machinery, autonomous vehicles and robots virtual models are needed in mining industry [16]. However, the gaming industry is booming at a much higher rate than ever before, have shown significant interest in developing advanced reality games. Even entertainment industry is not lagging behind using virtual models created by object reconstruction.

As real-life models are getting more demand than artificially produced content, creating a VR objects does not inherently need any measurements to be made. They can be modelled from right from scratch. As real-life models are getting more demand than artificially produced content, creating VR objects does not inherently need any measurements to be made. They can be modeled from right from scratch. Just because of the evolution of higher computing and graphics processing rates, real-world data can potentially be generated as realistic looking models in a more automated and quicker way [17].

In many cases, original dimensions with real-life objects texture are often combined with artificial texture by partial graphical manipulation to get an appropriate virtual model. In cases where only remains of the monuments are present but entire structure of that archaeological site has to be recreated in virtual reality [18].

2.1.2 Close-range Photogrammetry Applications in Industry

In [19], author has divided close-range photogrammetry into off-line and on-line systems.

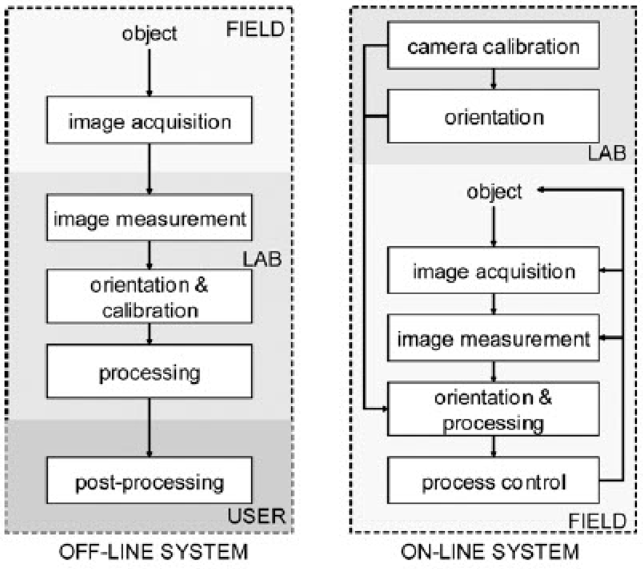


Fig. 2.1. `Close-range Photogrammetry Systems - off-line and on-line`

Off-line systems utilize high resolution DSLR cameras specifically wide angle lenses, retro reflective targets and give reconstructed object within minutes after bundle adjustment but this is more like one way work-flow. While on-line systems provide object data in a closed loop in real time which is linked directly to external processes [19]. See the above fig. 2.1

Off-line systems are used as 3D measurement tools and their application can be found in a variety of industrial areas:

- **Automotive** - Car body panel deformation, Quality control of supplier parts, Tooling and rigs adjustment, crash testing, etc [19].
- **Aerospace** - Measurement and tuning of mounting rigs, Antenna measurement, part-to-part alignment, etc [19].
- **Civil Engineering and Construction** - Measurement of water Dams, oil tanks and plant facilities etc [19].
- **Wind Energy Systems** - Deformation measurements in wind turbines and their production control [19].

2.1.3 Aerospace Industry

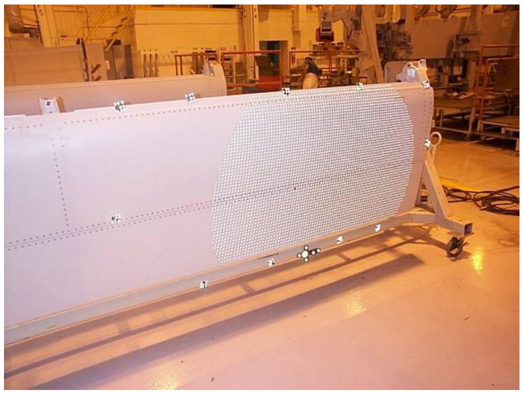


Fig. 2.2. Quality Assessment of Boeing 777 wing [10]

Though aircraft parts are complicated, the aviation industry demands manufactured parts of high precision and accuracy. Using to close-range photogrammetry measurement method, quality control inspection is done on aircraft components which are assembled or manufactured. They are inspected for the high quality requirements.

Boeing 777 wing is shown in the above fig 2.2. The V-STARS measuring system has inspected its quality [10].

Due to working conditions like a low-temperature vacuum, manufactured aerospace components have to check in that environment. Researchers have used Laser trackers and iGPS. However, these measuring systems require target detection rods placed in the object ball. As this process is in the vacuum, the non-contact close-range measuring system can facilitate inspection of an object in long-distance. European Space Agency has used photogrammetry for measuring the deformation of astronomical telescope which is shown in Fig 2.3 [10].

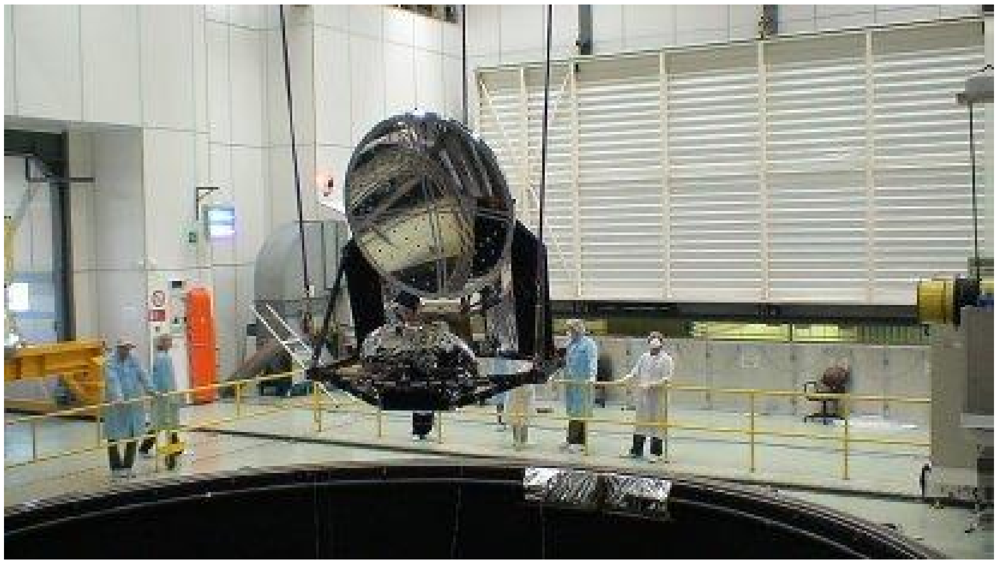


Fig. 2.3. Deformation of an Astronomical Telescope [10]

2.1.4 Automotive Industry

Automotive parts/components are mass produced and they need high production efficiency. By using on-line close-range photogrammetry system, its possible to achieve pointing and measurement of large Automotive parts. The system ultimately enhances the production of vehicles. See the attached image in fig.2.4 of BMW X5 BMW series automobile side panels measured using a close-range measuring system [10].

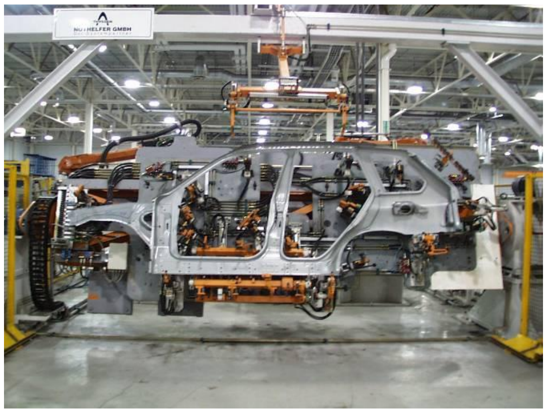


Fig. 2.4. Measurement of side panels of BMW X5 car [10]

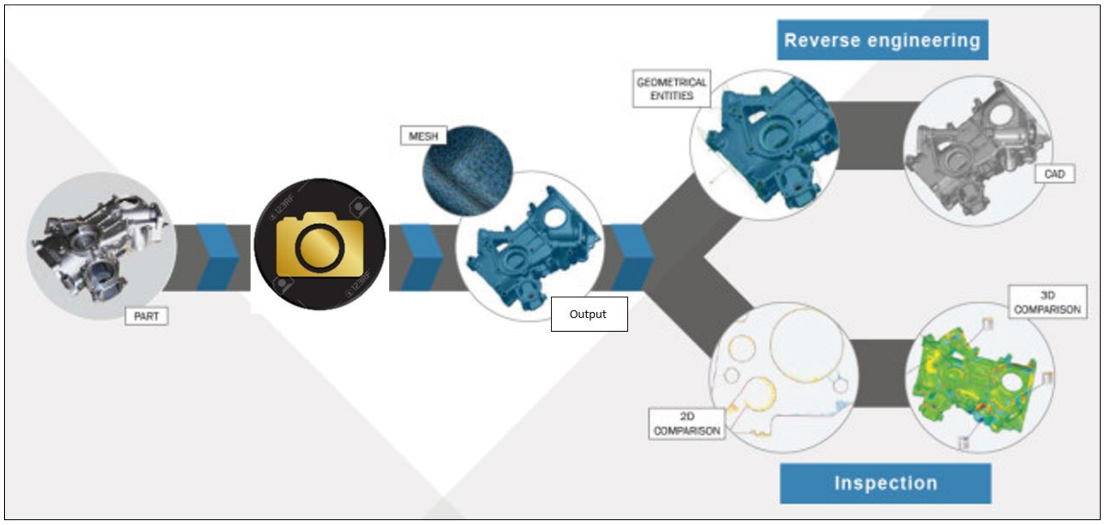


Fig. 2.5. General Approach of Photogrammetry in Automotive Industry

2.1.5 Ship Manufacturing

Compared to other industries, ships components are massive and diverse in shape and size. Close-range photogrammetry can achieve good accuracy over 10-12 m. So, if this system can provide accuracy near 0.1 mm, this can be used as an ideal tool for measurement of the ships components. See the following fig.2.6, how Japan IHI Corporation measure radial line(a ship component) [10].

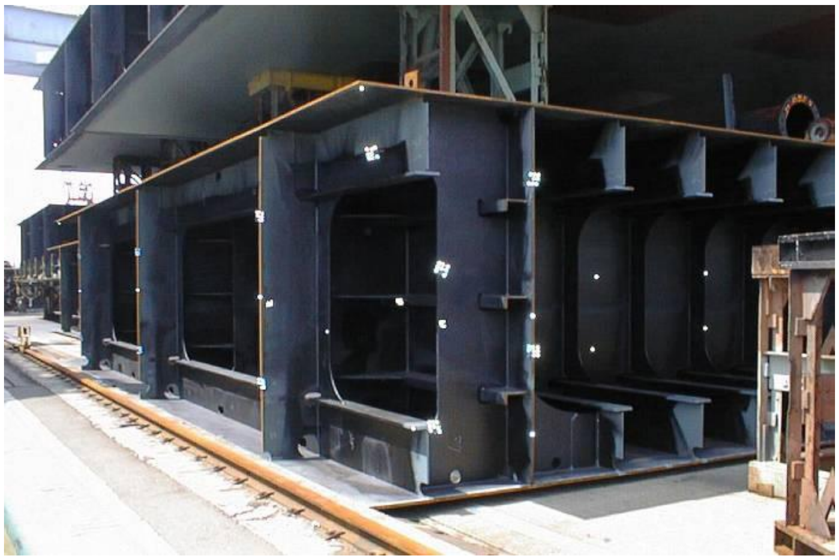


Fig. 2.6. Measurement of radial line - a ship component [10]

Since the 1980s, close-range photogrammetry has been used in the area of Ship manufacturing. In Atkinson [20], for estimating the shape of the hull of a container vessel, 1500 landmark points, and some 140 photos were used. Photogrammetry is applied for defining the location and state of internal apparatus, and for life-cycle support.

In [4], they have not only reconstructed the geometry of points but also the topology, in the form of connections between measured points. A skilled operator had photos at hand with all the needed information to perform those operations efficiently. See the Fig.2.7 and Fig.2.8 to understand, collection of photos and obtained reconstruction result [4].

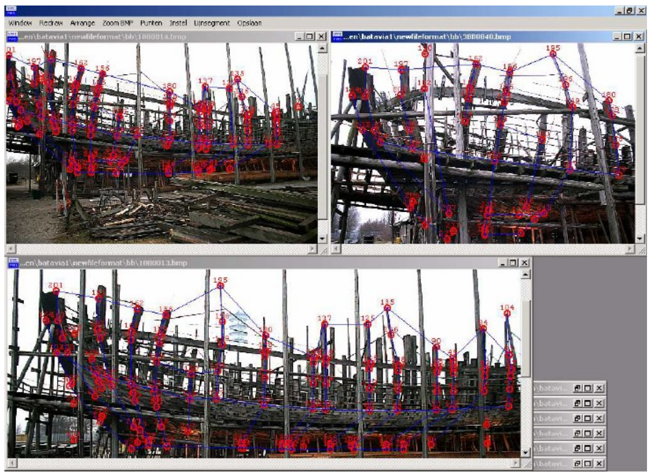


Fig. 2.7. Targets on ship, marked and connected using photos [4]

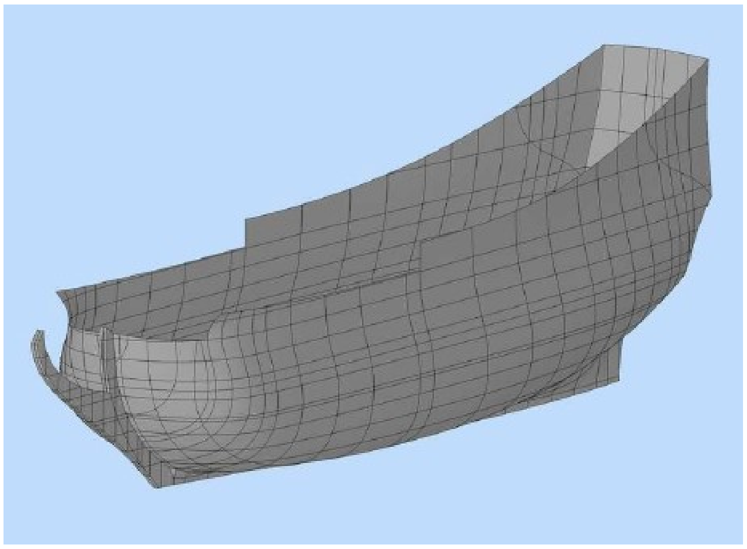


Fig. 2.8. Post-processed results of ship hull [4]

2.1.6 Weldments in the Industry

In [21], authors have proposed a new low-cost method for reconstruction of a scaled 3D model. For this, they have used the only tool which is a commercial (DSLR) camera with a macro-lens. Authors have also developed their own software for this purpose. For a detailed inspection of the weld, including the recognition of its surface imperfections and defects together with its discontinuities is allowed, enabling easy and automatic documentation and digitization of the quality inspection project [21]. Fig. 2.9 shows the data collection procedure for the different welding samples.

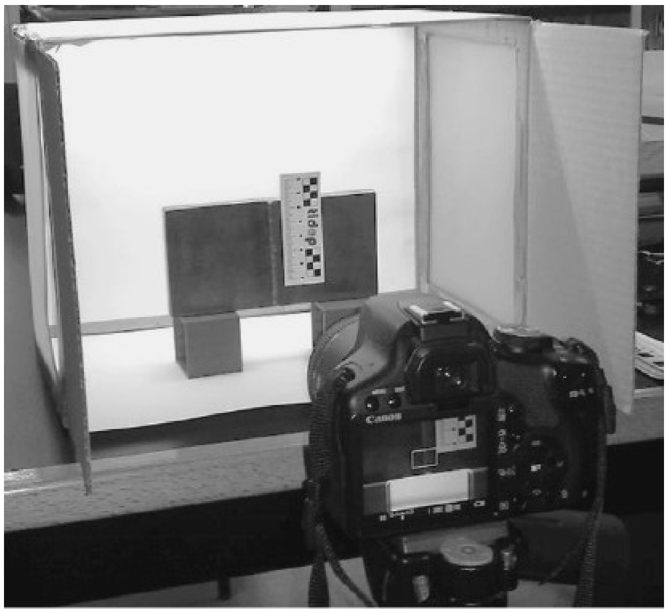


Fig. 2.9. Data Collection Procedure of welding sample [21]

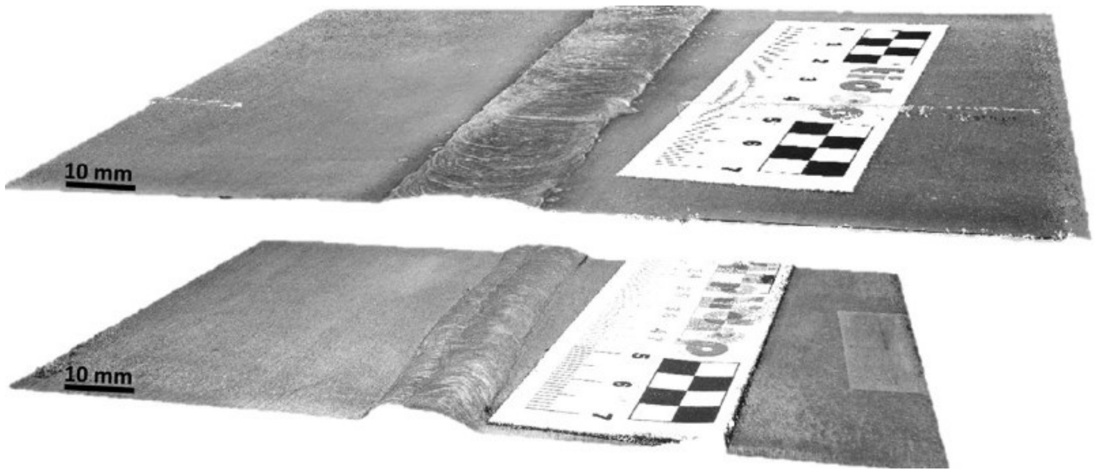


Fig. 2.10. 3D Dense point cloud surface models of welds [21]

2.1.7 Gas Turbine

Gas turbines are enormous which are generally up to 13 meters in length, 5 meters in height and weigh up to 400 tons. Components of this scale represent a considerable challenge in production as these gas turbines operate in demanding working conditions like ultra-high combustion temperatures, pressures and vibration. Also, there are large centrifugal forces, transient loads acting on them. Therefore precise dimensional accuracy is desired to ensure maximum factor of safety and stability [22].

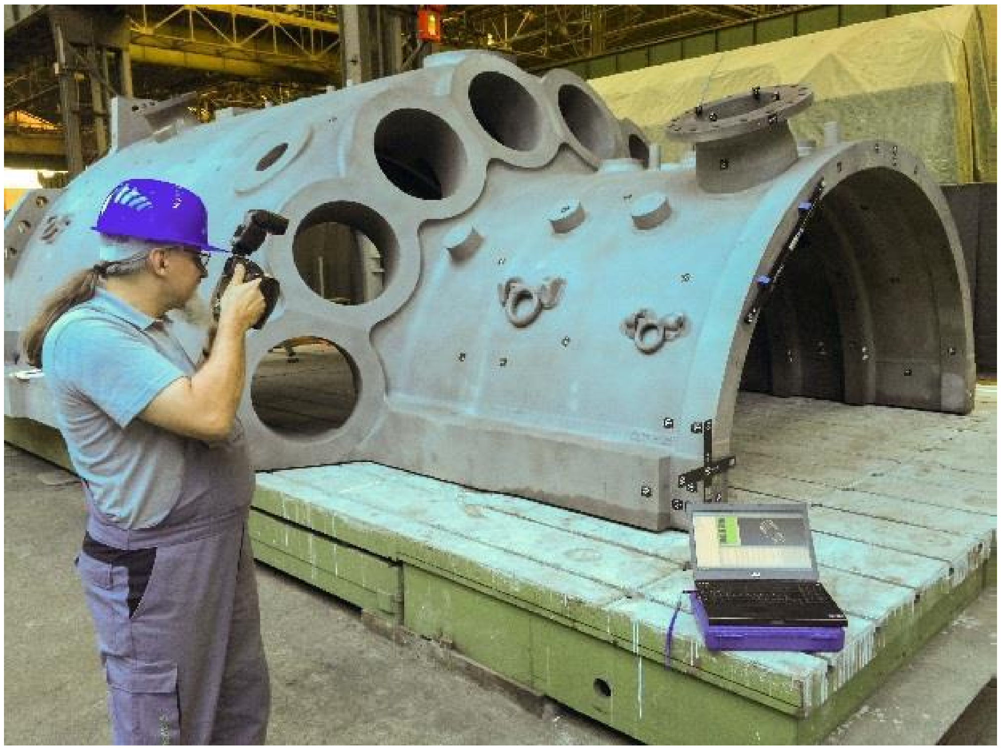


Fig. 2.11. Photo collection using high resolution digital camera [22]

2.1.8 Static Structural Analysis

In [23], authors present a general method for the assessment of truss structures of timber roof construction. These truss structures have complex geometries but its material properties, physical properties are unknown and cant be evaluated experimentally. They have performed many investigations on truss structures for measurement, deformation analysis and static structural testing by non-contact methods. These non-contact methods include topographic methods, close-range photogrammetry and scanning methods [23].

See following figures for reference.

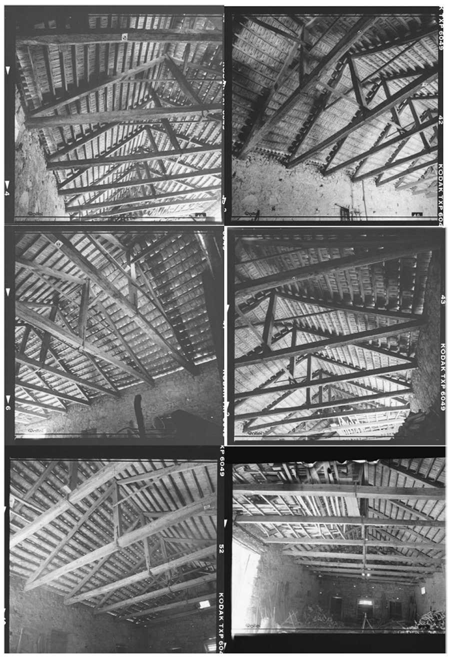
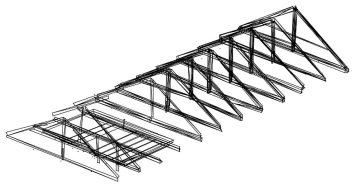
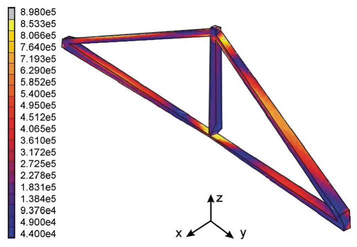


Fig. 2.12. `Images Captured using high resolution device` [23]



(a) 3D Wire-frame model.



(b) Von-Misses Stress

Fig. 2.13. Photogrammetry used for Structural Analysis [24]

2.2 Photogrammetry Test Setup

As mentioned earlier, Photogrammetry uses many images to crack the collinearity equations of collected data. In close-range photogrammetry, image matching is processed based on stereo-pairs. Photogrammetry software needs parallel images with at least 60% overlap. Also, photogrammetry involves control points in order to scale and recognize features. The position of the points must be identified for accurate data investigation to generate the 3D model [8]. Thus capturing photographs of the object is the crucial stage of entire process.

2.2.1 Acquisition of Images

During image acquisition process, it is important to know whats need to be measured and reconstructed using techniques like photogrammetry and dense matching. Along with that, an appropriate contrast and triangulation of surface points for photogrammetry should be considered. Thus, it is recommended to adjust within a correct angle of triangulation and an extreme variation of perspective views between successive photographs [24].

Author suggests to avoid high angular views and recommends angular views should range from 60 to 110 degrees(i.e. consecutive photos should have 20 to 30 degrees of difference). See figure 2.14. The accuracy of the photogrammetric depends on the basis of relative angle between pairs of images [24].

For photogrammetry from small to large scale objects image acquisition method changes. Figure 2.14 shows how to capture small scale object and Figure 2.15 shows how to capture large-scale object for reconstruction [8].

Based on the scale of an object to be modeled, correct method of photo capturing changes [8]. Generally images are captured around the object in clock-wise direction and combining dissimilar heights and view angles for each image acquisition position [24].

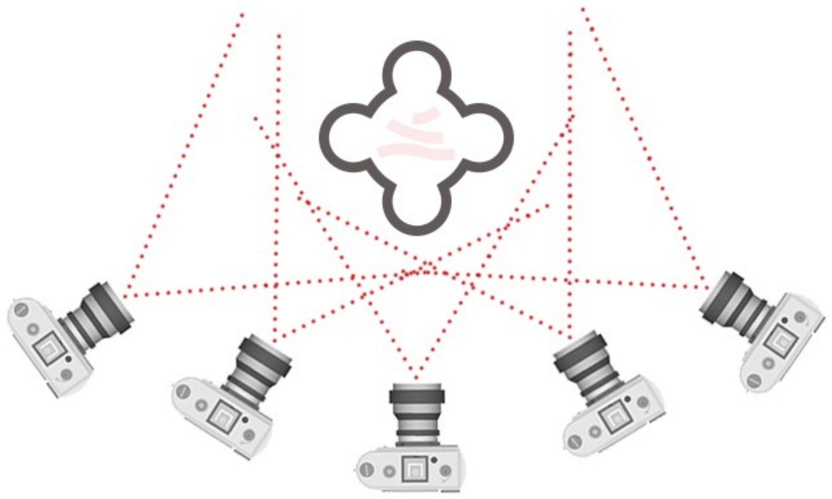


Fig. 2.14. `Image acquisition method for small objects' reconstruction` [8]

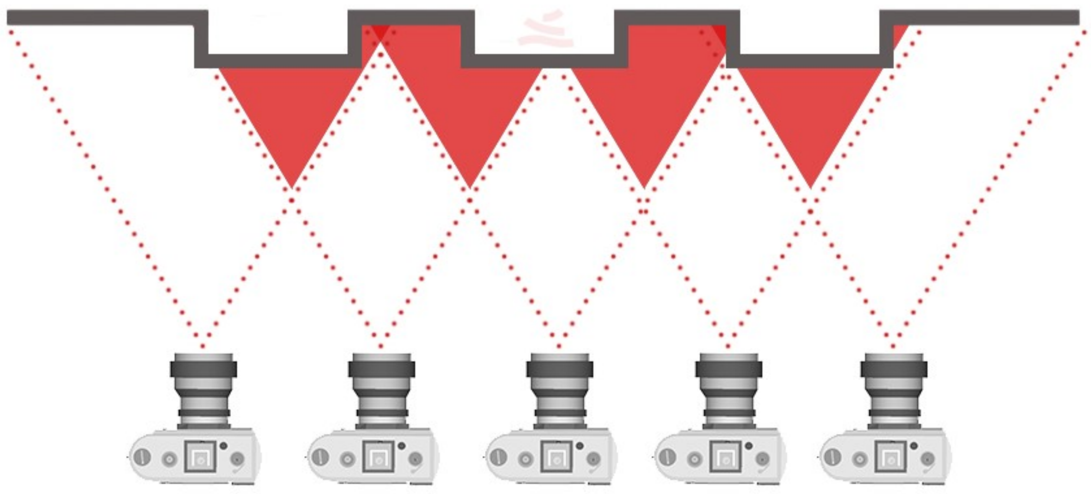
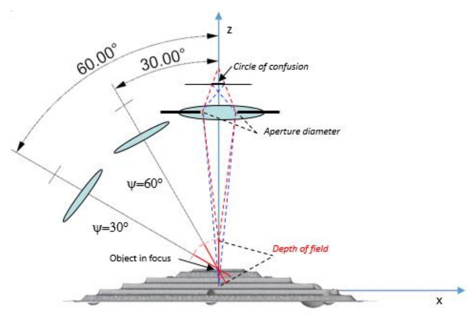
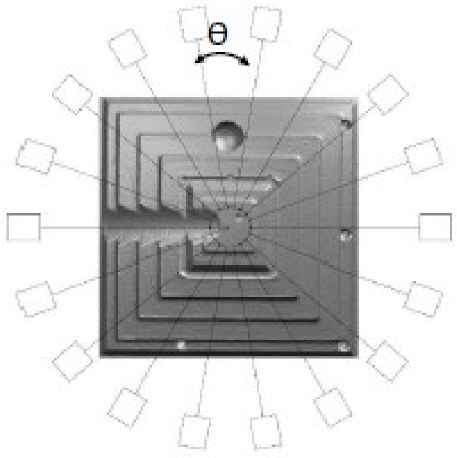


Fig. 2.15. `Image acquisition method for large objects' reconstruction` [8]

While acquiring small objects, depending on the shape and geometry of the object, a polar coordinate method for location of camera can be used. In this configuration, minimum two process parameters need to choose. One could be tilt angle and another could be step angle. As shown in the figure 2.16, Tilt angle (Ψ) denotes the tilt of the camera with respect to the xy plane of the object. And the step angle (Θ) is the rotating phase of the turning table. This generally controls the number of images and the overlying degree between two successive images. These are the two important factors which decides image capturing approach [25].



(a) Tilt angle (Ψ)



(b) step angle (Θ)

Fig. 2.16. Camera Positions Strategy

[25]

2.2.2 Setup Instructions

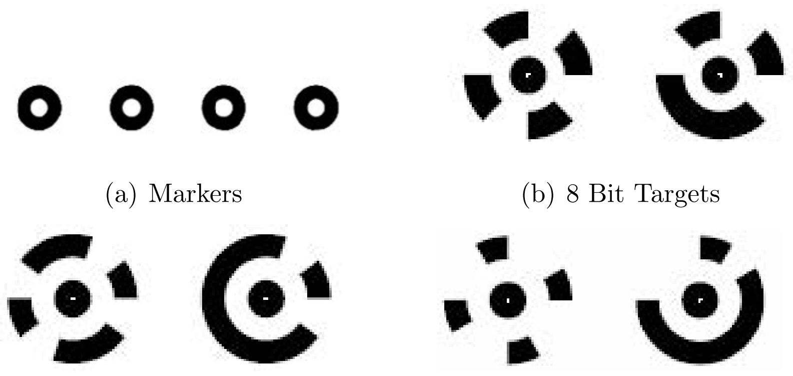
White cloth can be placed around the object scene to avoid detecting arbitrary points that could be processed as targets. This helps in the generation of the 3D point cloud data by avoiding unwanted data from the backgrounds of the object. This will reduce undesirable noise as well as the total time to perform the calculations of the point cloud and final mesh, as there are less points to consider in triangulation [1].

Use of tripod as stabilizer, is recommended in order to get desired sharpness of images. The scene should be properly illuminated with diffused light so as to avoid light reflections from object and dark shadows. If object surface is shiny, special removable matting powder could be sprinkled over objects body before taking images [26].

One of the option is to capture objects image outside on an overcast day, but this is not planned early. Static lighting environment is needed over longer stint to obtain enough images of acceptable quality. [1].

Preparation of the scene

In many research studies and company's case studies, markers and targets are used to get accurate feature recognition and measurement calibration. See figure 2.17 for reference. But, its not mandatory to have markers on object itself as conditions could be extreme.



(a) Markers (b) 8 Bit Targets

(c) 10 Bit Targets (d) 12 Bit Targets

Fig. 2.17. Markers and Targets for Measurement Calibration

The norms for the location of the coded targets are as following [24]:

- at least five shared targets between diverse images.
- Size of the mark should be an approximate size of 10 pixels
- Robustness is enhanced if there are eight or more targets
- The image reconstruction module of the software cannot identify coded targets other than red on black or white on black

Along with the targets, it is essential to have a calibrated length bar positioned on the scene that helps finding the scale of the captured images. Photographs should capture as many markers and coded targets in single picture as possible. To correctly reconstruct geometric features of object, photos can be taken in three different angles 75°, 55°, 35°(between scene plane and imaging surface plane) [26]. See the figure 2.18.

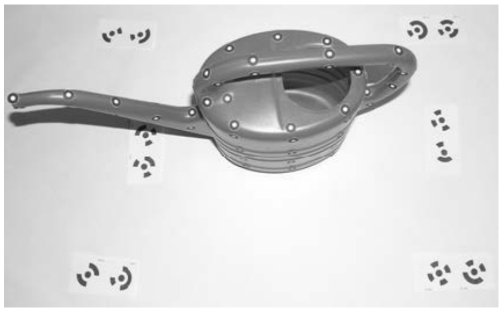


Fig. 2.18. Test setup with markers and coded targets [26]

Camera Adjustment

Once the scene background is set up, next important task is to make adjustments to camera parameters to suit illumination conditions of the room where images are being captured. Typically, the camera adjustment is for the coded targets to be detected in a photogrammetric process, viz. with maximum aperture and a zoom of 10x. In case of a dense matching, one should consider a balanced contrast on surface points to be captured. Hence, while making adjustments to the camera, a balance should be found between the coded Targets and the surface texture of the object [24].

In [24], author mentions the prime parameters need to be adjusted are as following:

These parameters should be maintained during complete image capturing procedure and measurement in order to solve equations with bundle adjustment. Software automatically identifies the camera and detects its parameters related to it like resolution and image sizes automatically.

Camera Calibration

Calibration for photogrammetry tools is vital for generating accurate model reconstruction. A reliable calibration method comprises controlled scenarios in a workroom. As discussed in earlier segments, calibration bars are used for scaling and measurement purposes. In-house reconstruction of household objects supports in analyzing the procedure of 3D model reconstruction through appropriate calibration [8].

In [8], author have found that even with low resolution cameras, high quality 3D models can be obtained using close-range photogrammetry in in-house setting. Anyhow, a higher quality camera produces a higher quality model. Also, the accuracy of reconstructed model depends on the number of images taken per object. As said higher the number of images will result a higher resolution 3D CAD [8].

2.2.3 Factors Affecting Close-range Photogrammetry

The accuracy of obtained reconstructed model from a photogrammetry measurement significantly relies on the many associative parameters that are tangled in the photogrammetric process. According to braybon [2], The most dominant factors contain following:

- **Camera and Lens Quality**: the resolution of the cameras has a substantial role in precisely locate the position of a targets and markers [2]. The accuracy is usually improved with smaller object pixel size. This Pixel means the smallest box size which stores image data of the captured object using corresponding camera. The object pixel size is affected by the megapixel of the device [7].
- **Sizes of Objects**: Smaller object sizes generally result into higher accuracy of reconstructed model.
- **Number of Images**: This is probably the most dominant factor defining the accuracy of models. Increasing the number of overlapping images enhances the

accuracy of the output 3D model. On the other hand, capturing the coded targets in more than four images does not improve results significantly [7].

- **Area of focus of Images**: Consecutive photographs taken with broader angles, result in the higher the accuracy of the recognition. As discussed earlier, angle of intersection should not be less than 60°. The ideal angle of intersection would be 90° [2]. See figure 2.19 illustrating accuracy pyramid and factors' influence on reconstructed model accuracy.

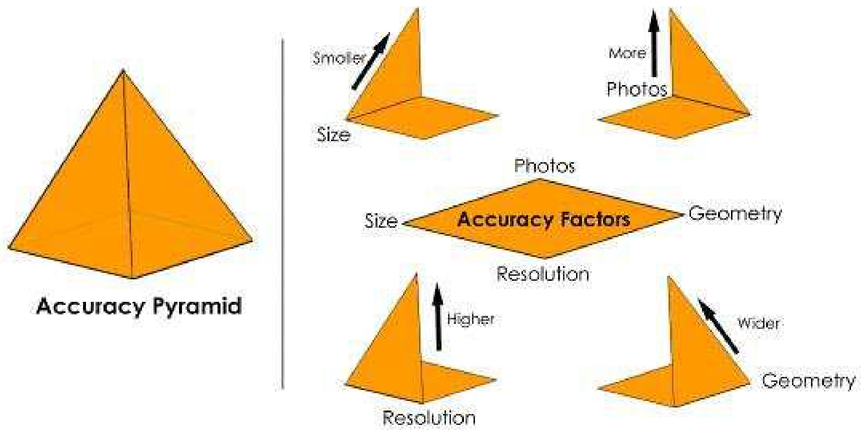


Fig. 2.19. `Accuracy Pyramid for Photogrammetry` [2]

Scaling Photogrammetry Models

When an image is captured, the photogrammetric measurements have no scale size or dimensions data. To scale objects captured in the image, the critical aspect is to know the minimum one measurement value in the visible image which can be used to produce coordinates. If the real coordinates of two or more points in the captured image are known earlier, this could be utilized to compute the dimensions and hence provide the scale of image.

Other method for evaluating the scale of a photo is to use a targeted fixture and measure along the object. The known distance between the target marks can be used to scale the photographs. The most common form of scaling fixtures is scale bars [2].

In order to find scale errors, more than one measurement should be known which are used for scale measurement. When using only one distance to scale measurement and in case if it has error, the entire model will be incorrectly matched to its original dimensions. With other method of using more than one scaling measurement, scaling errors can be identified and avoided [2].

2.2.4 Aspects for Successful Photogrammetry

According to Luhmann [19], to implement photogrammetry successfully in the industry, lot of technical tools and components are desired to provide affordable and efficient system. Here is the list which summaries required tools and corresponding technical troubles:

- **Cameras for imaging** resolution (number of pixels), available lenses, acquisition and data transfer speed, camera stability, synchronization, data compression, etc.;

- **Target and Illumination** representation of object features, target shape and dimensions, the wavelength of light devices, permissions to touch object, illumination power and measurement volume;

- **Imaging Configuration** number of camera stations, desired measurement accuracy, network design, redundancy, robustness, self-calibration ability, datum definition and object control, self-control of orientation and calibration;

- **Image Processing** automation of target recognition and identification, subpixel measurement of target center, multi-image matching approaches, feature tracking, and handling of outlines and scene artifacts;

- **3D model Reconstruction** methods for determination of 3D coordinates (e.g., spatial intersection, bundle adjustment) and error statistics;
- **Data interfaces** integration into CAD/CAM environments, machine and data interfaces, user interaction and displays, etc.;
- **Verification for Accuracy** reference bodies, reference data, standards and guidelines, and acceptance tests [19].

With above mentioned aspects, author says close-range photogrammetry system is a series of complex procedure which include proper design, setup and operation. The applicability of a method depends on technical issues as well as a function of required cost-performance ratio, system assistance, documentation, quality assurance and interdisciplinary skills. Due to this, there are less than 10 professional companies around the globe those provide a system in photogrammetry field probably. Though, the market for optical 3D measurements is significantly increasing and bids promising scenarios for the future [19].

2.2.5 Types of Cameras used for Close-range Photogrammetry

The imaging camera device is an important subsystem of an industrial close-range photogrammetry system. As suggested earlier, selection of an appropriate imaging device is decided by accuracy requirements, resolution, acquisition speed and frame rate, synchronization, amount of data, spectral information, the field of view, image scale, digital interfaces and cost. Generally, it is recommended to use cameras with the highest resolution, imaging speed and accuracy to provide maximum efficiency and productivity concerning system costs and return on investment (ROI) [19].

At present, there is a vast range of cameras and imaging sensors are available with high resolutions (>60 MP), high frame rates(>2000 Hz), pixel sizes variable amid about 1.4 and 15 μm. The author gives an overview of various imaging devices [19].

DSLR Camera

High-resolution DSLR cameras are designed for semi-professional photographic work with a range of exchangeable lenses, high-capacity storage devices and powerful batteries. Depending upon absolute accuracy requirements, these cameras can be regarded as partially metric, with changing interior orientation, even from image to image. DSLR cameras are primarily used for off-line applications, i.e., the measurement of stationary objects. Appropriate cameras in the standard small format are offered by companies such as NIKON, CANON and SONY, whereas medium-format cameras are available from ROLLEI, HASSELBLAD or ALPA. See camera examples in Fig 2.20 [19].

(a) Nikon Camera

(b) Hasselblad Camera

Fig. 2.20. DSLR Camera [20]

Digital video and high speed cameras

Dynamic activities can be captured using digital cameras which have higher frame rates (e.g., video cameras or high-speed cameras). These digital cameras are controlled through a computer workstation interface. In [19], as per author video cameras with stereo-typically 1.3 MP sensors and frame rates of 1030 Hz are used in a variety of applications of photogrammetric on-line systems, tube inspection stereo navigation and robot guidance.Digital high speed cameras are usually equipped with CMOS sensors that enable fast data access, programmable field of view, extremely short exposure times and high dynamic range.

In the market, there are many newly developed cameras with spatial resolutions, see figure 2.21 which is one of the example of High speed camera from PCO dimax [19].

Fig. 3. High-speed camera PCO dimax.

Fig. 2.21. `PCO dimax High-speed camera` [19]

A custom high-speed camera for photogrammetric measurements has been developed by the AICON company, shown in Fig. 2.22, When two or more high speed cameras are used for greater image acquisition rates, synchronization of these cameras is a demanding task. This method is used during dynamic measurement of distortion of car in crash studies. The technology of stereo beam splitting could be utilized as an substitute to the multiple cameras. As shown in the fig. 2.22, it is possible to acquire synchronized stereo photos with only on camera through the use of an optical beam splitter [19].

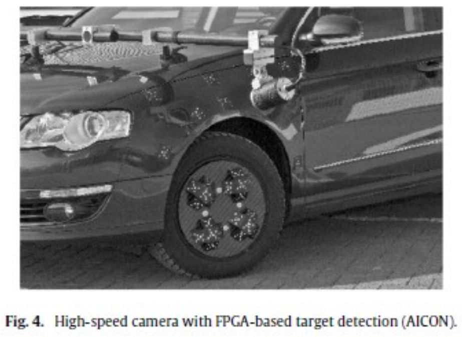


Fig. 4. High-speed camera with FPGA-based target detection (AICON).

(a) AICON high speed camera

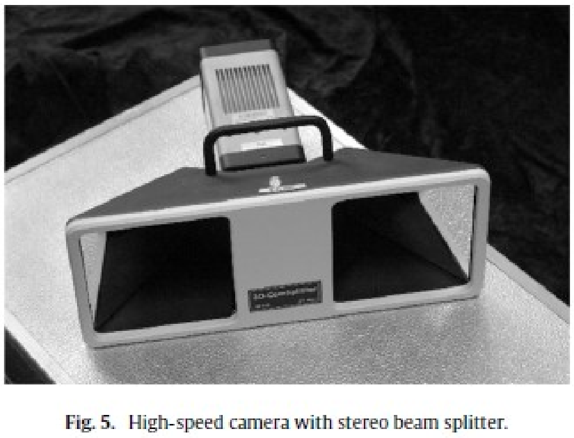


Fig. 5. High-speed camera with stereo beam splitter.

(b) stereo beam splitter

Fig. 2.22. High-speed Camera [20]

Photogrammetric Cameras

There are only a few cameras designed explicitly for close-range photogrammetric applications. The classical metric camera method with steady interior orientation involves high additional effort in terms of optical and mechanical sensor design [19].

The key benefit of these cameras is their guaranteed stability and the resultant reduced need for sporadic or on-the-job calibration, for example in applications where high accuracy is commanded without the technical likelihood for concurrent camera calibration. See figure 2.23 which shows photogrammetric cameras from GSI and Axios which are designed for high accuracy industrial metrology [19].

2.2.6 Which Camera to use for Photogrammetry

In [8], author summarizes comparison between professional digital cameras and cellphone cameras. For Close-range photogrammetry method, cellphone camera or digital cameras are used. Cellphones are moderately cheap and need fewer know how to control. Cellphones have built-in sensors which can collect 3D locations and high-resolution photographs.

(a) Metric Camera GSI INCA 3

(b) Metric video camera AXIOS 3D SingleCam (776 × 582 pixels).

(b) Metric Video Camera AXIOS 3D SingleCam

Fig. 2.23. Photogrammetric Camera [20]

Cell phones are global and easily accessible to the research public. Nowadays cellphones are as accurate as metric cameras which not only save time but money also. When calibrated, mapping with cellphone devices enhances accuracy [8].

In [27], author has inferred that cellphones used in close-range photogrammetry are: fast and useful, efficient, higher accuracy compared with high-resolution Digital cameras. Other researches have concluded that cellphones are adequately accurate for small-scale projects that dont need high accuracy. Todays camera cellphones are proficient enough for creating high-resolution models as there is a lot of technological advancement in this area [8].

Studies predict that cellphones and digital cameras (DSLR) provide almost similar results for 3D construction. However, digital cameras are cost-effective and time -saving which can provide high-quality 3D CAD model. The primary difference between DSLR cameras and cellphones is accessibility and image quality. Cell phone acquires a compressed image (JPEG), while the DSLR captures a RAW uncompressed image. Compression of a photo reduces quality [8].

3. PHOTOGRAMMETRY METHODOLOGY AND TOOLS EVALUATION

If you own the most important tool which is a camera (could be DSLR or cellphone), then it is incredibly affordable to use close-range photogrammetry for industrial purpose. Along with this, you need a photogrammetry toll to reconstruct a 3D object from the photographs you have captured [28].

Similar to other things, photogrammetry software is available with many extra features and functionality. Developers from industry have published commercial solutions for photogrammetric methodologies which could be useful in industrial and manufacturing applications. Though, there are some programs offered for free download [28].

In this chapter, based on website and blogs, the following section is discussing available commercial, professional packages and open source codes used to obtain 3D CAD models from images. Moreover, we discuss which free photogrammetry software tools could be integrated with SOLIDWORKS CAD software using its API. See table 3.1 for Photogrammetry tools available in the market [28, 29].

For integrating photogrammetry tool with SOLIDWORKS, the rationale would be to assess with open source codes which can provide reconstruction results in surface mesh file. This generated mesh file can be used for applications like quality inspection and reverse engineering. This section describes several open source codes used for industrial photogrammetry. See Fig. 3.1 for proposed work-flow for integrated photogrammetry.

In this proposed workflow two dashed boxes, one on the left side which contains test setup, camera calibration and image acquisition has more of physical factors. The other box on right has key steps which will help reverse engineer the objects from their images within SOLIDWORKS.

Mesh Prep Wizard, XTRACT3D, CAD Modelling and Deviation Analysis are add-in features of SOLIDWORKS. In order to complete reverse engineering workflow using photogrammetry using one software, object reconstruction steps (highlighted in translucent yellow) needed to address. Hence in this chapter assesment and selection of photogrammetry tools have been explained.

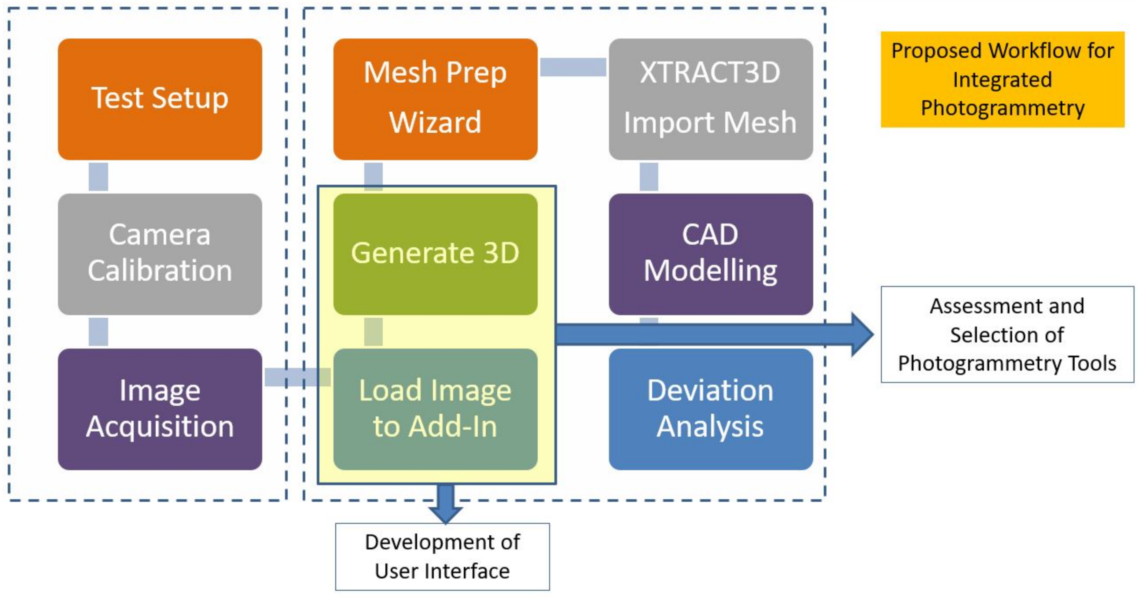


Fig. 3.1. `Photogrammetry to Reverse CAD Workflow`

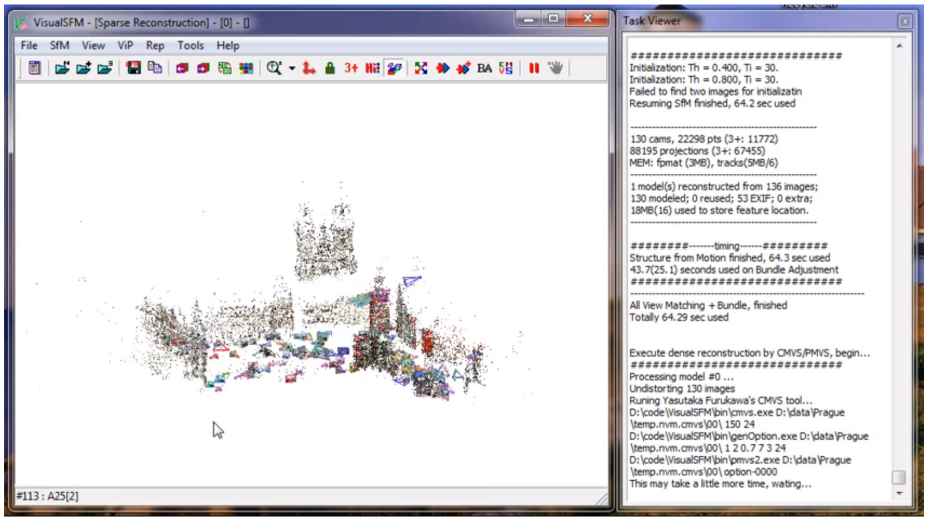
According to all3DP [28], the best free photogrammetric softwares available as follows:

- **VisualSFM** -
  - As the name VisualSFM suggests, point clouds are generated using structure from motion method of the captured object in this photogrammetry tool. This photogrammetry tool gives user options between command-line (CLI) and graphical user interface (GUI).
  - A software engineer at Google named Changchang Wu has developed this product along with other researchers. While using versatile photogramme-

try solution, the user has to set more than a dozen parameters to tweak the results as per requirements.

– The primary step of generating a point cloud needs less than four mouse button clicks. As mentioned in its documentation, this program is available for free for personal, educational non-profit use.

– This program just provides point cloud of captured object; to get surface mesh user has to use another surface generating tool like MeshLab.


VisualSFM - [Sparse Reconstruction] - [0] - []
File SfM View ViP Rep Tools Help
Task Viewer
#113 : A25[2]

```
################################
Initialization: Th = 0.400, Ti = 30.
Initialization: Th = 0.800, Ti = 30.
Failed to find two images for initializatin
Resuming SfM finished, 64.2 sec used

----------------------------------------------------
130 cams, 22298 pts (3+: 11772)
88195 projections (3+: 67455)
MEM: fpmat (3MB), tracks(5MB/6)
----------------------------------------------------
1 model(s) reconstructed from 136 images;
130 modeled; 0 reused; 53 EXIF; 0 extra;
18MB(16) used to store feature location.
----------------------------------------------------

########--------timing-------#########
Structure from Motion finished, 64.3 sec used
43.7(25.1) seconds used on Bundle Adjustment
################################
-----------------------------------------------------------------
All View Matching + Bundle, finished
Totally 64.29 sec used


Execute dense reconstruction by CMVS/PMVS, begin...
################################
Processing model #0 ...
Undistorting 130 images
Runing Yasutaka Furukawa's CMVS tool...
D:\code\VisualSFM\bin\cmvs.exe D:\data\Prague
\temp.nvm.cmvs\00\ 150 24
D:\code\VisualSFM\bin\genOption.exe D:\data\Prague
\temp.nvm.cmvs\00\ 1 2 0.7 7 3 24
D:\code\VisualSFM\bin\pmvs2.exe D:\data\Prague
\temp.nvm.cmvs\00\ option-0000
This may take a little more time, wating...
```

Fig. 3.2. `VisualSFM interface` [28]

- **COLMAP** -

  – COLMAP is available for download from Github for building its (.dll) for free using the command line (CLI). Also, its available with the free version of GUI.

  – Similar to other photogrammetry tools available in the market, COLMAP can reconstruct objects automatically using single-camera or stereo camera setups.

- As researchers develop this program, it has many vastly superior choices. Users those want to produce fast and reliable 3D object mesh; it is not required to use advanced options from this software. Anyhow, they can improve the quality of the mesh.
- Though the package can transfer a 3D mesh, the user can only operate the dense point cloud the viewport. Similar to the case of VisualSFM, cleaning and refining of the 3D mesh can be fulfilled with other programs like MeshLab.

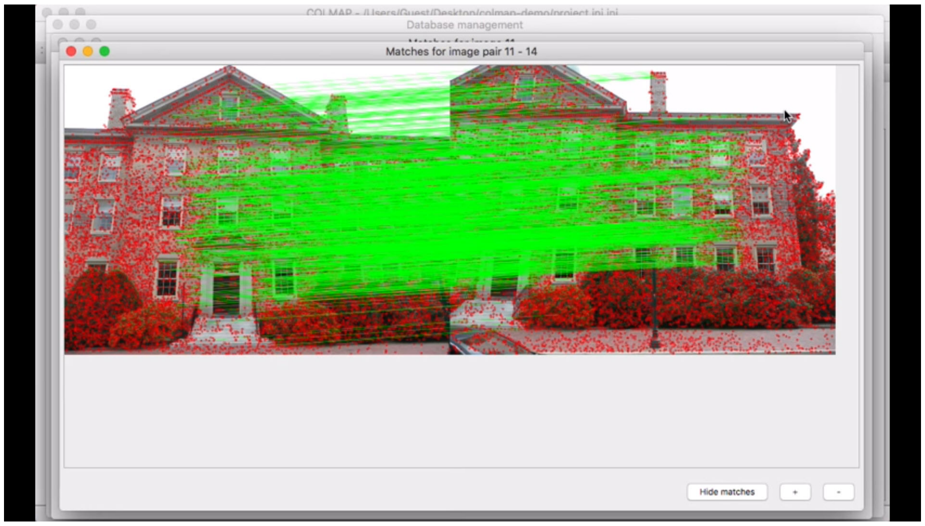


Fig. 3.3. `COLMAP interface` [28]

- **MicMac -**
  - The French National Geographic Institute have developed open source code for photogrammetry methodology which is named as MicMac.
  - The French National Geographic Institute have developed open source code for photogrammetry methodology which is named as MicMac. As its erudite feature-set, it is principally matched to the expert or academic users, but it is also handy to universal users. MicMac has been developed as a

product after continuous research and academic project and it is a versatile program which has been applied in fields like forestry, cartography, cultural heritage preservation, environmental protection and private businesses.

  - MicMac can generate 3D models and orthographic images of the real-life object. Apart from this, the photogrammetry software can deal with any object and any scale you insert in the program. Micmac is not only capable of surveying large plots of land but also of scanning small objects.
  - With editing tools suggested, sometimes portray MicMac as a possible answer for metrology and site surveying. Detailed guides and custom tutorials are featured on MicMacs wiki page which can guide users quickly.

- **Meshroom -**
  - Another open-source code which is free for photogrammetry is Meshroom built on AliceVision framework. This algorithm is compiled with a simple node-based methodology which connects necessary steps to reconstruct images to 3D object.
  - Many of the other program available steps would have to be operated individually; however, in the case of Meshroom, all steps form a pipeline of nodes. A user has to click Start from the top of the workspace for program initiation.
  - Along with this simple pipeline, a user has the option to tweak parameters from its elaborate pipeline to get the result as per requirement.
  - Along with this simple pipeline, a user has the option to tweak parameters from its elaborate pipeline to get the result as per requirement.
  - For illustration, as per the requirement in mobile gaming or high-resolution rendering, the user can edit the texture step to set the resolution of the texture map. The more time the user spends with free photogrammetry tool, user can achieve custom, better results.

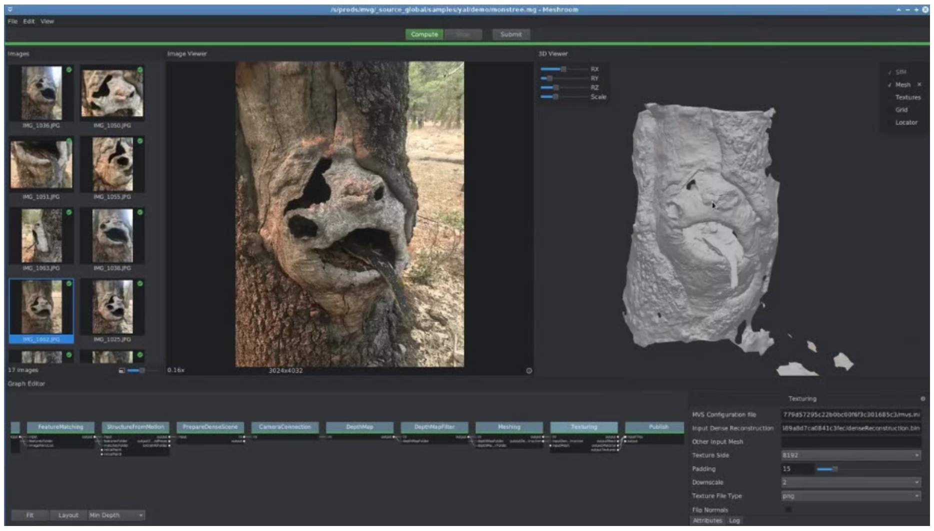
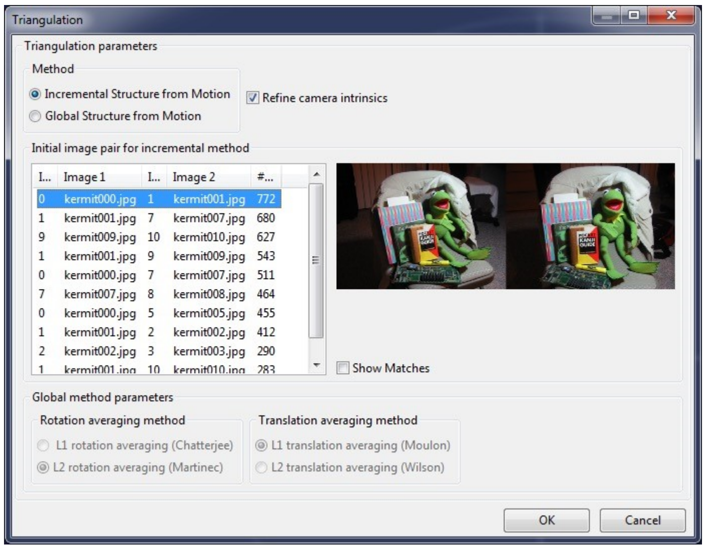


Fig. 3.4. `Meshroom interface` [28]

- **Regard3D** -
  - Regard3D is a open source photogrammetry code that is based on the structure-from-motion methodology to reconstruct 3D models. Though Regard3D is free of cost, it is still a compelling software in terms of reconstruction.



Triangulation

Triangulation parameters

Method

[✓ Incremental Structure from Motion]
Global Structure from Motion

[✓ Refine camera intrinsics]

Initial image pair for incremental method

I...	Image 1	I...	Image 2	#...
0	kermit000.jpg	1	kermit001.jpg	772
1	kermit001.jpg	7	kermit007.jpg	680
9	kermit009.jpg	10	kermit010.jpg	627
1	kermit001.jpg	9	kermit009.jpg	543
0	kermit000.jpg	7	kermit007.jpg	511
7	kermit007.jpg	8	kermit008.jpg	464
0	kermit000.jpg	5	kermit005.jpg	455
1	kermit001.jpg	2	kermit002.jpg	412
2	kermit002.jpg	3	kermit003.jpg	290
1	kermit001.jpg	10	kermit010.jpg	283

Show Matches

Global method parameters

Rotation averaging method

L1 rotation averaging (Chatterjee)
[✓ L2 rotation averaging (Martinec)]

Translation averaging method

[✓ L1 translation averaging (Moulon)]
L2 translation averaging (Wilson)

OK Cancel

Fig. 3.5. `Regard3D interface` [28]

  - Setting in and getting used to with Regard3Ds settings, parameters to tweak in order to achieve desirable results need a a considerable amount of time.
  - The Regard3D program even contains inclusive gears for editing the point cloud before producing a 3D surface mesh. For total beginners in the photogrammetry sector, Regard3D gives good insights. Its website has all the documentation and custom tutorials user might need to get started quickly.

- **OpenMVG** -
  - OpenMVG is an open source library for photogrammetry solution which uses the Multiple View Geometry (MVG) methodology.
  - The limelight generally appears to be on Structure from motion (SFM) methodology of photogrammetry and many integrated tools associated with this.
  - To use OpenMVG photogrammetry pipeline, first of all, a user needs to compile the library for usage, OpenMVG needs some experienced computer tech user to get up and running on operating systems like Windows, macOS or Linux.
  - As OpenMVG designed for simplicity and maintainability, this tool is accessible and can be associated with other photogrammetry tools.

3.1 Software Evaluation

In this previous section, introduction about free open source codes for photogrammetry has been made . Now, its time to evaluate these softwares based on common example and factors associated with.

As shown in the table 3.1, all of these photogrammetry software packages produce a 3D data set of points, which is referred as 3D point cloud. From this list of packages, most of the programs generate triangulated surfaces using previously created point cloud. Generally, in order to get maximum benefit out of these ope source packages in any research applications, the software program should be associated with preview window. This can show the photographic images with the created surfaces.

According to all3dp [28], not all of these open source codes generate point cloud data. Packages like COLMAP, VisualSFM and OpenMVG are specifically programmed to return output in 3D point cloud with discrete points. Other tools like Meshroom, MicMac and Regard3D generate optimized 3D surface meshes from the point cloud data.

Another method used for photogrammetry projects to generate the 3D surface is by exporting 3D point cloud data into separate 3D CAD system (3rd party application like - MeshLab) to generate surface mesh as per users' requirement to corresponding projects.The 3rd-party CAD system could also be used not only for manipulation, measurement the surface but also for determining the orientation of planar features [30].

According to shaffner [30], author has discussed difficulties faced by researchers and complexities associated with exporting data sets, surfaces to another CAD platform for analysis. Experienced CAD users around the world using high-end computer systems, this method is generally preferred or acceptable. Thus, here comes the evaluation criteria for mentioned software packages are included in the following considerations:

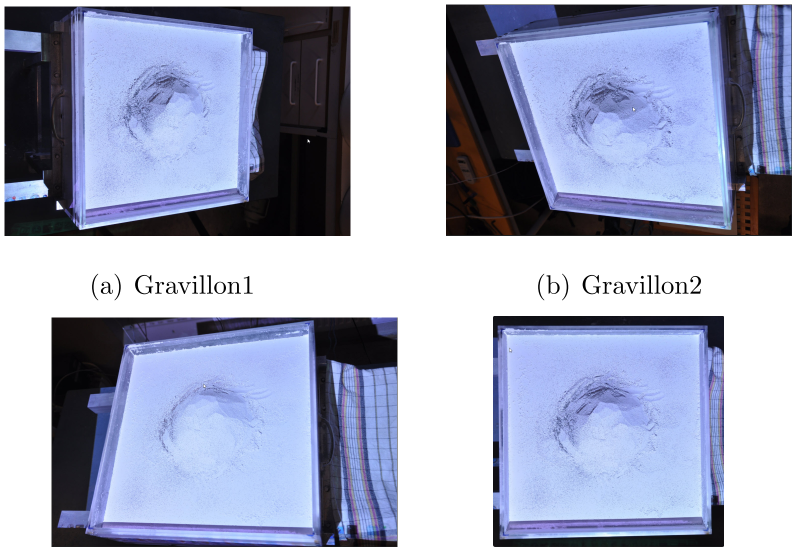
- Efficient image processing procedure to generate 3D object reconstruction data.
- Minimal operation time and cost
- Data export capabilities to other CAD systems
- Built-in data post-processing tools (or is additional software needed)

During this photogrammetry research, referring table 3.1, three open source codes have been selected for evaluation.

Selection criteria for photogrammetry tools were based on the cost of tool and complete work-flow until surface reconstruction. These categorised tools are known as Regard3D, Meshroom and MicMac.

For evaluation, a common photoset ”Gravillon” is used and then using tutorial documentation results were obtained using each photogrammetry tool. This ”Gravillon” dataset is light by design which has 4 images. This data-set is explicitly designed for beginners with light photogrammetry background by MicMac engineers.

L.Girod developed Gravillon dataset who is from the University of Oslo, Norway. This dataset was captured to model a volcano created by O.Galland. See figure 3.6 where this ”Gravillons” dataset has been displayed [31].



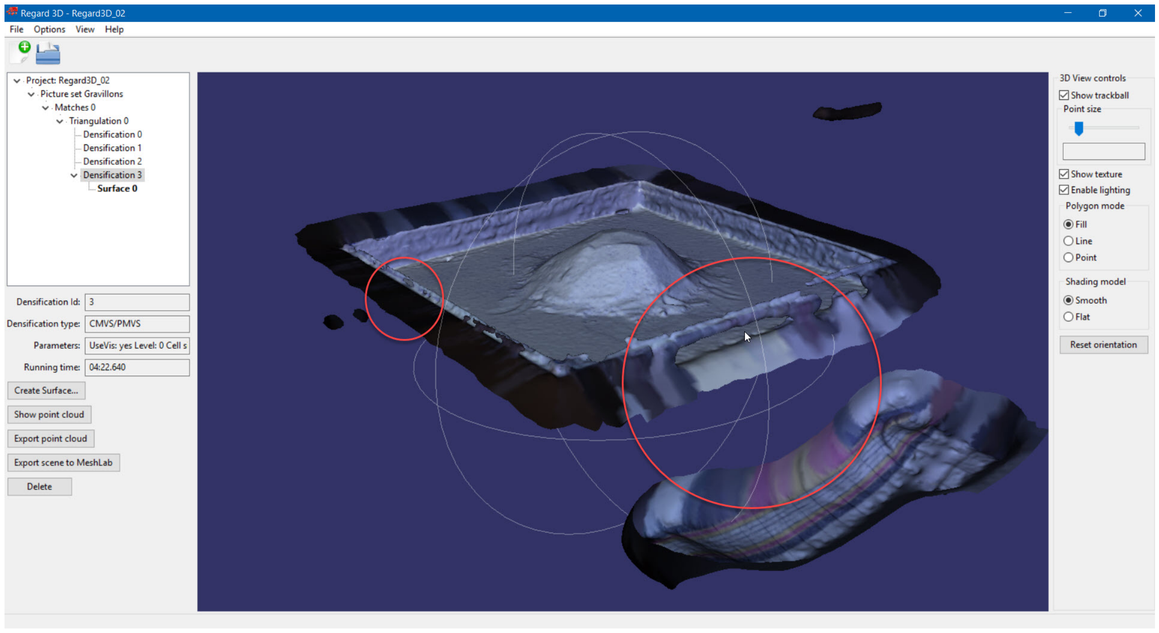
(a) Gravillon1 (b) Gravillon2

(c) Gravillon3 (d) Gravillon4

Fig. 3.6. Gravillon Dataset by MicMac [32]

3.1.1 Regard3D

While operating on Regard3D, image folder could be selected and output folder path could be determined at the start. After uploading the gravillon dataset, image matches can be computed using the ”compute matches” button.



Densification Id: 3
Densification type: CMVS/PMVS
Parameters: UseVis: yes Level: 0 Cell s
Running time: 04:22.640

[✓] Show trackball
[✓] Show texture
[✓] Enable lighting
Polygon mode: [✓ Fill] Line Point
Shading model: [✓ Smooth] Flat

Fig. 3.7. `Regard3D Output`

Once the image pair match has been computed, subsequently triangulation could be assessed using global Structure from motion methodology. The output of this step is in sparse point cloud format.

Followed by this surface is created using ultra-small size. This entire process right from image matching till surface creation took 4min 53 sec.

As you can see in figure 3.7, on one of the side walls has a hole in it which can be later post-processed, but this increases additional operational time. Also, sidewalls are not correctly reconstructed using the available images pair matches.

After performing SFM and surfacing operations, these file data of photogrammetric result can be exported in *.ply and *.obj file formats. Regard3D has its built-in preview window where the user could analyze the final output of a particular setting in the same graphics area. If required, the user might change and tweak with various parameters. The general observation about Regard3D says it needs considerable time to understand different parameters to get even low simulation result.

3.1.2 Meshroom

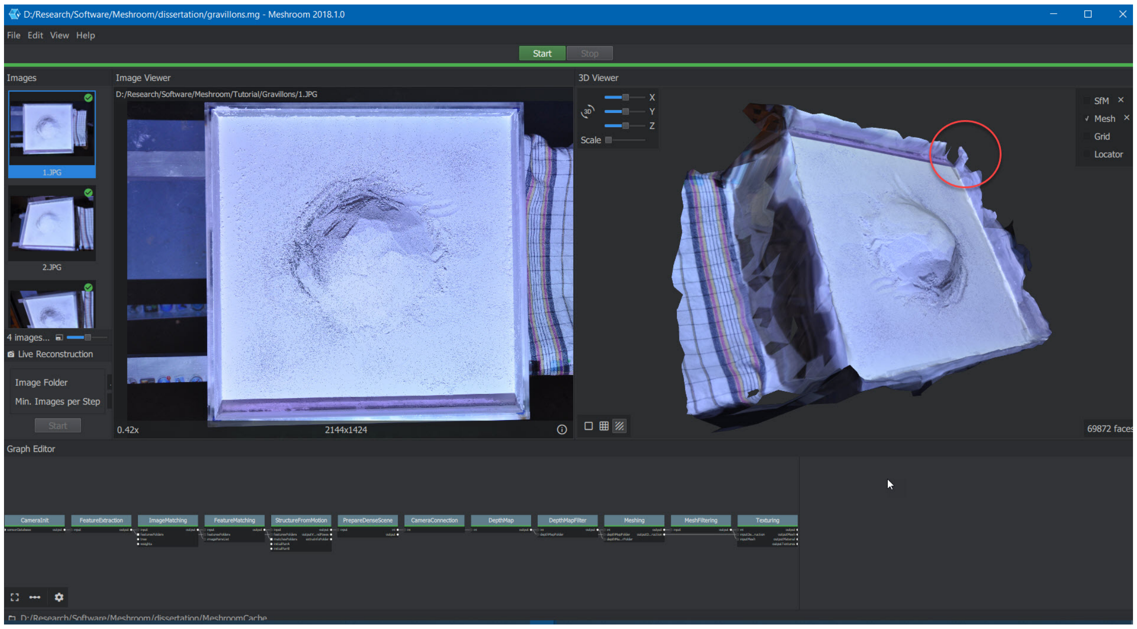


Fig. 3.8. `Meshroom Output`

While working with Meshroom, image folder could be selected and output folder path could be determined at the start. Meshroom has an option to drag and drop in the image log section. After uploading the gravillon dataset, it just has a shortcut, standard method; the user has to hit Start and meshroom generates corresponding files automatically.

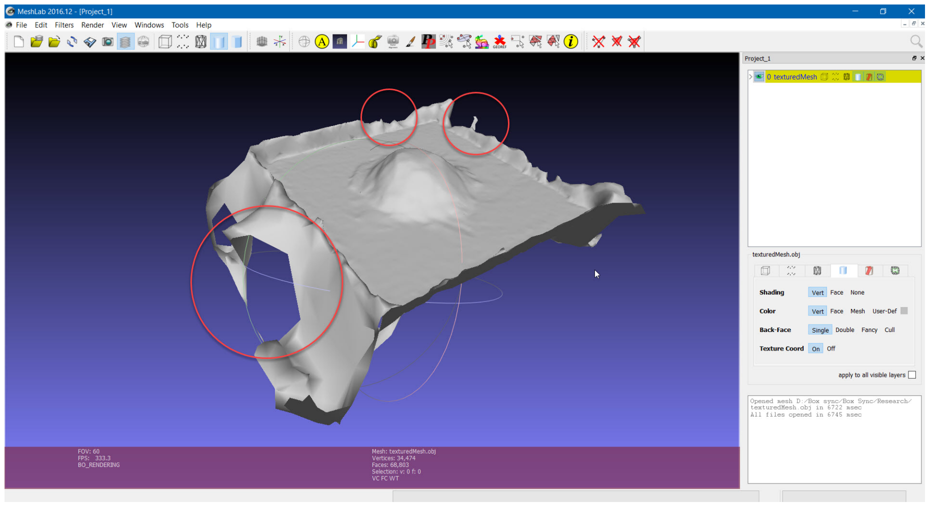
Once the images have been inserted, subsequently photogrammetric reconstruction generated using global Structure from motion methodology. The output of this entire process is in the surface mesh file. This entire process right from image insertion till surface creation took 1min 46 sec.

As you can see in figure 3.8, comparing the result with Regard3D shows that Meshroom has a better result with less post-processing. One of the side walls has a hole in it which can be later post-processed, but this increases additional operational time.

After performing automatic photogrammetry method, this file data of result can be exported in *.ply and *.obj file formats with textures.

Meshroom has its built-in preview window where the user could analyze the final output, SFM, camera views and mesh. If required, the user might change and tweak with various parameters, but the user can't save different settings for analyzing.

3.1.3 MicMac


MeshLab 2016.12 - [Project_1]
File Edit Filters Render View Windows Tools Help
Project_1
0 texturedMesh
texturedMesh.obj

Shading [✓ Vert] Face None
Color [✓ Vert] Face Mesh User-Def
Back-Face [✓ Single] Double Fancy Cull
Texture Coord [✓ On] Off
apply to all visible layers ☐

```
FOV: 60
FPS:   333.3
BO_RENDERING
```

```
Mesh: texturedMesh.obj
Vertices: 34,474
Faces: 68,803
Selection: v: 0 f: 0
VC FC WT
```

```
Opened mesh D:/Box sync/Box Sync/Research/
texturedMesh.obj in 6722 msec
All files opened in 6745 msec
```

Fig. 3.9. `MicMac Output`

While working with MicMac, as per the tutorials provided, command prompt with mentioned lines has to be used in order to start this photogrammetric method. In the existing image folder batch file need to create with a bunch of command line operations in it and output folder path is the same image folder.

As this is the Command line based open source code, everything needs to build first using terminal operations. Thus, the user must know how to use command prompt or terminal (in case of Linux OS) operations and commands. More than that, the user should know about passing arguments to the photogrammetric operations. Meshroom has an option to drag and drop in the image log section.

After uploading the gravillon dataset, it takes out a significant amount of time, as the user has to enter command line instructions at every step and MicMac generates corresponding files after completion in the same folder.

The output of this entire process is in the surface mesh file. This entire process right from image insertion till surface creation took 16min 38 sec. As you can see in figure 3.9, comparing the result with Regard3D and Meshroom shows that MicMac has a little poor result. One of the side walls is badly reconstructed which can be later post-processed, but this increases additional operational time.

After performing automatic photogrammetry method, this file data of result can be exported only *.obj file formats without textures. MicMac doesn't have its built-in preview window as this command line program. If required, the user might import this surface file in MeshLab CAD mesh analyzing software.

The general observation about Meshroom says it needs significant time to understand the command line procedure to get even low simulation result.

3.2 Scaling and Comparing Results from Photogrammetry Tools

In order to verify the efficiency of MicMac, Meshroom and Regard3D, output from these tools were compared with scan mesh file of same Gravillons object. This scan mesh is shown in Fig. 3.10.

For point cloud to point cloud comparison, all mesh files of gravillons are exported from mesh file type to point cloud file type. Using scan point cloud as standard reference, one by one all photogrammetry tools' output files are compared. The "Compute C2C" feature was used for calculation of absolute distances of sample cloud to reference cloud based on nearest neighbor.

The output mesh files were then exclusively brought into Cloud Compare which is an open-source 3D point cloud software. Upon import the photogrammetry mesh data sets had an alternate scale and direction. In request to compare them to the scan mesh data, the photogrammetry mesh files expected to both be scaled and adjusted.

Cloud Compare is a software bundle fit for bringing in, adjusting and analyzing distances between two different point clouds. Cloud Compare additionally has point to point estimation tools and the capacity to duplicate or scale whole point clouds. Both of these highlights were used so as to scale the software data sets.

Subsequent to scaling, the photogrammetry point cloud arrangements were adjusted to the scan point clouds utilizing Cloud Compare. This was achieved utilizing at least three common points. Perceiving that a the ineffectively adjusted dataset could create erroneous outcomes during examination, every arrangement was investigated for precision outwardly and quantitatively. Cloud Compare computes a root mean square (RMS) worth dependent on the arrangement points picked. At the point when a bigger number was accounted for by the software, extra points were picked in exertion to diminish this worth and accomplish an increasingly exact outcome. The arrangements were outwardly examined by flipping on and off the other data set from various vantages to check whether a visual move happened. In the event that the datasets appeared to outwardly be balanced in interpretation or turn,extra or interchange arrangement points were picked.

When a decent arrangement was accomplished, the photogrammetry based point cloud was physically sifted along these lines to that of the check data or scan data. This was done physically by evacuating recognizably errant points or islands of data points from the point cloud. In examples where the subsequent data seemed to contain detectably errant points, yet no reasonable line could be resolved for isolating the errant points, no points were expelled from the data set. These points could be viewed as even more a promontory than an island. CloudCompare (v. 2.6.2) has a separating alternative called 'SOR' or Measurable Outlier Removal.

This filter was kept running on all photogrammetry data sets with default software estimations of '10' for the number of points utilized in mean separation estimation and '1.00' for the standard deviation multiplier limit. See Fig. 3.11 which shows point cloud comparison between Regard3D output and scan file.

(These distances are shown in units because Cloud compare doesn't provide actual file's unit. Reference and sample files are scaled with maximum overlap.) Statistical comparison is shown in Table 3.3.

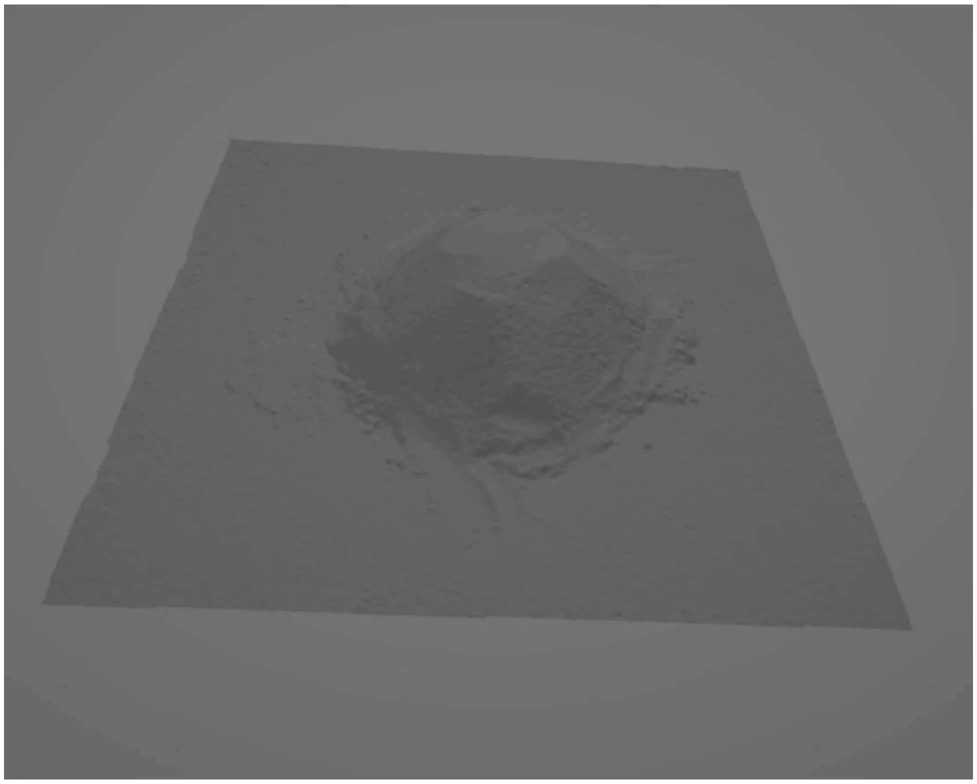


Fig. 3.10. `Scan Mesh of Gravillons`

3.3 Selection of Tool for Integration

As discussed earlier, Regard3D, Meshroom and MicMac have their own advantages and disadvantages. All of these open source codes provide information about all steps of photogrammetry calculation and supply statistical testing of the derived parameters.

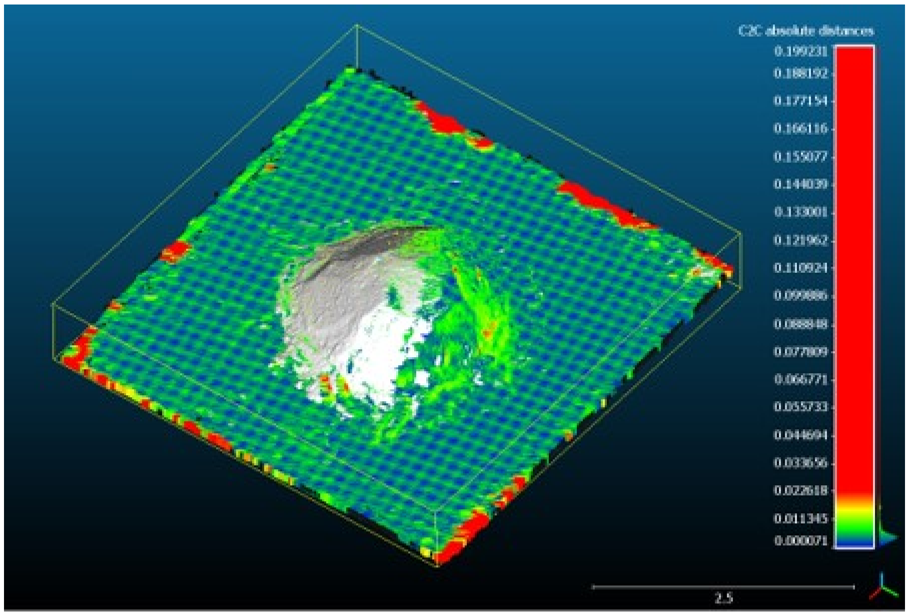


Fig. 3.11. Point Cloud to Point Cloud comparison between Scan and Regard3D output

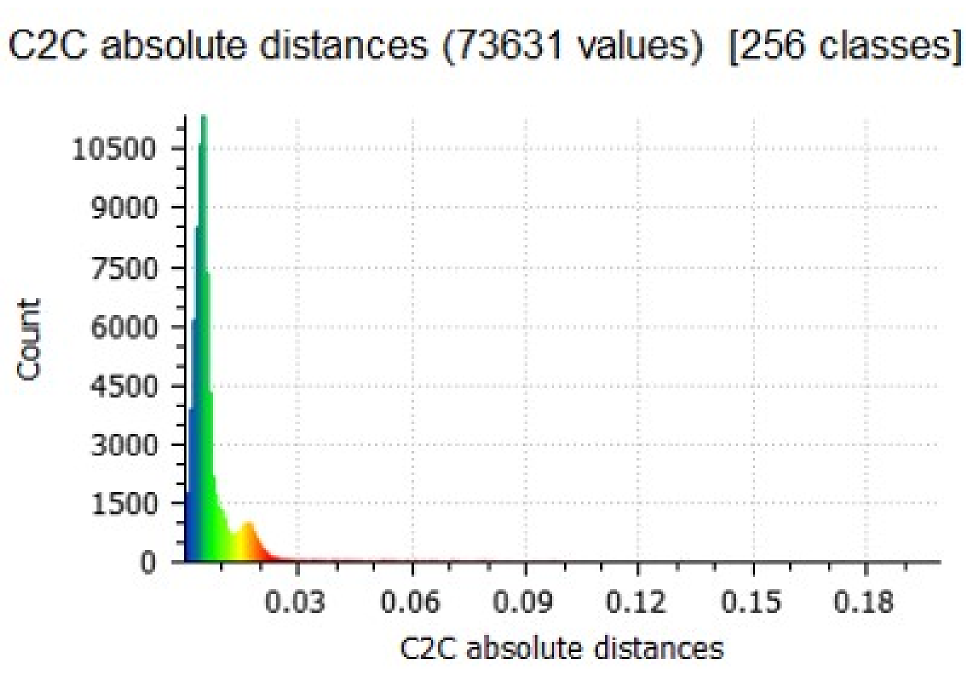


Fig. 3.12. Normalized Distribution of absolute distances for Regard3D

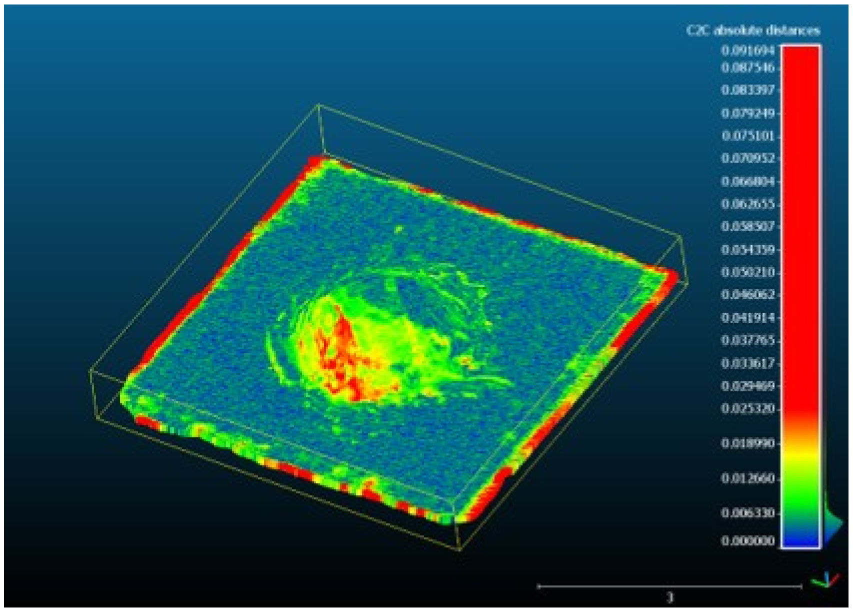


Fig. 3.13. Point Cloud to Point Cloud comparison between Scan and Meshroom output

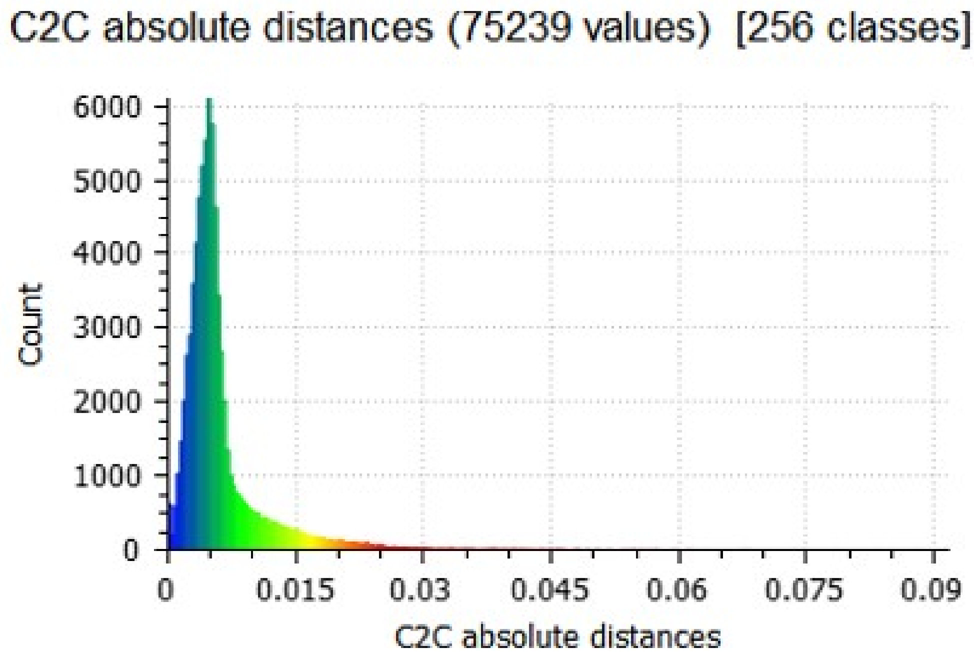


Fig. 3.14. Normalized Distribution of absolute distances for Meshroom

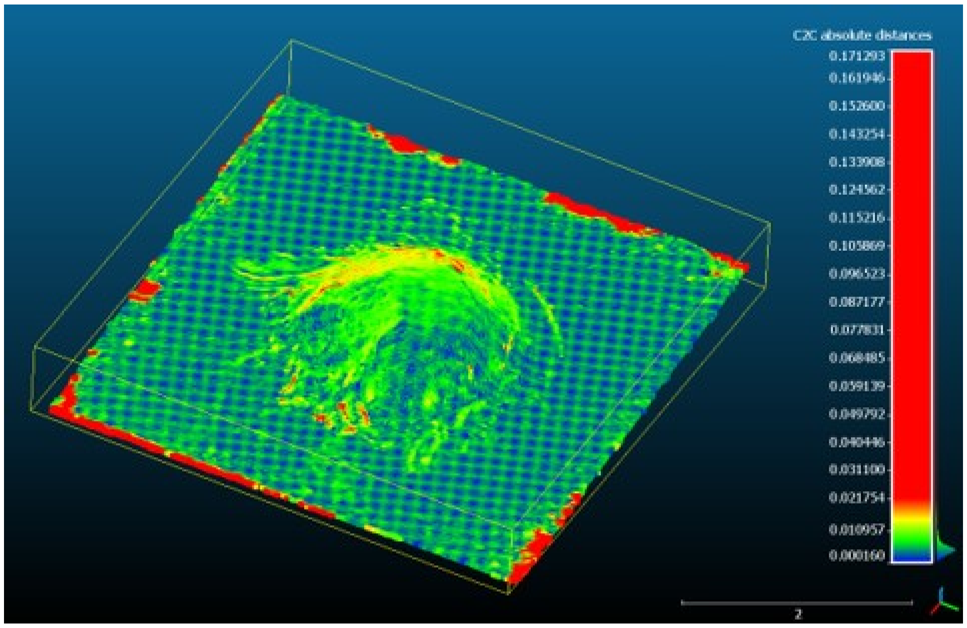


Fig. 3.15. Point Cloud to Point Cloud comparison between Scan and MicMac output

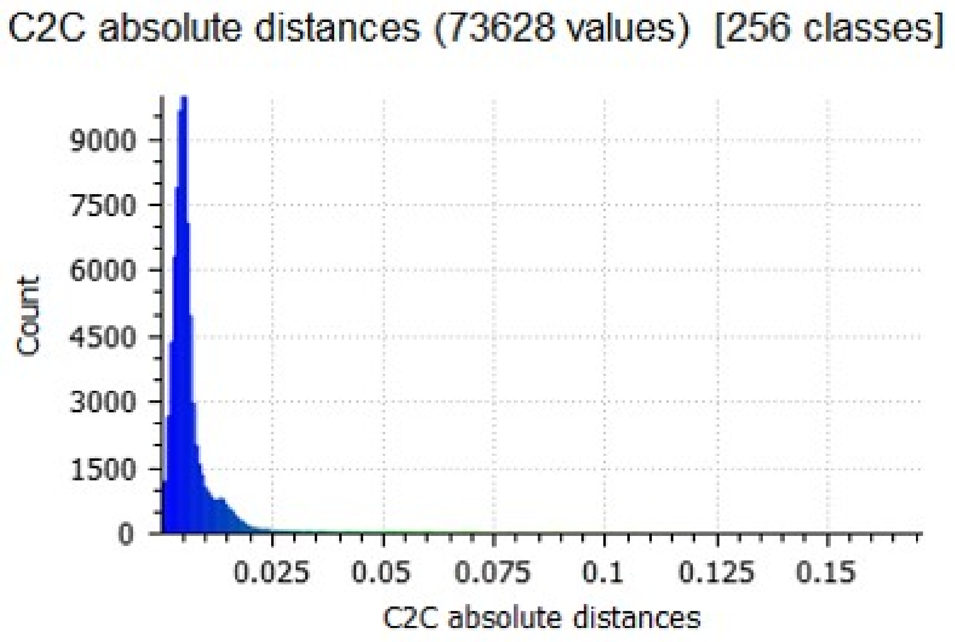


Fig. 3.16. Normalized Distribution of absolute distances for MicMac

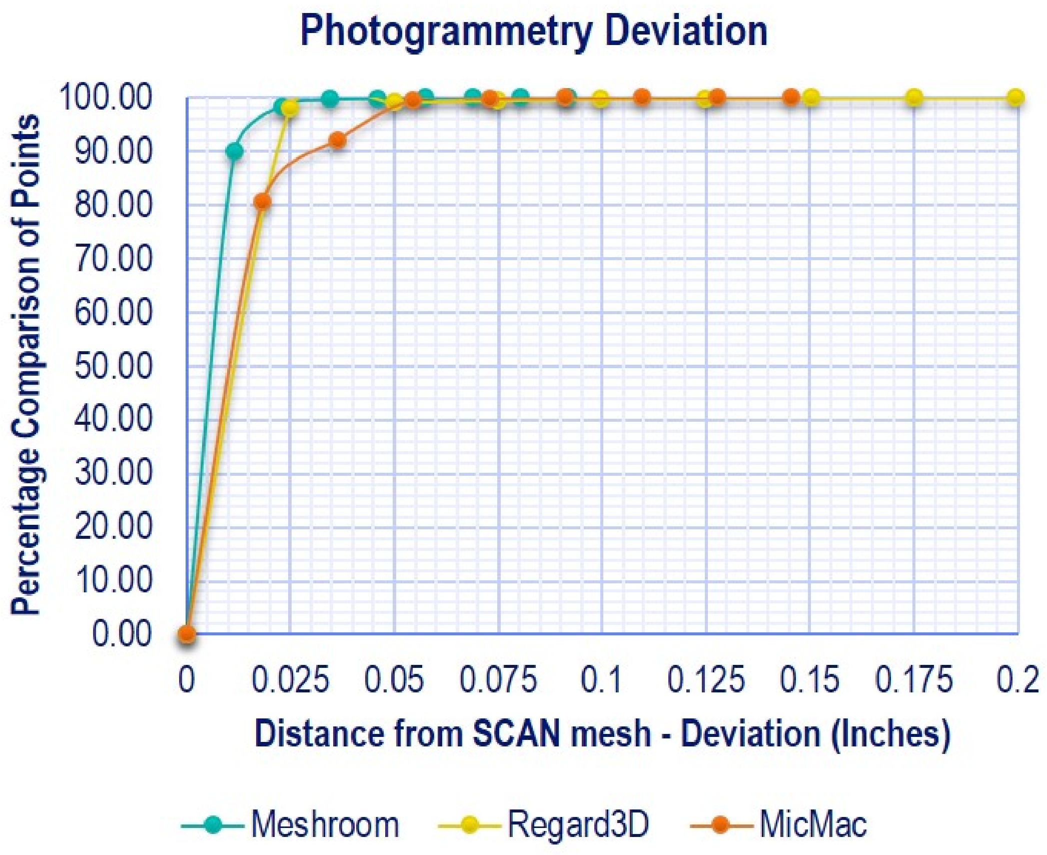
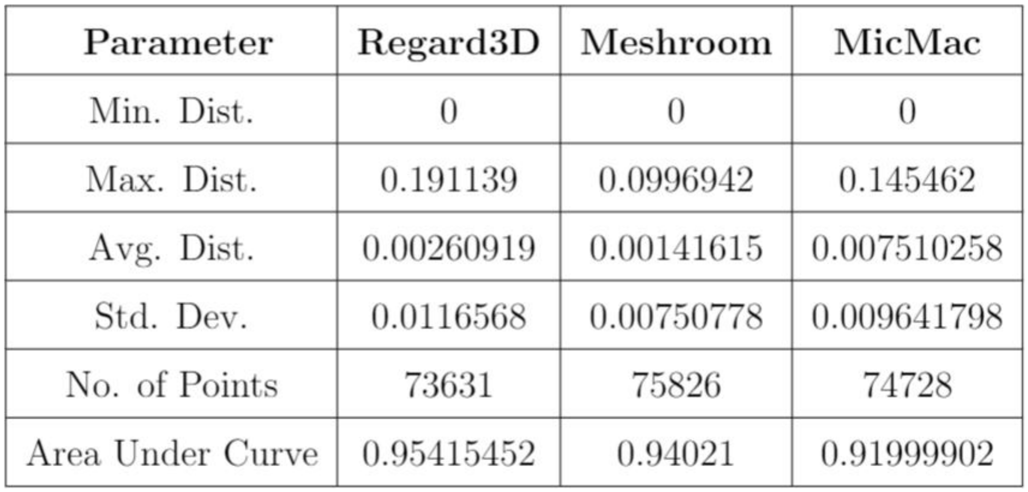


Fig. 3.17. `Comparison of c2c deviation of photogrammetry tools`

It's visible that MicMac photogrammetry tool is particularly more sensitive to noise than Meshroom and Regard3D; this is the reason perhaps it throws errors while capturing every minute detail [32]. Counter-intuitively, these error with MicMac results into missing out on huge data of captured image scene which is not ideal requirement for integration.

On the point note, MicMac does not create intermediate files of different procedure which helps saving storage space in the user's system. During cloud compare analysis, its been discovered that MicMac has generated lesser number of points in the cloud than Meshroom and Regard3D (Refer Fig 3.17). Also, when scales are converted in 0 to 1, the area under curve for MicMac in graph is standing last. Thus the model



Parameter	Regard3D	Meshroom	MicMac
Min. Dist.	0	0	0
Max. Dist.	0.191139	0.0996942	0.145462
Avg. Dist.	0.00260919	0.00141615	0.007510258
Std. Dev.	0.0116568	0.00750778	0.009641798
No. of Points	73631	75826	74728
Area Under Curve	0.95415452	0.94021	0.91999902

Fig. 3.18. `Statistical Comparison of Tools`

reconstructed is inferior in quality. It took more time than other tools, as MicMac is based more on CPU than GPU. There is no doubt about changing parameters in the tool's setting in MicMac, but this program is hard-coded which makes it difficult to manipulate. In [29], the author has rated various photogrammetry software on qualitative and user compatibility basis and MicMac scored 2 out of 10 which makes it last in standings. Henceforth, eliminating MicMac from integration consideration would be a logical decision.

Now for SOLIDWORKS integration, Regard3D and Meshroom are the best available options. As mentioned in Fig.3.17. Regard3D's area under curve (0.95415452) is little more than Meshroom's area under curve (0.94021). But Regard3D has higher values of maximum deviation, average deviation and standard deviation. Apart from that, Meshroom is creating more number of points in the cloud than Regard3D.

As discussed earlier, meshroom needs less post-processing operation than Regard3D to achieve approximately correct reconstructed object from the same set of images. Another deciding factor for integration is the time of operation for the photogrammetric method. Meshroom has produced a moderately better quality of object within less time of computational time of 1 min 46 sec to Regard3D's 4 min 53 sec.

In [29], the author has summarized about troubles getting correct, appropriate results with Regard3D and rated Regard3D 5 out of 10. During this evaluation, Meshroom has come out as a simple but versatile tool which has options between automatic and advanced settings.

As you can see in fig.3.7, meshroom photogrammetry tool has several nodes in its pipeline. Each of those nodes could be altered as per requirement. Also considering, computational time for same simple dataset, ability to match scan data Meshroom is better in all 3 photogrammetric tools discussed. Therefore, in this research, the methodology of Meshroom photogrammetry tool has been studied and later it's integrated with SOLIDWORKS.

Table 3.1.
Photogrammetry solutions available in the market (year 2019)[29,30]

Name	**OS**	**Price**	**Interface**	**Vendor**	**Surface**
Bentley ContextCapture	Windows	Paid	GUI	Bentley	YES
iWitnessPro	Windows	Paid	GUI	Photometrix	YES
Photomodeler	Windows	Paid	GUI	Photomodeler Technologies	YES
Autodesk ReCap	Windows	Paid	GUI	Autodesk	YES
RealityCapture	Windows	Paid	GUI	CapturingReality	YES
Metashape	Windows, macOS, LINUX	Paid	GUI	Agisoft	YES
3DF Zephyr	Windows	Paid	GUI	3DFLOW	YES
COLMAP	Windows, macOS, LINUX	Free	GUI & CLI	COLMAP	NO
Meshroom	Windows	Free	GUI	Alicevision	YES
MicMac	Windows, macOS, LINUX	Free	CLI	MicMac	YES
Regard3D	Windows, macOS, LINUX	Free	GUI	Regard3D	YES
VisualSFM	Windows, macOS, LINUX	Free	GUI	VisualSFM	NO
OpenMVG	Windows, macOS, LINUX	Free	CLI	QGIS	NO

Table 3.2.
System used for Photogrammetry Softwares [32]

System information	**Specifications**
Processor	8th Gen. Intel Core i7-8750H Processor(up to 4.1GHz)
Chipset	Intel HM370
Operating System	Windows 10 Home 64bit English
RAM Memory	16GB, 2x8GB, DDR4, 2666MHz
Hard Drive	128GB SSD + 1TB SATA Hard Drive
Video Card	NVIDIA GeForce GTX 1060 6GB GDDR5 Max-Q Design
Display	15.6-inch FHD (1920 x 1080) IPS Anti-Glare

4. PHOTOGRAMMETRY ALGORITHM

As discussed in the previous chapter, Meshroom is built on Alice vision framework with a simple node-based methodology which connects necessary steps to reconstruct images to the 3D object.

The Default pipeline of meshroom has a total of 12 nodes which results in a complete reconstructed 3D model from its images. Let's walk through these default nodes of the photogrammetry one by one -

As per Alicevision Meshroom' s documentation [33]-

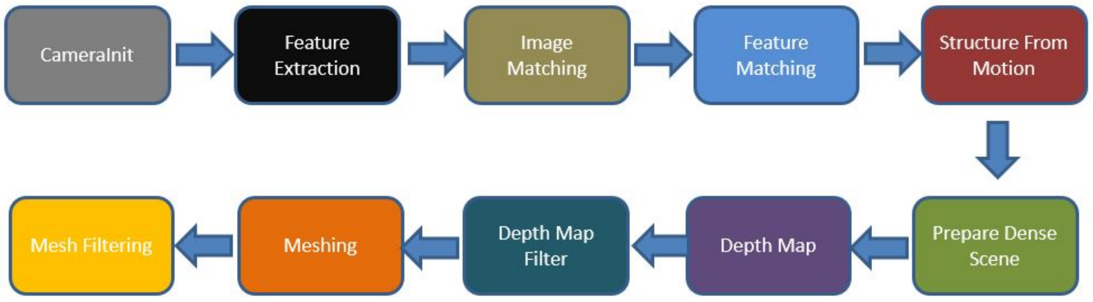


Fig. 4.1. `Meshroom Photogrammetry Pipeline`

1. **CameraInIt** -

   - The very first step generates a (.SFM) file. For photogrammetry, camera/sensor is essential equipment to record images. Every camera has parameters like camera/sensor type, focal length, size of images can be captured and other parameters.
   - While using photogrammetry tool, it is recommended to store camera information or data which can be used for custom camera calibration in later

stages. Thus in this CameraInit stage, a file that could store all necessary data about a variety of cameras need to be stored in an array of matrix format.

- In computing, JSON (JavaScript Object Notation) is an open standard file format that utilizes human-readable text to transmit data objects consisting of attribute-value pairs and array data types (or any other index value) [34].
- SFM files are JSON files that store Camera Size, Sensor information, distortion coefficient camera extrinsic matrices, bundle points.

2. **Feature Extraction**

- The main objective of this node in photogrammetry is to extract characteristic sets of pixels that are, to some level, unaffected to altering camera viewpoints throughout image capturing. Therefore, a feature in the image scene should have similar descriptions of feature in all images.
- The most familiar feature recognition method is the SIFT (Scale-invariant feature transform) system. Regardless of rotation, scale and translation, SIFTs preliminary motto is to identify discriminative pixel sets in the first photo which could be compared with discriminative pixel sets in the second photo.
- The extracted pixel sets are centered at stable points of interest as this is a relevant aspect that only occurs at a specific scale. Keeping this in mind, the gist of the algorithm is that to some extent one could be benefited from SIFTs invariance property to address image transformations which occur when viewpoints of the camera are changed during image capturing procedure.
- By computing a pyramid of scaled-down images, one image can be represented at various scales. Scale-space maxima of the Laplacian representation is calculated by the SIFT method. Laplacian representation is a

precise image energy-based representation of the image. SIFT uses nominated Gaussian differences. Point of interest generally linked by these maxima. It then samples for each one of these maxima a square image patch whose origin is the maximum and x-direction is the dominant gradient at the origin. Detailed information is linked to each keypoint [33].

- The detailed description is characteristically stored in 128 bits. This description contains statistics of gradients calculated in pixel patches about the keypoint. Region size is determined by the keypoint scale while the orientation is determined by the principal axis determines [33].

3. **Image Matching**

- After feature extraction by SIFT method, in this step, the objective is to match images that observing similar parts of the scene. In this, image pair is created by image recovery methods to find images that share nearly the same information without resolving every feature matches in minutiae.
- The objective is to restructure the image in a dense image descriptor which calculates the distance between all dense images descriptors proficiently.
- The vocabulary tree approach is one of the most common technique to create this image descriptor. Once all extracted features descriptors inserted into this method, it categorizes their descriptors after comparison to the ones on every node of this tree [33].
- Every leaf on this tree is associated with one feature descriptor. The index of conforming leaf can store this feature descriptor in the tree. Then, the image descriptor is signified by this group of denoted leaves' indices [33].
- It is now conceivable to check if different images share similar information by comparing these image descriptors with others.
- The output feature extraction folder(s) and descriptors are inserted as input to Image matching node. During this process, a vocabulary tree

file (.tree) provided in Meshroom's Github folder can be used for indexing purposes. When the user computes node by node results, weight file can be given a name; otherwise, the weights are computed on the database built with the provided set [33].

- To get an accurate 3D model from image reconstruction, a higher number of image capturing is advised. In this image matching step, a minimal number of images to use the vocabulary tree can be set between 0 to 500 images. Any number of images less than the set threshold parameter are computed for matching combinations.
- Along with this parameter, the number of descriptors user load per image can be limited and retrieval of the number of matches can be set.

4. **Feature Matching**

- The goal of this node is to match all features between appropriate and qualified image pairs.
- Firstly, this node accomplishes photometric matches between the set of descriptors using 2 input images. For every feature in image A, a list of contender features in image B is obtained. To remove bad matching candidates, Meshroom assumes that theres only one right match in the other image [33].
- Thus for each feature descriptor on the 1st photo, this step look for the two nearby descriptors and uses a relative threshold between them. This postulation eliminates features on repetitive assembly but has exhibited to be a robust standard [33].
- This node gives an output of a list of feature matching contenders validated by only a photometric criterion. Identifying two closest descriptors in a 2nd image for every feature is intense for computation with the brute force method; though many researchers have optimized this existing algorithm. [33].

- This node in Meshroom then uses the features positions in the images to make geometric filtering by using epipolar geometry in an outlier detection framework called RANSAC (RANdom SAmple Consensus) [33].
- Later, this node arbitrarily select a minor set of feature correspondences and compute the fundamental (or essential) matrix; then it checks the number of features that authenticate this model and iterate through the RANSAC framework [33].

5. **Structure from Motion**

- The goal of this node is to comprehend the geometric relationship associated with every observation provided by the input images and conclude the rigid scene structure with 3D points, their position, their orientation and internal calibration of every camera.
- The Incremental pipeline is an upward reconstruction procedure. Firstly, this node calculates an initial two-view reconstruction and then it is iteratively stretched by introducing other views.
- Further, it combines every feature matches between image pairs into a track. Each track is assumed to characterize a point in space, noticeable from numerous cameras. At this node of the pipeline, input still comprises many outliers. During this fusion of matches, this node also removes disjointed tracks [33].
- Then, the incremental algorithm has to select the top matching initial pair of images. This selection is vital for the superiority of the final reconstruction of the object. It should indeed deliver robust matches and comprise reliable geometric data. Also, this image pair should capitalize on the number of matches and the repartition of the conforming features in every image. Nevertheless, at a similar time, the angle between the following camera positions should also be big enough to offer reliable geometric data.

- Then this step calculates the fundamental matrix between these two photos and assumes that the 1st one is the origin of the coordinate system. Now, Meshroom registers position and orientation of 2nd camera and triangulates conforming 2D features into 3D points [33].
- Further, meshroom node selects every image that has adequate relations with the features that are already reconstructed in 3D space. Based on these 2D-3D relations, this SFM node completes the resectioning of every new camera. The resectioning is a Perspective-n-Point algorithm (PnP) in a RANSAC framework to identify the position and orientation of the camera device that validates most of the features relations. On each camera, a non-linear minimization is completed to refine the pose [33].
- From these new cameras positions, some tracks become observable by two or more resected camera devices and node triangulates these positions. Then, node launches a Bundle Adjustment to improve the whole thing: extrinsic and intrinsic parameters of all camera devices as well as the position of all 3D points. Meshroom SFM node filters Bundle Adjustment results by removing every observation that has large reprojection error or inadequate angle between viewpoints [33].
- After triangulation of new points, more candidates are available for the next best scene view selection. Bundle adjustment iterates this by adding camera poses and triangulates new 2D features into 3D points and eliminates 3D points which had become invalidated until node cant find a new view [33].

6. **Prepare Dense Scene**

- "Prepare Dense Scene " node's primary function is to undistort images. This step produces undistorted EXR photos which eliminate projection conversion steps and depth calculation back and forth from the distortion function.

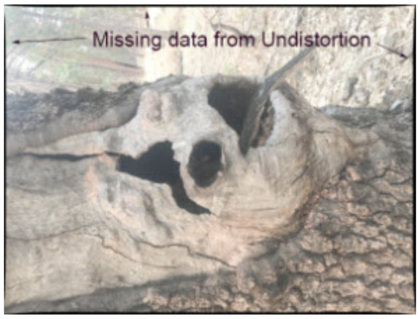


Fig. 4.2. `Prepare Dense Scene`[34]

7. **Depth Map Estimation**

- For every camera that has been determined by SfM, this node retrieves the depth value of each pixel of the image. To estimate depth value, numerous approaches like Block Matching, Semi-Global Matching (SGM) or ADCensus exist. This node focuses on the SGM method executed in AliceVision meshroom pipeline [33].
- For every image, the node selects the N best/closest camera devices nearby. This node selects front-parallel planes based on the intersection of the optical axis with the pixels of the selected neighboring cameras. This node produces a volume W, H, Z with numerous depth contenders per pixel of the image. This node estimates the similarity for each one of the pixels. The similarity is calculated by the Zero Mean Normalized Cross-Correlation (ZNCC) of a minor area in the main photo reprojected into the other camera device [33].
- This node generates a volume of resemblances. For every neighboring photo, this node collects resemblances into this volume. This generated volume consists of noisy resemblances. This node applies a filtering step along X and Y axes which collects local costs which radically decrease the score of large secluded values.

- Finally, the node selects the local minima and substitute the particular plane index with the depth value saved into a depth map volume. This depth map has banding artifacts as it is constructed on the original collection of depth values. Moreover, a refining step is applied to compute depth values with sub-pixel accuracy [33].
- All these depth maps could be calculated autonomously as parallel-processing. Later, this node applies a filtering stage to ensure uniformity between various cameras. A compromise is chosen based on both resemblance value and the number of comprehensible cameras to retain inadequately reinforced surfaces without totaling artifacts [33].
- Since this node can take a long time, there is a parameter to allow you to run groups of different cameras as different standalone commands. So if a user has 1000 cameras, then the user could depth process group of cameras with different machines on a farm. Alternatively, running in smaller groups can be useful so that if one machine crashes, the user doesn't have to rerun the whole process [33].

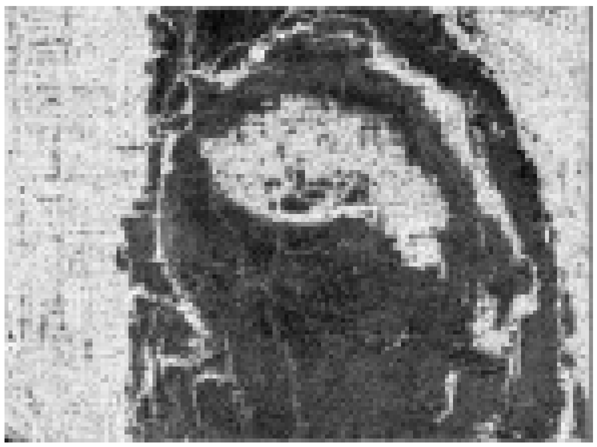


Fig. 4.3. Depth Map Estimation [34]

8. **Depth Map Filter**

    - The original depth maps are not wholly consistent; certain depth maps are claimed to see areas that are occluded by other depth maps. This Depth Map filter node isolates these areas and forces depth consistency [33].

9. **Meshing**

    - The objective of this node is to produce a dense geometric surface illustration of the image scene. Firstly, this node fuses every depth map into a global octree where compatible depth values are combined into the octree cells [33].
    - Later, this node performs a 3D Delaunay tetrahedralization. Then a complicated voting procedure is implemented to calculate weights on cells and weights on facets linking the cells [33].
    - A Graph Cut Max-Flow is applied to cut the volume optimally. This cut signifies the mined mesh surface. This node also filters terrible cells on the surface. Finally, this node applies a Laplacian filtering on the mesh to eliminate local artifacts [33].
    - At this stage, the mesh can also be simplified to diminish redundant vertices.

10. **Mesh filtering**

    - In this node, post-processing is performed on the mesh from the meshing node with some refinements. This mesh filtering node performs actions such as smoothing of the mesh, removal of large triangles, keeping the largest mesh but removing all other small mesh [33].
    - Some of these operation nodes are not necessarily needed for particular applications so you can custom those parameters as required.

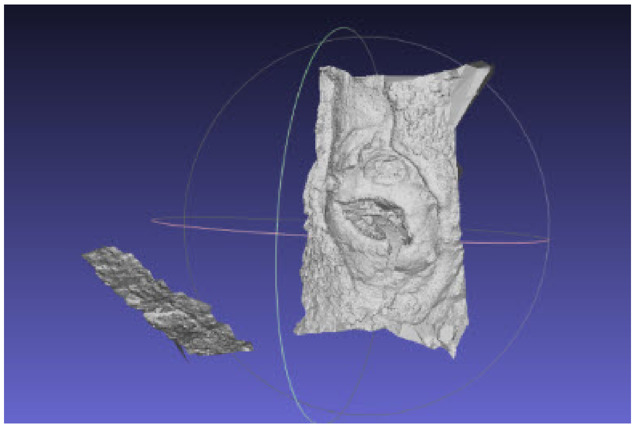


Fig. 4.4. `Meshing Node` [34]

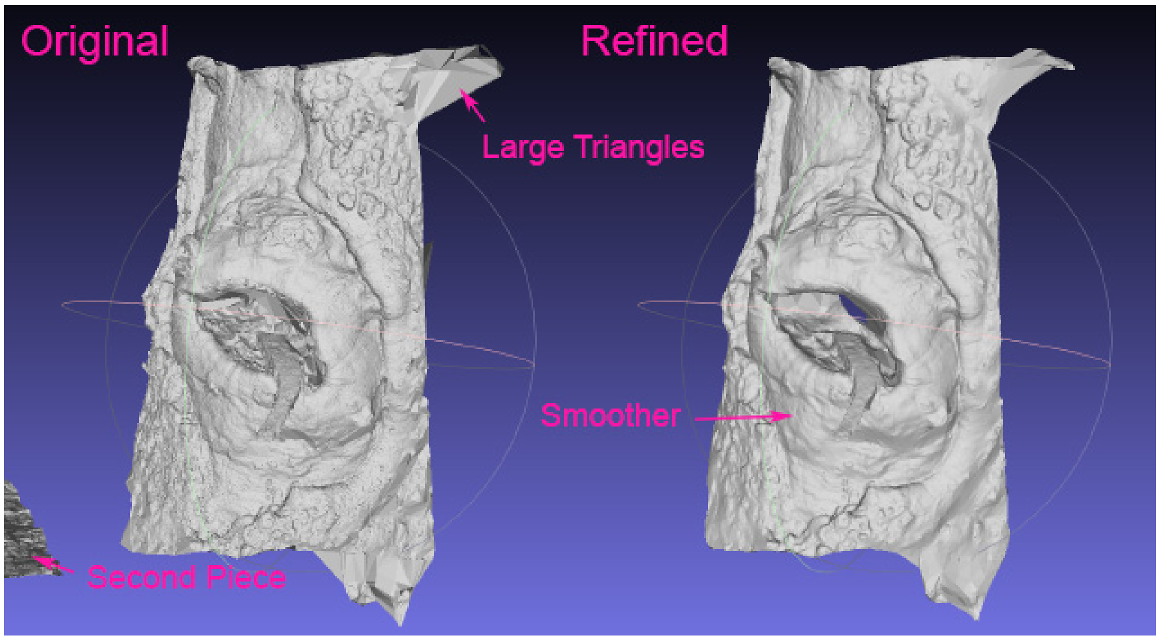


Fig. 4.5. `Mesh Filtering Node` [34]

5. INTEGRATING IN SOLIDWORKS API

In this research, SOLIDWORKS add-in has been created using C# (C Sharp) programming language. The following section describes the complete procedure of creating add-in taskpane in SOLIDWORKS.

SOLIDWORKS

SolidWorks is a solid modeling CAD (Computer Aided Design) and CAE (Computer Aided Engineering) computer program that works on the Microsoft Windows operating system. Dassault Systmes publishes SolidWorks. Along with eDrawings, collaboration tool, and DraftSight (2D CAD product), SolidWorks sells numerous versions of the SolidWorks CAD software. SOLIDWORKSs user base spread out from individuals to big companies and covers a massive cross-section of manufacturing market sectors. The direct competitors to SolidWorks are PTC Creo Elements/Pro, Solid Edge, and Autodesk Inventor and other softwares. [35].

C# programming language

C# is pronounced as C sharp, which is a general-purpose, multi-paradigm programming language covering strong typing, lexically scoped, imperative, declarative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines [36].

It was industrialized by Microsoft within its .NET initiative and later accepted as a standard by ISO (ISO/IEC 23270:2018) and Ecma (ECMA-334). C# is one of the programming languages intended for the Common Language Infrastructure [36].

5.1 SOLIDWORKS API

SOLIDWORKS API help reference guide describes the SOLIDWORKS Application Programming Interface (API), which user can utilize to mechanize and modify the SOLIDWORKS software as per requirements [37].

The API comprises more than a thousand functions that user could access from Visual Basic for Applications (VBA), Visual C#, VB.NET, Visual C++ 6.0, and Visual CLI/C++. These functions offer direct entree to SOLIDWORKS functionality such as creating a line, inserting an existing part into a part document, or confirming the parameters of a surface/bodies [37].

The user can locate the SOLIDWORKS primary interop assemblies (SolidWorks.Interop.*.dll) and type libraries (*.tlb) in the install_dir folder. Locate the SOLIDWORKS redistributable interop assemblies in: install_directory> api>redist.

These SOLIDWORKS interops were programmed using Microsoft .NET Framework Version 4.5.1. The user can use these in SOLIDWORKS 2018 .NET macros and add-ins by VB.NET and C# languages.

Each compatible interface is available in the SOLIDWORKS API along with their associated properties and methods. SOLIDWORKS support team recommends that the user should use the most recent version of the API from SOLIDWORKS. To use SOLIDWORKS API, the user must be familiar with VB.NET, VBA, Visual C++ 6.0, Visual C#, or Visual CLI/C++ [37].

5.2 SOLIDWORKS Add-in Procedure

SOLIDWORKS Interop files

So the very first thing the user needs to do is go to C Drive Program Files or where the user has installed SolidWorks folder, and then inside SolidWorks folder the user needs to browse for SolidWorks Interop DLL files.

Once these SolidWorks Interop DLL files are located, the user has to copy all these files and paste it anywhere in a folder called references by users choice. Once these SolidWorks Interop DLL files are located, the user has to copy all these files and paste it anywhere in a folder called references by users choice.

.dll files

Dynamic Link Library(.dll) file contains a library of functions andother information that can be accessed by a Windows program. Links to the essential .dll files are generated whenever a program is being launched. Two types of links can be established static and dynamic. If a static link is formed, the .dll files are in usage till the time the program is running; while in case of dynamic links, these .dll files are used when it's necessary. This dynamic condition helps the program to manage the memory of the computer and hard drive space efficiently.

These .dll files could also be utilized by more than one program simultaneously. Some of the DLLs provided with the Microsoft Windows operating system whereas others are linked during new programs installation. Typically the user doesn't need to open a .dll file directly as the program that utilizes; it loads .dll files automatically if desired [38].

Project Setup

For this research, Microsoft Visual Studio 2015 professional version has been used. A new project has been created inside the visual studio with a visual c# class library. There are quite a few class libraries like .NET CORE, portable other than the plain class library. Out of these options, visual C# plain class library has been selected and project has been stored at a particular location on computer's hard drive.

After the creation of the project in a solution, the properties of this project should be edited to change the assembly name and information.

Assembly name could be the same as out default namespace. In the assembly information, assembly COM is made visible so that SolidWorks can use the DLL later because if it runs on comm, so it needs to be able to access everything in this file.

Adding References

After the assembly properties are being set up, references to this SOLIDWORKS API assembly added from the folder with SOLIDWORKS interop dll files. For this project, the following references have been added: sldworks.dll, swcommands.dll, swconst.dll, swdocumentmgr.dll and swpublished.dll.

swconstant stores all specific enumerator values while swcommands have got other parts and swdocumentmgr is required for working on files when they're not open. Swpublished allows access to the SOLIDWORKS user custom interfaces [37].

Creating Interface

As discussed in the earlier reference section, swpublished is responsible for allowing access to custom interface in SOLIDWORKS. In this step, created public class need to connect via an interface called ISwAddin which uses swpublished namespace from references. After creating this add-in interface, the further step is to implement two functions which are to connect and disconnect from SOLIDWORKS.

In this part, add-in registers the registry entries to regedit and responds to SolidWorks telling it to connect and disconnect and it creates the task pane window. A region is created as private members which contain some required variables. A private integer for the SolidWorks cookie to the current instance of SOLIDWORKS.

A taskpaneview member which is inside sldworks namespace is required for the add-in's user interface. A blank user control interfaces is created using project options.

This design created by programmers is injected into SOLIDWORKS as a user interface as actual control. In C#, its easy to drag and drop buttons from toolbox for user control. The user interface size can be edited as per requirement and buttons can be placed as desired.

See figure 5.1 for the primary user interface of the photogrammetry pipeline in SOLIDWORKS. Another private variable is created for instances to control user interface inside SOLIDWORKS add-in view. For the current instance of the SOLIDWORKS application a private variable is created.

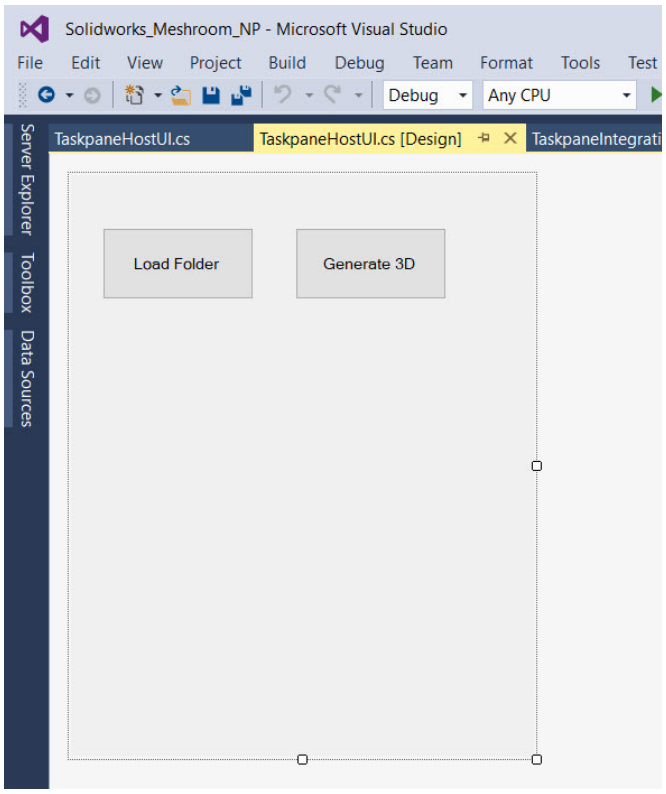

Solidworks_Meshroom_NP - Microsoft Visual Studio
File Edit View Project Build Debug Team Format Tools Test
Debug
Any CPU
TaskpaneHostUI.cs
TaskpaneHostUI.cs [Design]
TaskpaneIntegrati
Server Explorer
Toolbox
Data Sources
Load Folder
Generate 3D

Fig. 5.1. `Add-in User Interface Design`

Registering Unique ID

Created Add-in design is going to appear on the right-hand side in SOLIDWORKS and it needs a unique ID so that it can be differentiated between created add-in and every other add-in. Thus for registering a unique ID, a public constant string swtaskpane_progID is created. Now, this defined member can be added to designed user interface's code using runtime interop services namespace.

For current project, SW-Meshroom is the unique ID to the add-in used for the registration in COM. then while constructing the add-in, a user control can be effectively injected by merely passing this PROG ID.

Implementing User Interface

Next step is to implement the ISwAddin interface. So the functions like ConnectToSW and DisconnectFromSW are needed for SolidWorks add-in at a minimum to respond when the program is told to connect and disconnect.

ConnectToSW is called when Solidworks has loaded our add-in and wants us to do our connection logic. During this SOLIDWORKS instance and SOLIDWORKS cookie ID are stored. Now one other thing is needed that is to set callbacks; these come into play later on but this is just a standard thing needed to do. Now, at the location of the add-in in SOLIDWORKS, the user interface can be injected using a function block called LoadUI. In this LoadUI block, user interface can be injceted with image of the logo on the add-in.

DisconnectFromSW is called when Solidworks is about to unload our add-in and wants us to do our disconnection logic. Disconnect works similarly to ConnectSW there's very little to do here, this is to unload our UI in essence. DisconnectFromSW cleans up created user interface during closing. This cleanup includes the deleting temporary memory add-in view and COM reference.

COM Registration

Now to get that add-in to load into SolidWorks, it's needed to register add-in via COM. This add-in needs the registry windows regedit actually to recognize this class and make it a COM object. The way that COM works as it reads the registry for class

IDs and finds out a particular class and then loads it in. To do this, its required to create some COM register functions and unregister functions. To do this, its required to create some COM register functions and unregister functions.

This is flagged by COM register function so that while calling RegAsm, it knows where to find this function and it calls this function. So this registers .dll as COM and this add-in needs to integrate with SOLIDWORKS registry. In this step, SOLIDWORKS finds add-in .dll file with defined path and temporarily installs it (registers it) while SOLIDWORKS is opening. When SOLIDWORKS is closing, then this step deletes this data while unregistration.

5.3 Algorithm Behind Buttons

Now at this stage, SOLIDWORKS add-in has been created with a primary design with two buttons. One of these buttons can be used for loading image and the other one can be used for object reconstruction from those images using Meshroom pipeline.

In this injected user interface, image folder needs to be loaded for photogrammetric operation. For photogrammetry methodology, only a few types of image file formats are allowed. Following image file formats are acceptable into pipeline: JPEG(*.jpg,*.jpeg), PNG(*.png) , TIFF(*.tif,*.tiff), BITMAP(*.bmp). Thus while using this user interface one has to make sure that the folder which is being selected for photogrammetry must contain files with formats as mentioned earlier.

In user interface design code, an instance has been created using FolderBrowserDialog class which belongs to System.Windows.Forms namespace. Through this like every other folder browser dialog box, the user has to select the path to image path. This folder path is later used for photogrammetry's first step in pipeline CameraInit as input.

Meshroom photogrammetry pipeline is programmed in the Python programming language. This photogrammetry pipeline has several stages for object reconstruction as discussed in the earlier chapter about Meshroom. So with the available source code

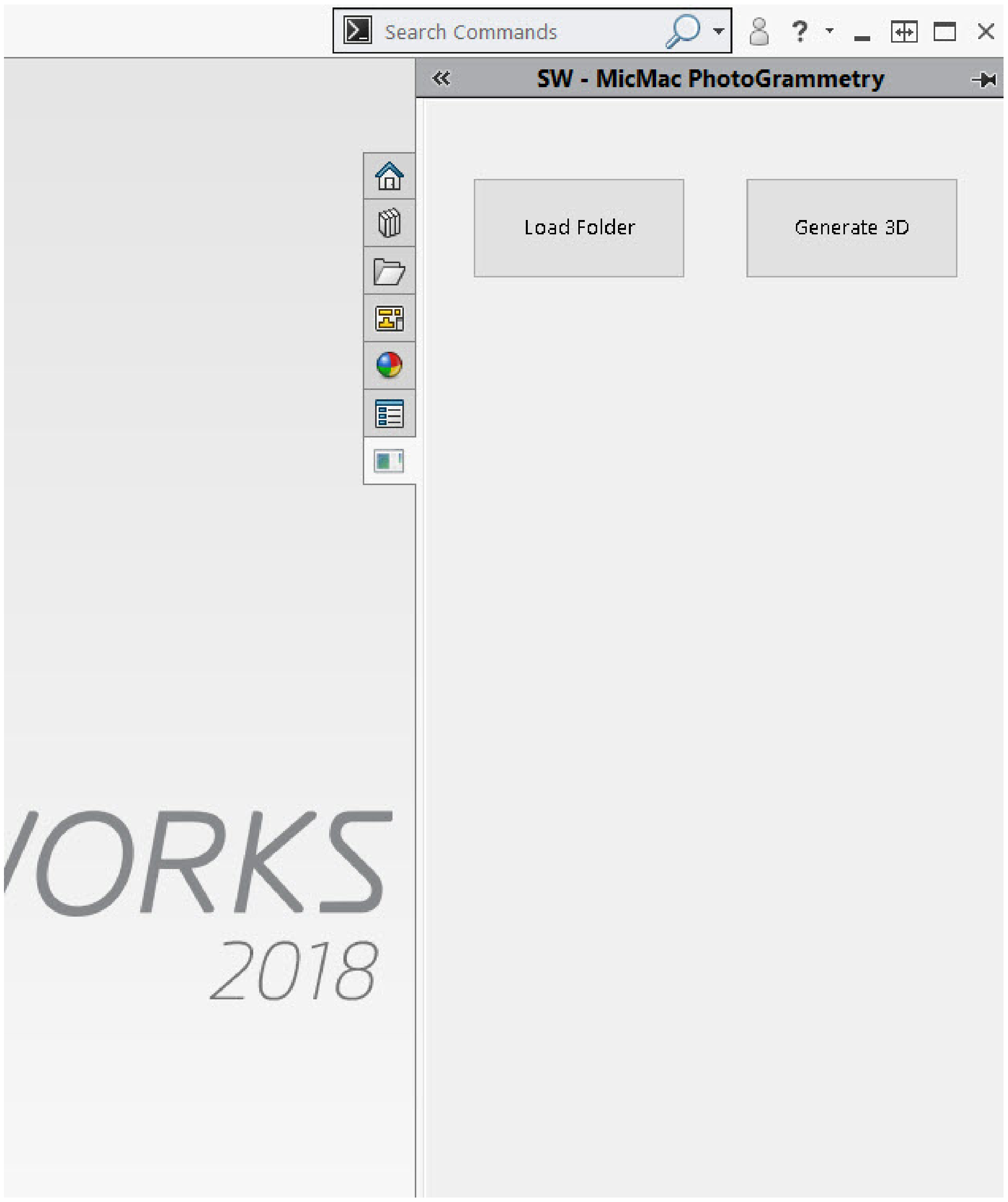


Fig. 5.2. `Add-in View In SOLIDWORKS UI`

in python, there is a need for creating separate executable files for each stage which can be built using Python compiler and command prompt. However, these executable files need arguments and parameters to execute without any problem. The following section describes arguments for each stage of the photogrammetry pipeline which is being used behind Generate 3D button.

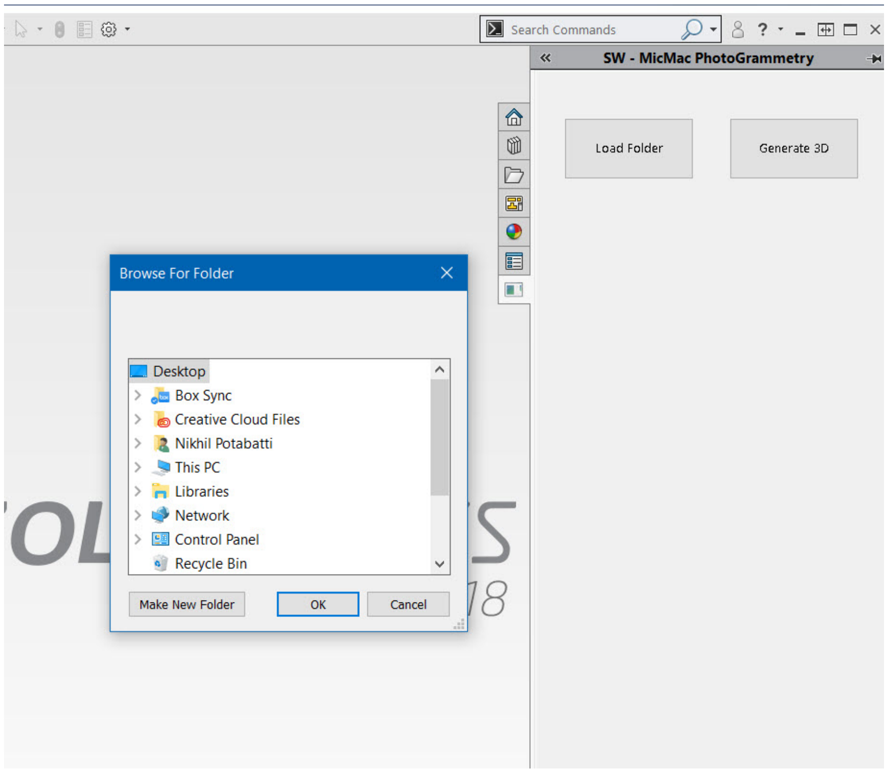


Fig. 5.3. `Folder Browser in SOLIDWORKS`

For CameraInit stage, the path for Images' folder as input. Based on this, the camera sensor database is updated for sensor information which is later used to create SFM file. This SFM is an output from this stage and it is stored in a separate folder.

The output from CameraInit stage is used for the feature extraction process in the pipeline. This node saves several files in a separate file for Feature Extraction. These files are computed using SIFT describer type. There are several other describer types are available other than SIFT like SIFT Float, AKAZE, CCTAG so on.

The SFM file created in Camerainit stage is used as input in the Image matching node. Along with this, the output folder of the feature extraction node and tree file are passed as arguments to Image Matching node. After computation, the output file is stored into a separate image matching folder. Argument for minimum number of images for this image matching node could be altered if required.

In the Feature matching node, Brute force is used as the default photometric matching method. In this node, inputs are the CameraInit generated output SFM file and image pairs which are calculated using Image Matching node. Also, the argument for feature extraction folder is passed to this node. Again, output results of Feature matching are saved in a separate folder. In subsequent nodes, these output files are carried forward and the object is reconstructed in .obj file format. All nodes' outputs are stored in separate folders to streamline all nodes. Once the object is reconstructed, the message box appears which reads "the 3D model created!"

So this is the code which can be inserted behind Generate 3D button in SOLDIWORKS Add-in. After reconstruction of the object from its images, this mesh file can be imported into SOLIDWORKS from its folder for visualization and further processing. Later on, this mesh file can be post-processed using Mesh prep wizard inside Solidworks. See code in Appendix.

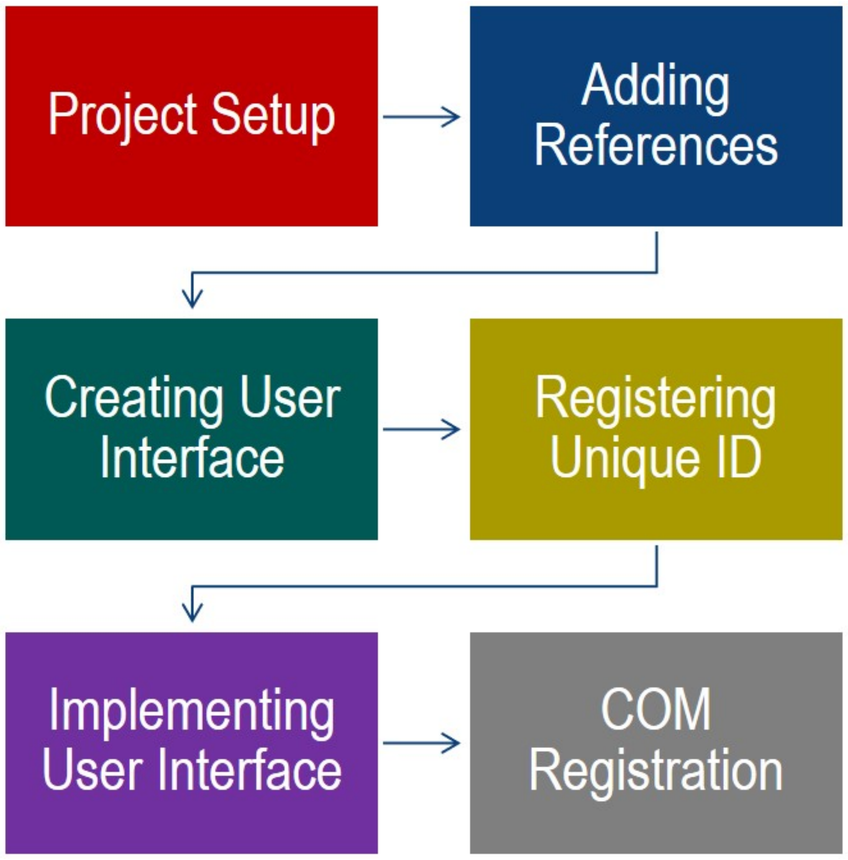


Fig. 5.4. `SOLIDWORKS Add-in Setup Procedure Flowchart`

6. CASE STUDIES AND VALIDATION

In this chapter, the case study about using this integrated Photogrammetry pipeline and reverse engineering has been discussed. For the following case study, a casting part for manufacturing flywheels is used. See the following Fig.6.1

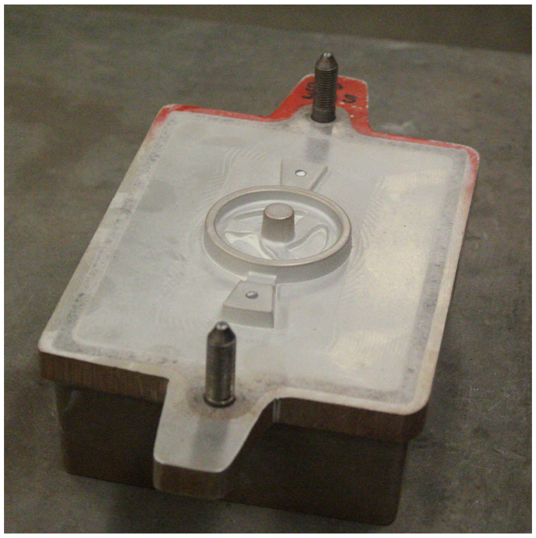


Fig. 6.1. Flywheel Casting Part

6.1 Case Study I

6.1.1 Photoshoot

To capture multiple images for case study I and II, a Canon EOS 700D camera is mounted on the tripod. Canon EOS 700D camera's specifications are mentioned in the following Table.6.1. The ambiance was well lit with diffused ceiling light so that noises can be removed from the photogrammetric process. For this photoshoot, Canon's 1855 wide angle lens is used.

Table 6.1.
Canon EOS 700D Camera Specifications [40]

System information	**Specifications**
Type	DSLR
Indicative Price	USD 750
Resolution	5208 x 3476
Type of Sensor	CMOS
Sensor photo detectors	18.1 Megapixels (MP)
Sensor size	14.9 x 22.3 (mm)
Focal length multiplier	1.6
Shutter speed range	1/4000 - 30.0
Frame rate	5.0 fps

The camera was set to capture RAW images; thus it could not eliminate or compress details in the scene. These Raw files generate image of the size in between 20-30 MB. However, here is the catch, for photogrammetry pipeline only JPEG images are allowed. If images are captured in JPEG format, resulting images are of the size around 10MB which loses to a converted JPEG file (approx 15 MB). Hence , these images are later converted to JPEGs using Adobe Photoshop by exporting settings.

With this camera, a total of 42 images are acquired from different angles and variable heights using a tripod as a stabilizer. The entire photoshoot took 20 min to get as many angles as possible for accuracy. These 42 images include 14 top level, 14 mid-level and 14 base level photographs. For this particular case study, the flashlight wasn't utilized during the photoshoot. At last, two dimensions, width and length were measured using digital vernier caliper. These measurements will come into play during scaling of the object reconstruction results. As discussed in earlier chapters, photogrammetry results do not provide original scaled models. Thus many researchers have used coded targets or markers for scale calibration.

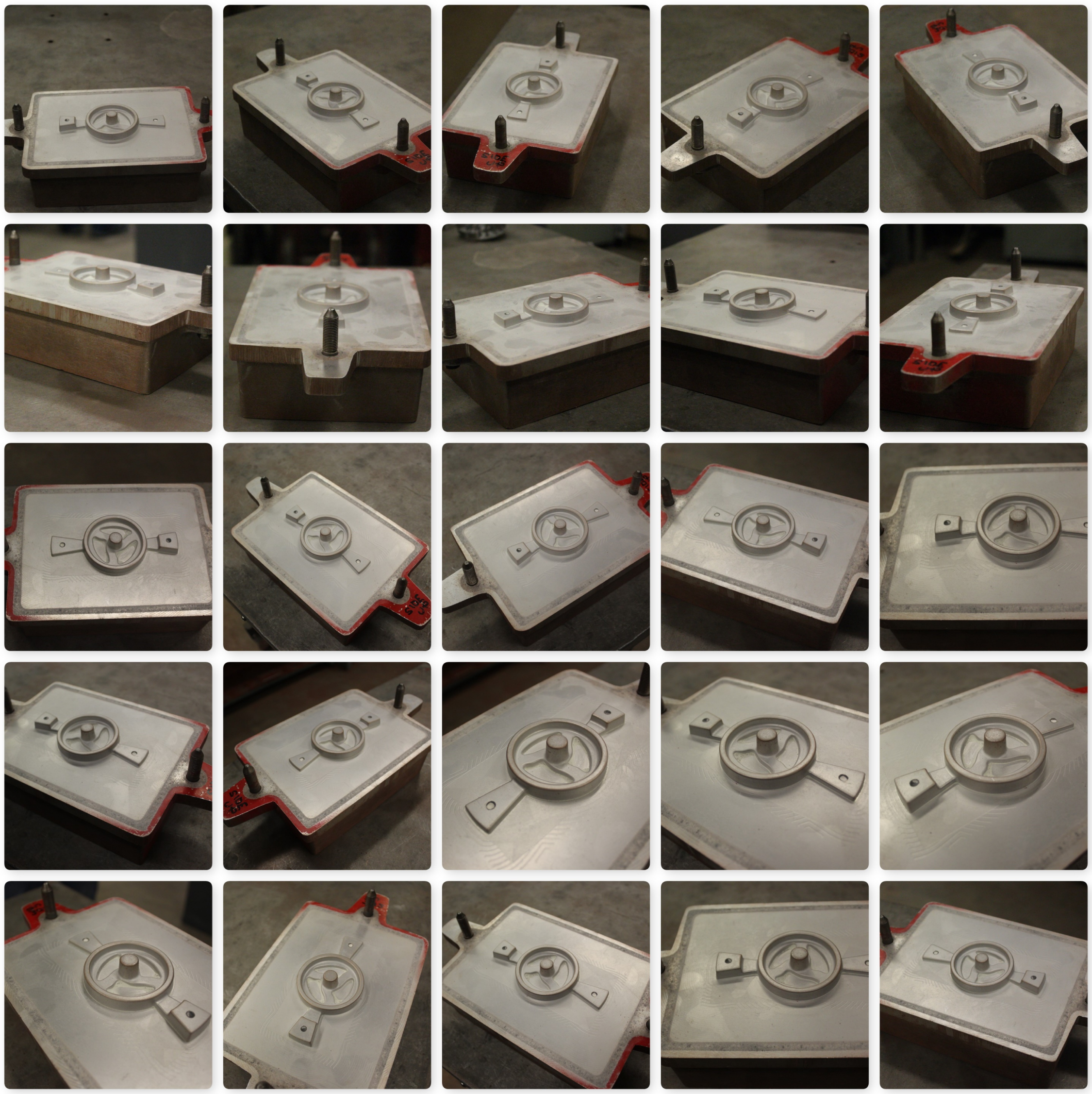


Fig. 6.2. `Image acquisition for Flywheel`

6.1.2 Photogrammetry Add-in in SOLIDWORKS

Once these acquired images converted into JPEG format, they are stored in a separate folder. This folder is then selected using the ”load folder” option in the add-in's user interface from SOLIDWORKS. After that, the photogrammetry pipeline is executed by clicking on button Generate 3D. For this case study, in particular, the time taken for complete object reconstruction is about 2 hours and 4 minutes. When add-in showed a message ”3D model Created”, the mesh file is opened as Scanto3D mesh file in .obj format. See the following figure 6.3.

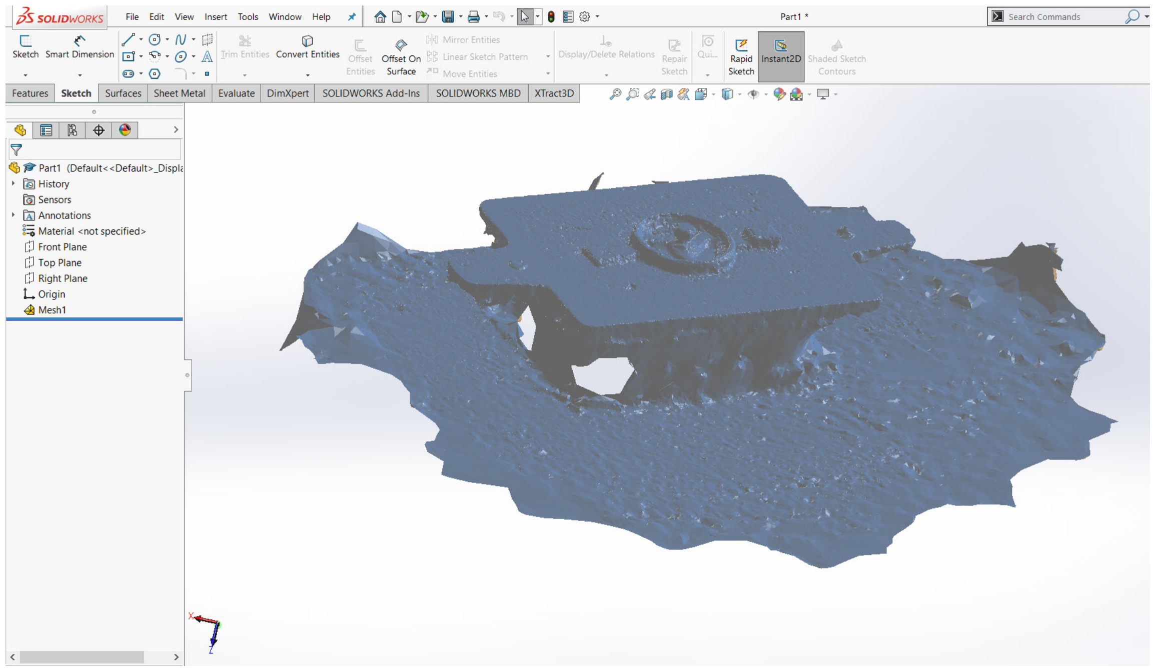


Fig. 6.3. `Flywheel Mesh file after reconstruction`

As shown in the figure, this reconstructed mesh file needs to be post-processed. For this particular operation, the mesh prep wizard is used from SOLIDWORKS. Using Mesh prep wizard, holes can be knitted, unnecessary surface mesh patches can be removed and surface mesh file can be smoothed. See the following figure 6.4 for mesh results after post-processing.

6.1.3 Reverse Engineering using Mesh File

For reverse engineering purpose, XTRACT3D add-in from ”polyga” company used which works natively inside of the SOLIDWORKS software. Using this add-in, the

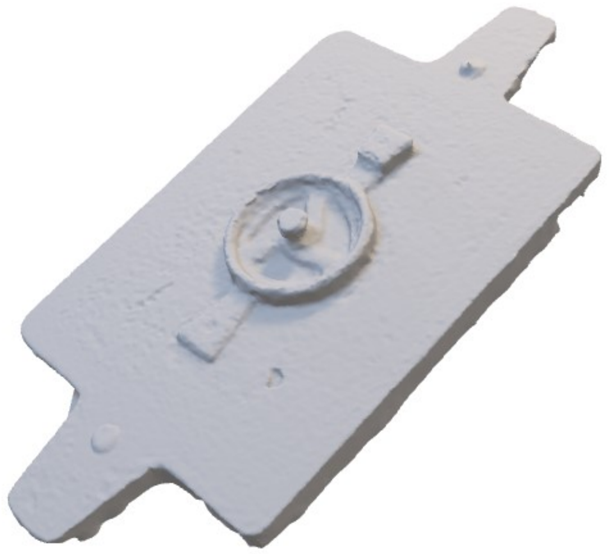


Fig. 6.4. Flywheel Photogrammetry Mesh Results after cleanup

user can import 3D scanned data and utilize this data as a reference for building 3D CAD models. Using XTRACT3D tool inside SOLIDWORKS, post-processed mesh is first oriented using points and moving the mesh as required.

Further, this photogrammetry mesh is scaled by using measurement data calculated using Digital caliper. When the mesh is positioned with the new coordinate system, then makes it easier to reverse engineer in the later stages.

After positioning and scaling, a scanned mesh of the same object is brought into SOLIDWORKS. The same Flywheel is scanned using CREAFORM GoScan 20TM scanner and scanned mesh is post-processed for cleanup. See table 6.2 for CREAFORM Scanner's Specification.

While using the 3D scanner, little bit talcum powder is spreded over flywheel's surface to hide the shiny reflections from the surface. Moreover, after removing the unnecessary area from the Scan mesh, it is imported into SOLIDWORKS for mesh overlay. Refer figure 6.5 for the scan mesh file after cleanup.

Table 6.2.
CREAFORM GoScan 20 scanner Specifications [41]

System information	**Specifications**
Accuracy	Up to 0.100 mm (0.004 in.)
Volumetric accuracy	0.300 mm/m (0.0036 in./ft)
Resolution	0.100 mm (0.004 in.)
Light source	White light (LED)
Measurement rate	550,000 measurements/s
Scanning area	143 x 108 mm (5.6 in x 4.3 in)
Stand-off distance	380 mm (15 in.)
Depth of field	100 mm (4 in.)
Output formats	.obj, .ply, .stl, .txt, .wrl, .x3d
Price	USD 17500

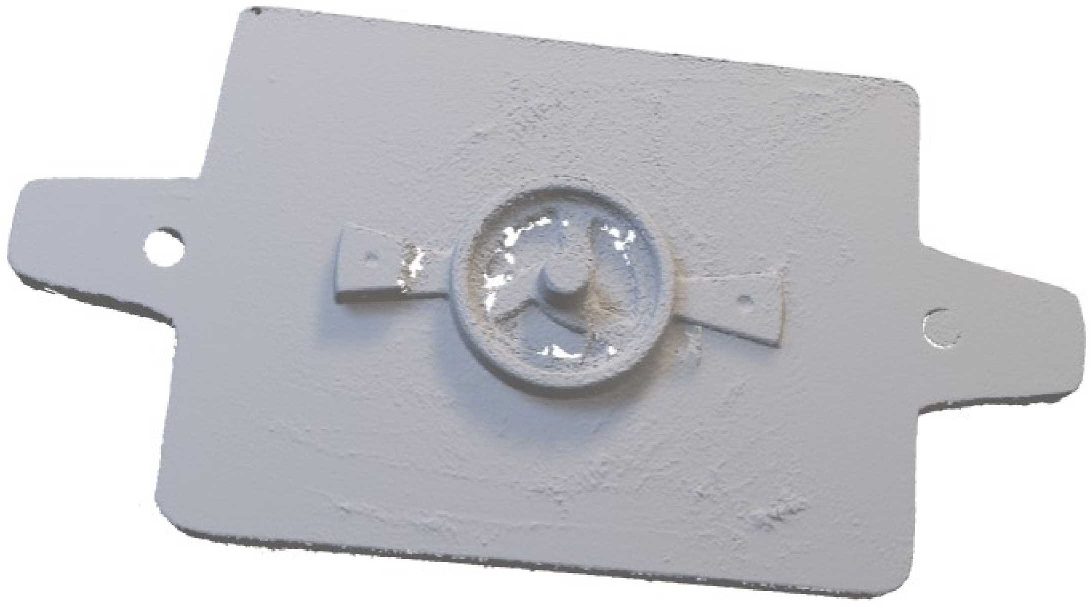


Fig. 6.5. `Scan Mesh Results after cleanup from CREAFORM scanner`

XTRACT3D provides scaling and orienting functionalities with the add-in; this can be used to overlay post-processed Scanned mesh on post-processed photogrammetry mesh. Different color codes can identify both meshes. Scan mesh file is denoted in pink while photogrammetry mesh file is denoted in ivory color as shown in the following figure. 6.6. For next step, scan mesh can be hidden from the model tree.

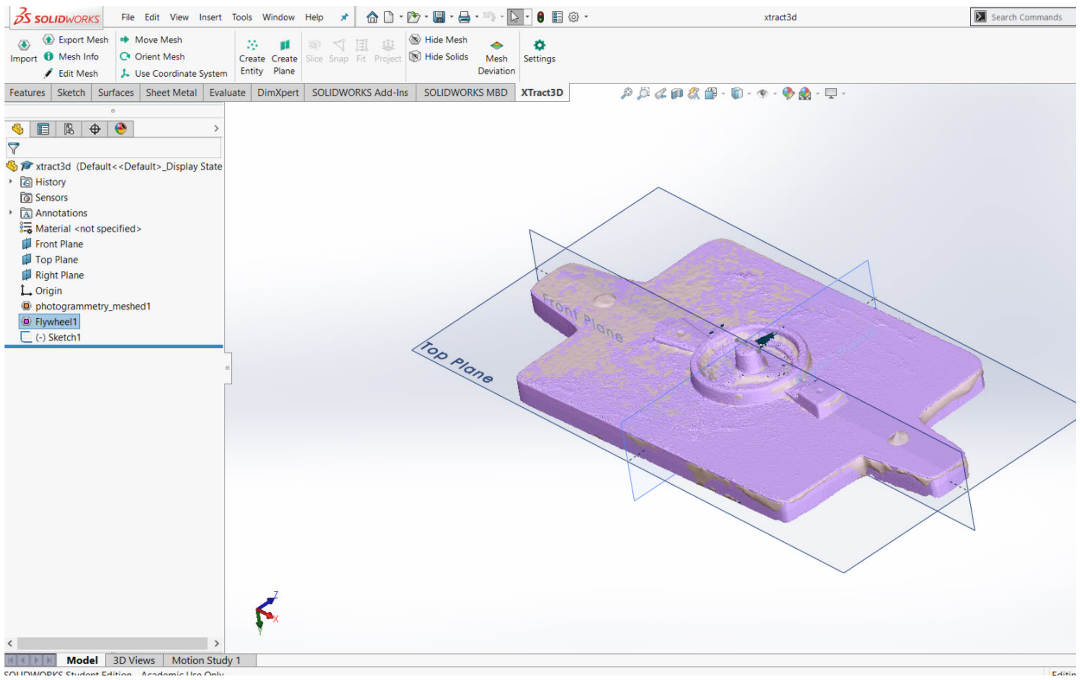


Fig. 6.6. Mesh overlay in SOLIDWORKS

Using slices about various planes in the mesh, XTRACT3D helps the user to model the part by the red contour in sliced plane views. The entire part could be intuitively modeled using this mesh file in SOLIDWORKS XTRACT add-in. See following figures 6.7 and 6.8 of reverse engineering from mesh file in SOLIDWORKS.

This reverse modeled part can be compared with mesh files by using mesh deviation feature in SOLIDWORKS. In this case study, newly modeled part is compared with scan mesh file as well as photogrammetry mesh file. See the following figures for mesh deviation analysis results. These results and work-flow is discussed in the next chapter.

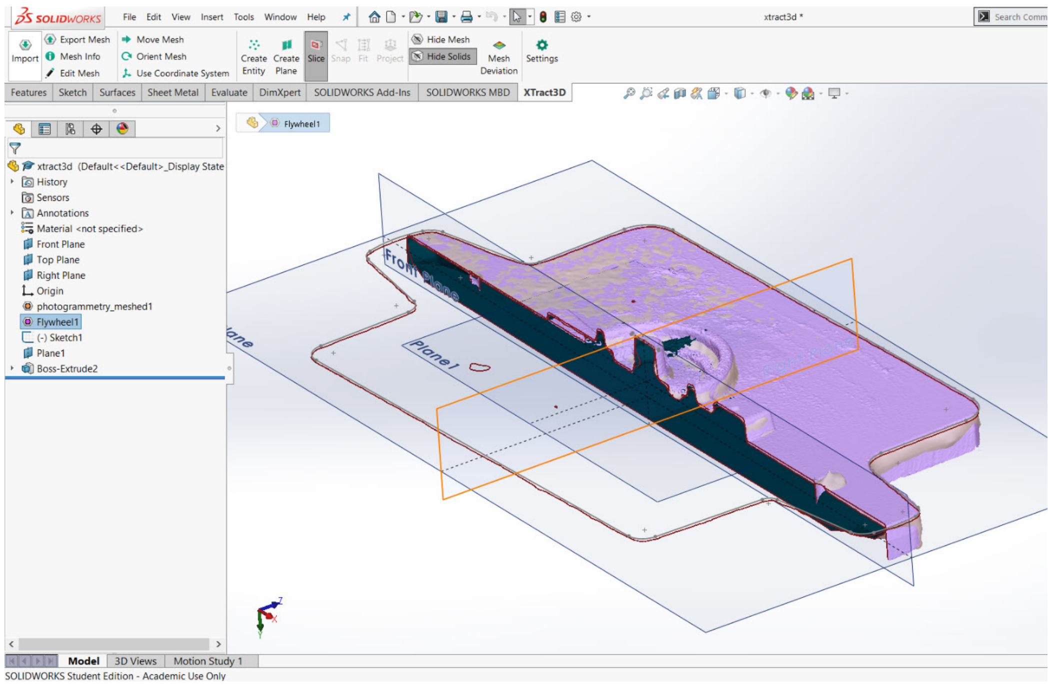


Fig. 6.7. Use of mesh slicing

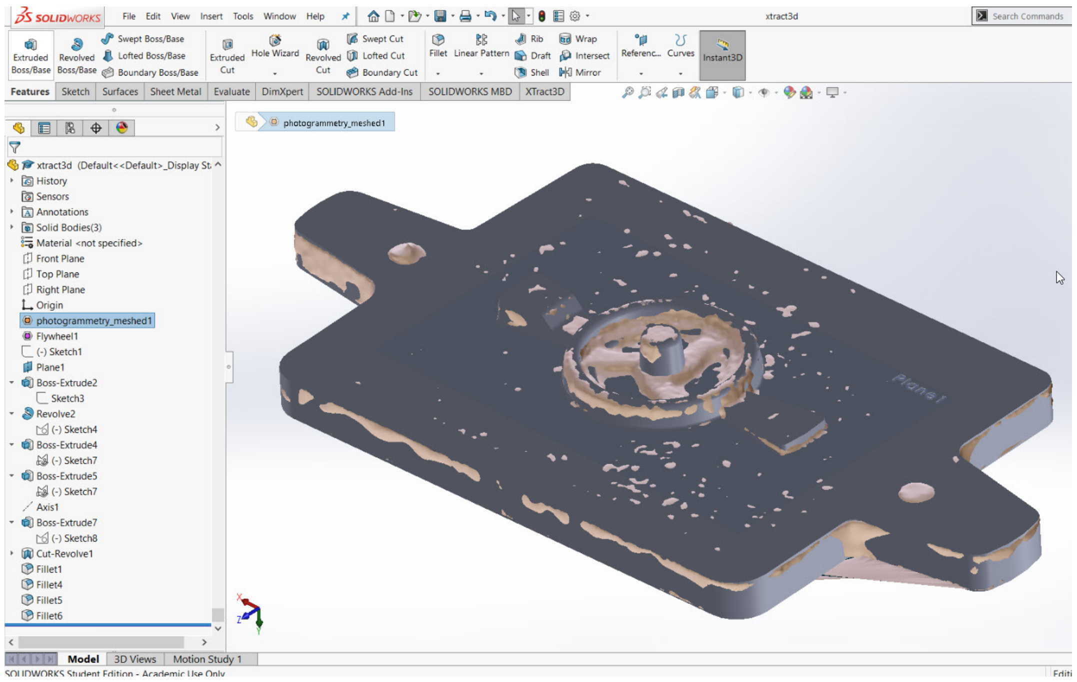


Fig. 6.8. Revere Modeling from photogrammetry mesh data

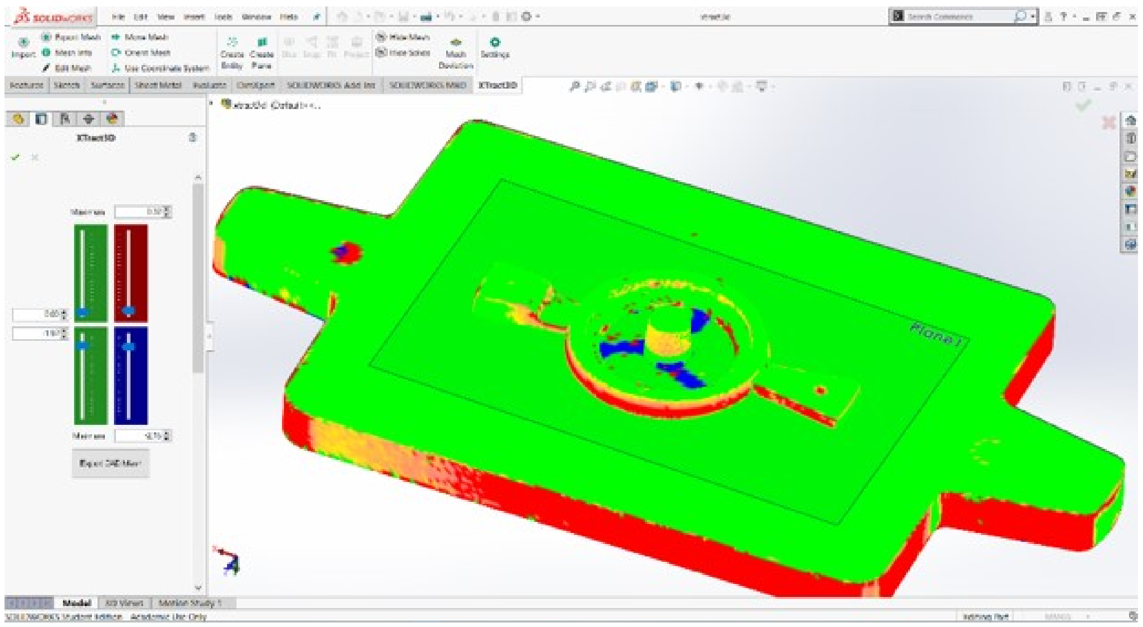


Fig. 6.9. Mesh Deviation Analysis between Scan Mesh and Modeled Part

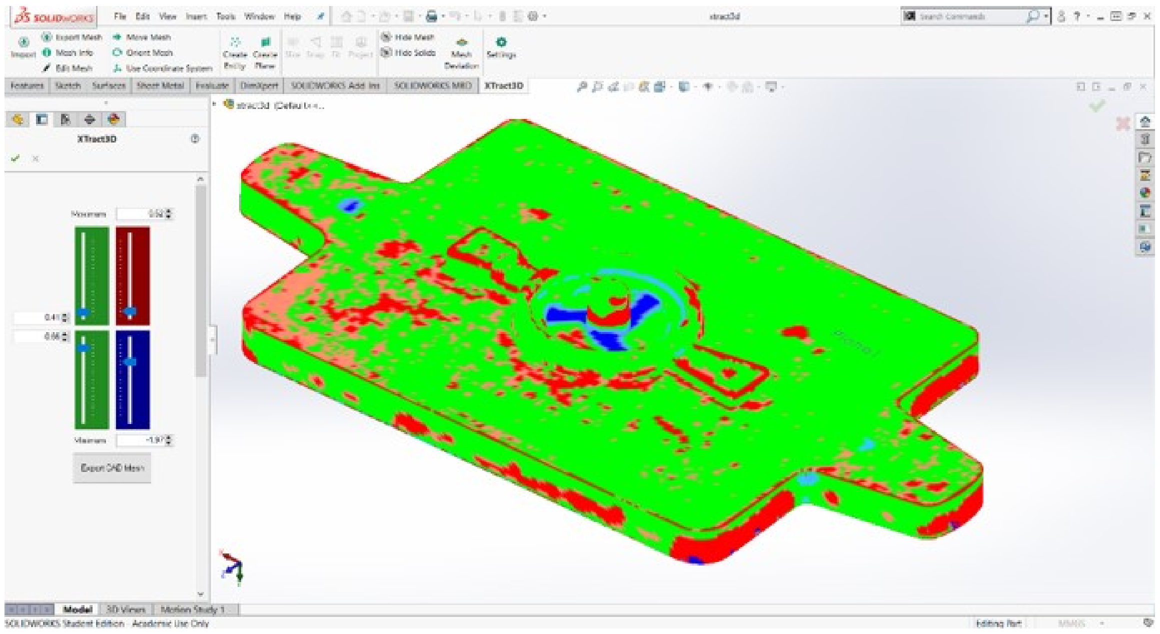


Fig. 6.10. Mesh Deviation Analysis between Photogrammetry Mesh and Modeled Part

6.2 Case Study II

For second case study, a gearbox casing is used which is shown in the following Fig. 6.11. For this case study same Image Capturing setup was used as Case Study I.

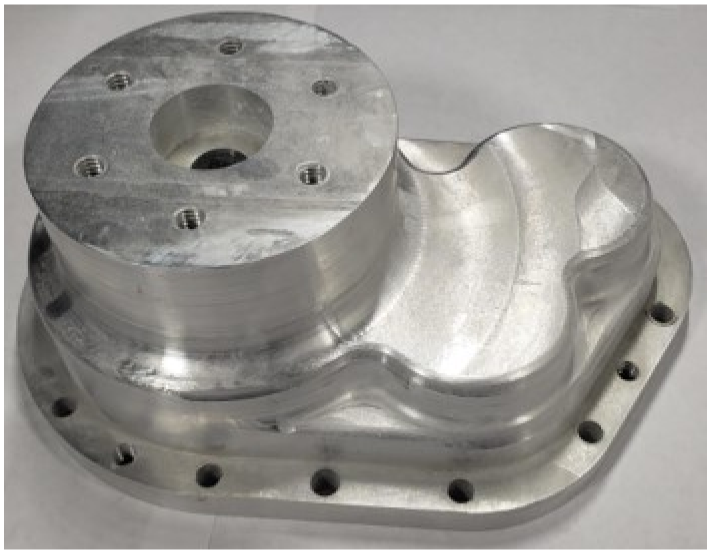


Fig. 6.11. Machined Gearbox Casing

The camera was set to capture RAW images; thus it could not eliminate or compress details in the scene. With the same camera, a total of 70 images are acquired from different angles and variable heights using a tripod as a stabilizer. The entire photo-shoot took 25 min. to get as many angles as possible for accuracy. These 70 images include 25 top-level, 25 mid-level and 20 base level photographs. For this particular case study, the flashlight wasnt utilized during the photo shoot. At last, two dimensions, width and length were measured using digital vernier caliper. These measurements will come into play during scaling of the object reconstruction results.

Once these acquired images converted into JPEG format, they are stored in a separate folder. For this case study II, in particular, the time taken for complete object reconstruction is about 4 hours and 10 minutes.

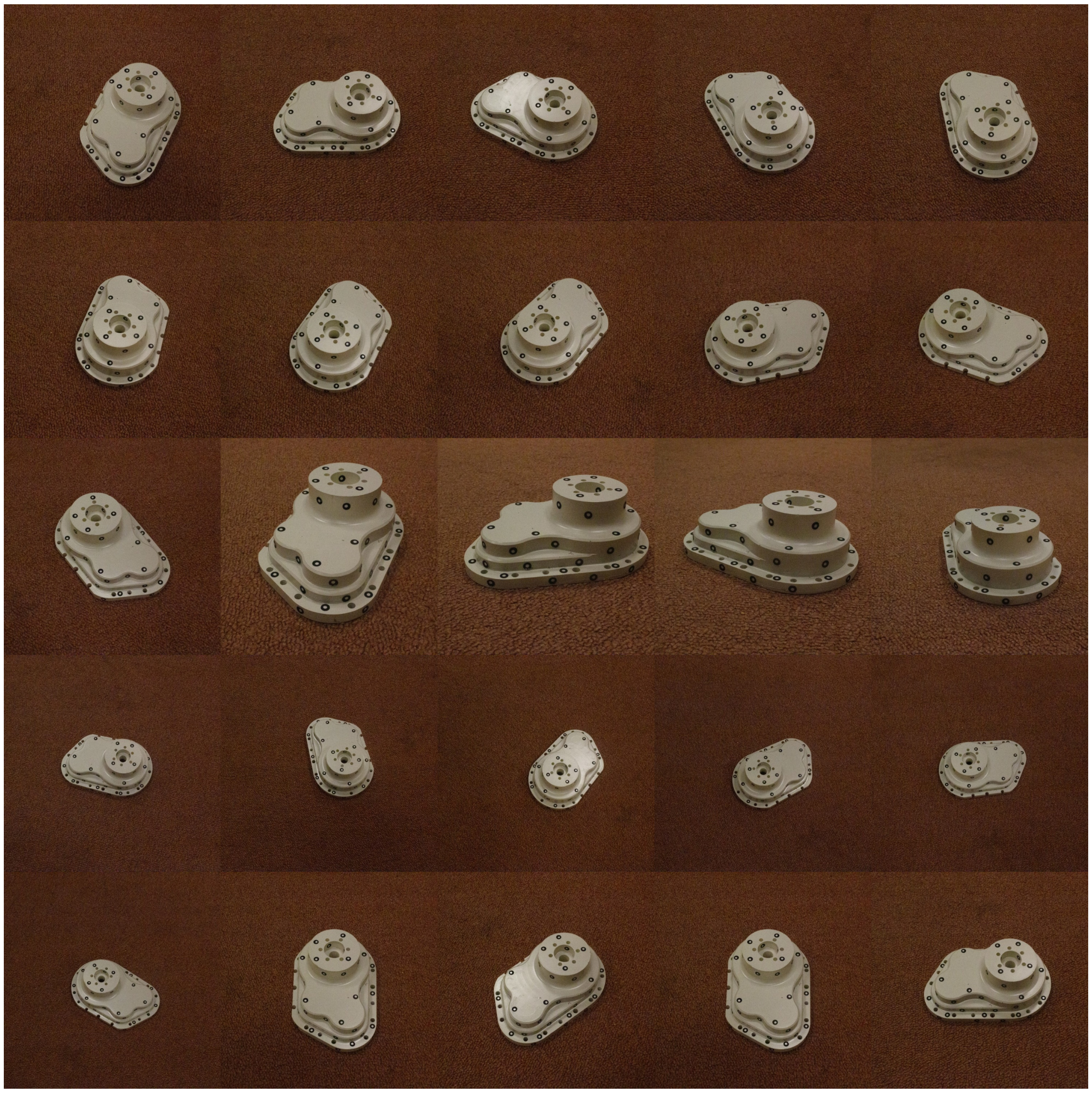


Fig. 6.12. Image Acquisition of Gearbox Casing

To reverse engineer CAD model, same XTRACT3D add-in used. Using this add-in, I have imported 3D scanned data and photogrammetry data as reference for building 3DCAD models. Using XTRACT3D tool inside SOLIDWORKS, post-processed mesh is first oriented using points and moving the mesh as required. Further, this photogrammetry mesh is scaled by using measurement data calculated using Digital caliper. After positioning and scaling, a scanned mesh of the same object is brought into SOLIDWORKS. The same gearbox casing is scanned using CREAFORM GoScan. Moreover, after removing the unnecessary area from the mesh, it is imported into SOLIDWORKS for meshoverlay. Refer figure 6.14 for the scan mesh file after cleanup.

This reverse modeled part compared with mesh files by using mesh deviation feature in SOLIDWORKS. In this case study, newly modeled part using is compared with photogrammetry mesh file. See the following Fig. 6.15 for deviation analysis results. The maximum mesh deviation of photogrammetry-reverse model against scan mesh is 0.09654 inches (2.452116 mm) and minimum mesh deviation for the same is 0.007783 inches (0.1976882mm).

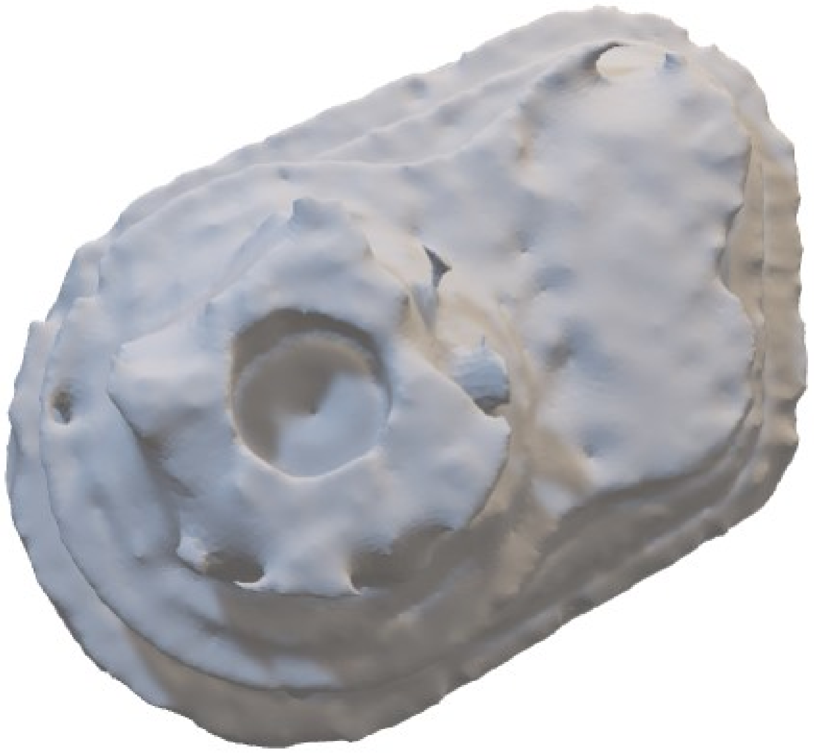


Fig. 6.13. Cleaned up Photogrammetry Mesh of Gearbox Casing

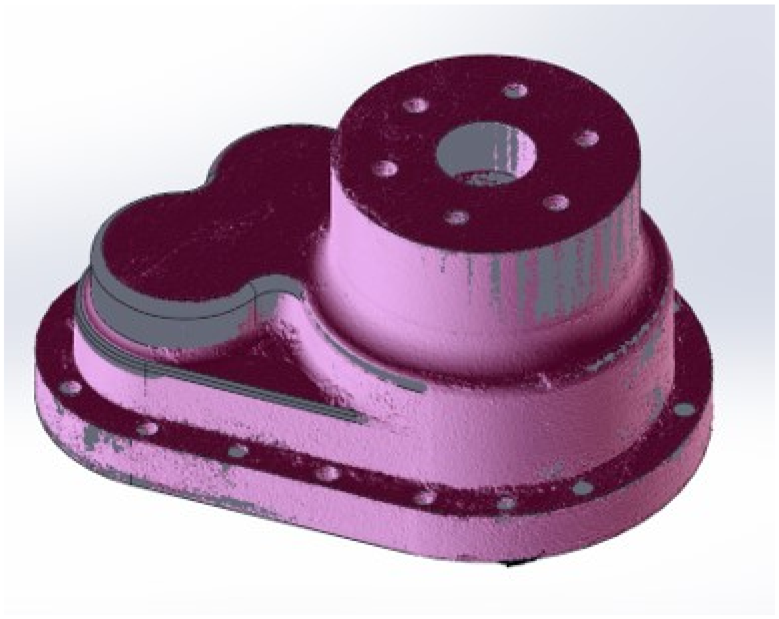


Fig. 6.14. Gearbox Mesh overlay in SOLIDWORKS

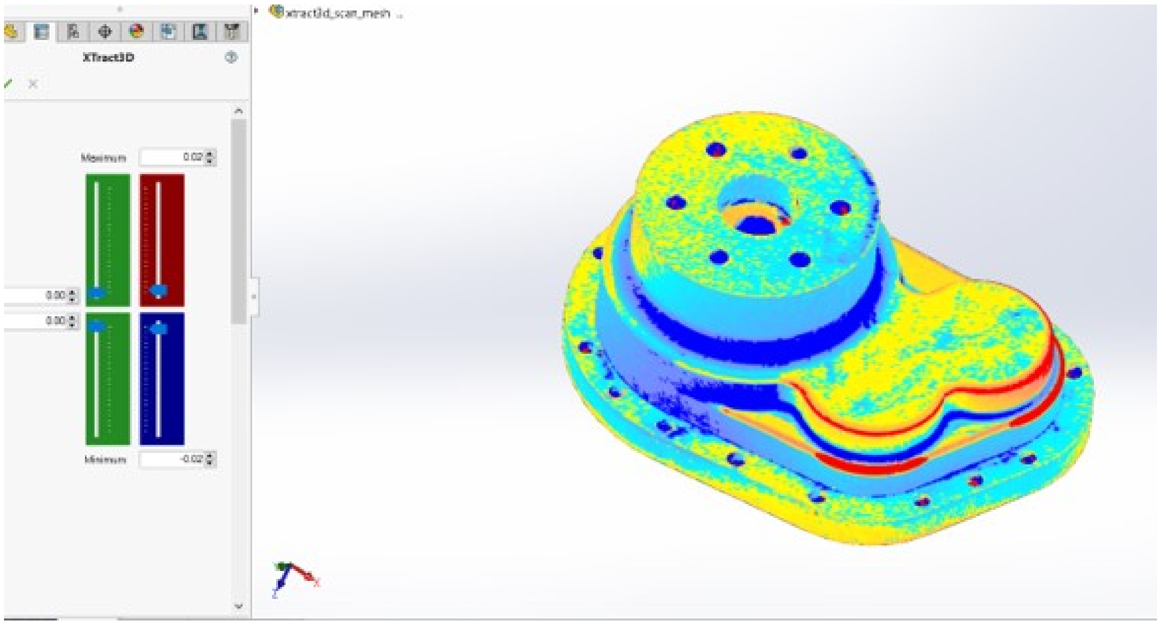


Fig. 6.15. Mesh Deviation Analysis between Photogrammetry Mesh and Modeled Part of Gearbox

6.3 Quantitative Analysis

After modelling parts using scan mesh and photogrammetry mesh as reference, the quantitative validation of the suggested methodology was performed. This validation procedure includes comparison of scan mesh - actual model and photogrammetry mesh to actual model.

Later on these reverse engineered CAD models were analysed for relative accuracy which can be seen in Fig. 6.16 and 6.18. For CAD comparison, cloud compare has Mesh-Mesh to comparative method. Thus, reverse engineered CAD models first converted to water-tight knit surface model and later exported to corresponding .stl files.

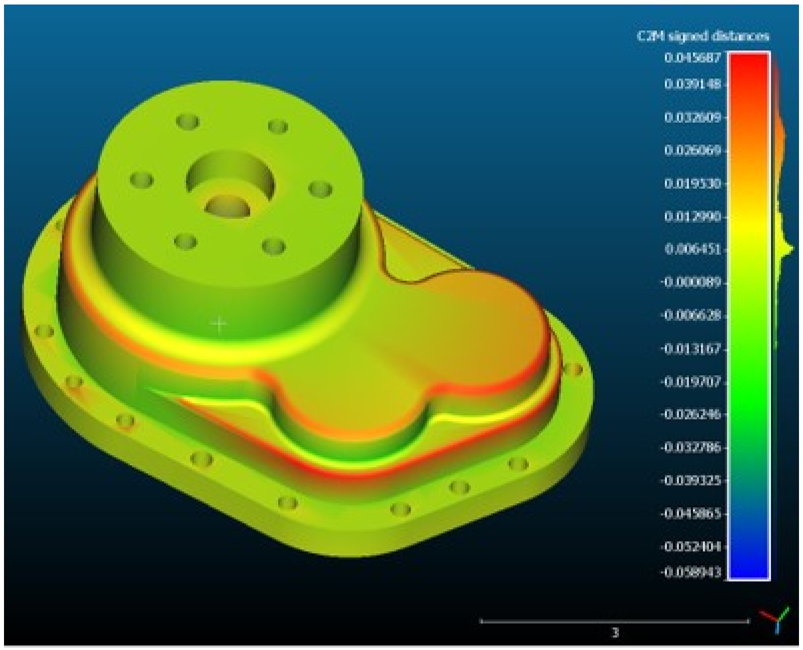


Fig. 6.16. `CAD to CAD Deviation Analysis between Scan - Reverse Model and Actual Model of Gearbox`

Actual CAD model was used as reference mesh and scan-reverse model/ photogrammetry model was used as sample mesh which needs to be compared with reference mesh. Using more than three points mesh were aligned with scale and translation adjustments. After registering these meshes were sampled against actual

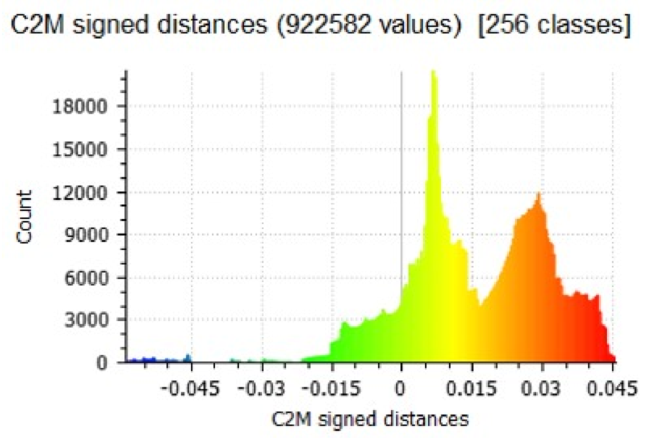


Fig. 6.17. `CAD to CAD Deviation Analysis - Statistical Values for Scan - Reverse model`

CAD model for deviation analysis. In both cases (photogrammetry-reverse model and scan-reverse model) number of sampling was set to 5,000,000 in order to reduce RMS value of alignment error and comparison.

Along with Mesh deviation contour image, their corresponding statistical values and normalized distribution of deviations are shown in Fig. 6.17 and 6.19. These deviations values were then exported to excel file and segmented in 8 absolute value range. Fig. 6.20 shows the graph of these deviation range against number of faces (values). This graph represents comparison of reverse CAD models versus actual CAD model of Gearbox Casing.

In the Fig. 6.21, the graph shows the parametric study conducted for Number of Images of object which is to be reconstructed versus accuracy of corresponding reverse CAD models. For Gearbox casing, in total 70 images were captured at various levels and angles. Then these photos were selected alternately for reconstruction from 60 images, 50 images, 40 images and so on. Similar to previous Mesh to Mesh comparison in CloudCompare tool, corresponding results were obtained and plotted against actual CAD file for comparison.

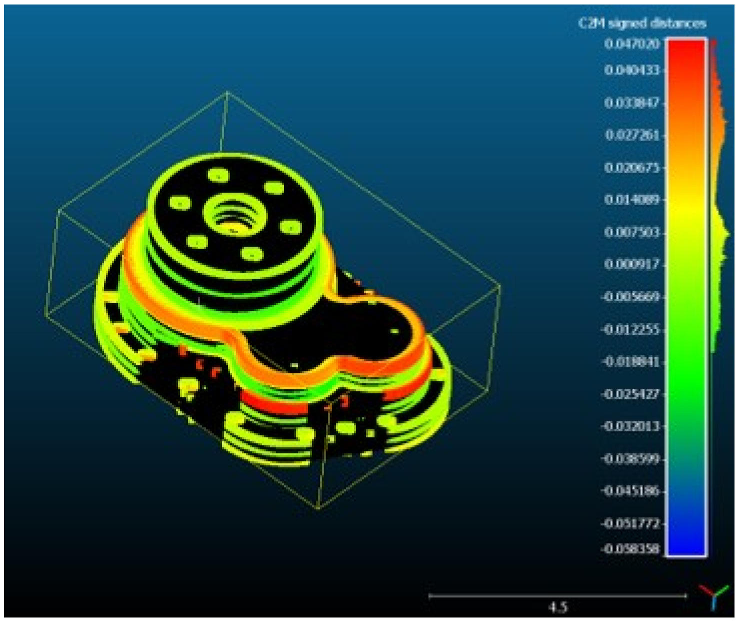


Fig. 6.18. CAD to CAD Deviation Analysis between Photogrammetry - Reverse Model and Actual Model of Gearbox

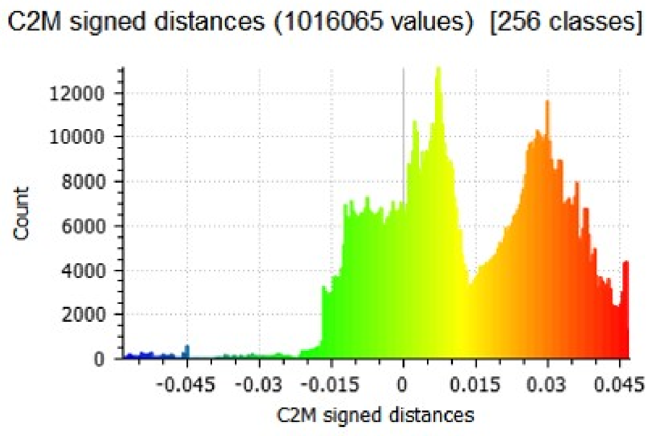


Fig. 6.19. Statistical Values for Photogrammetry-Reverse model

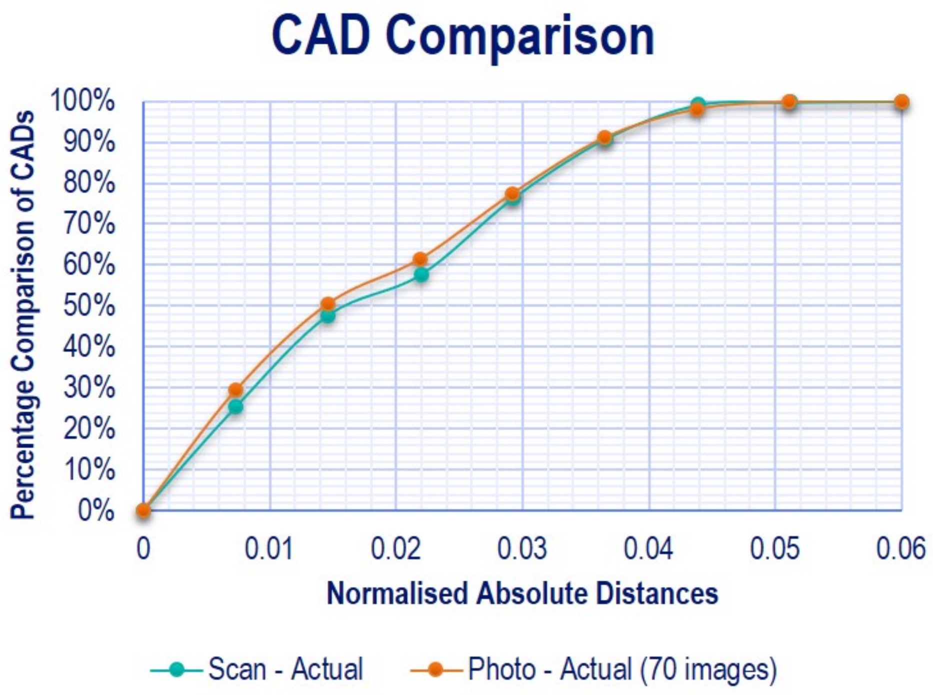


Fig. 6.20. CAD Comparison Analysis Graph

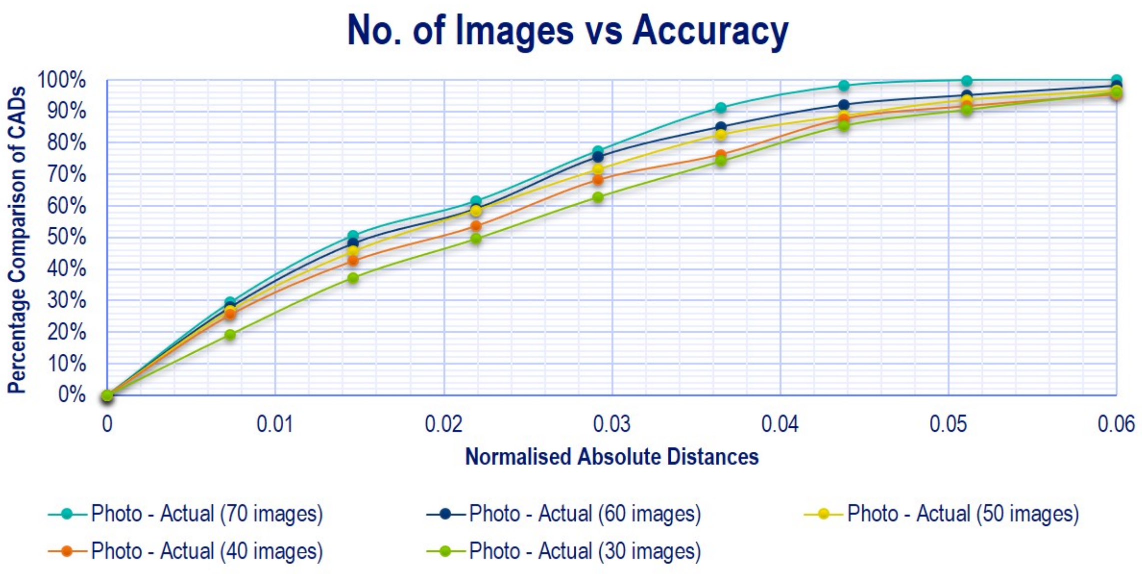


Fig. 6.21. Parametric Study - No. of Images vs Accuracy

7. RESULTS AND CONCLUSION

As presented in previous chapter, Meshroom Photogrammetry tool has selected based on comparative accuracy with scan results. This Meshroom pipeline was successfully integrated with SOLIDWORKS to make an reverse engineering workflow using photogrammetry. In order to test this workflow, two case studies has been presented. Following paragraphs comment on results obtained for case studies and their inference towards objective of this thesis.

The mesh deviation within XTRACT3D (reverse engineered model) resulted between scan data and modeled part for Case Study I (Flywheel) shows the maximum deviation of 1.61mm and minimum deviation of -1.97 mm. Also the mesh deviation for Case Study II(Gearbox casing) falls in between maximum of 2.452116mm to 0.1976882 mm. As per these case results, that the deviation of this photogrammetry reverse engineered model is near about +/- 2.5 mm.

This deviation value can be further reduced below 1 mm, but this depends on the quality of photogrammetry mesh obtained. Quality of this photogrammetric mesh is dependent on number of images, the default parameters set inside photogrammetry methodology, calibration of camera device for correct settings, image acquisition procedure.

In order to enhance the quality of this photogrammetry mesh the number of images has to be higher than 70 for Case Study II and more than 40 for case study one. The parametric study for number of images vs accuracy which has been presented with graph claims the same thing.

For this Gearbox case (Case Study II), I had actual CAD model of Gearbox casing. This actual CAD model was set as reference for CAD comparison. In the Fig.6.16, scan-reverse model has maximum deviation of 0.0727567 inches (1.84802018 mm), average deviation of 0.00837128 inches(0.212630512mm) with standard error 0.0190088 inches (0.48282352mm) when compared to actual CAD model.

While in the case of photogrammetry-reverse model(70 image), maximum deviation of 0.0727567 inches (1.84802018 mm), average deviation of 0.00896533 inches (0.227719382mm) with standard error 0.019532 inches (0.4961128mm) when compared to actual CAD model. These statistical values are higher for photogrammetry-reverse model as there are higher number of small deviations.

Based on result, it is imminent that photogrammetry-reverse model has higher average and standard deviation because of the higher number of smaller deviation values. Deviation values for photogrammetry-reverse model are more evenly spread out than scan-reverse model.

When these absolute deviation values were segmented and plotted for CAD accuracy, it is found that photogrammetry-reverse model is giving similar accuracy like scan-reverse model. Even, the photogrammetry-reverse model is performing little better because it has higher number of small deviations. Thus from this graph, it can be inferred that higher number of small deviations values and smaller number of high deviation values lead to better accuracy of reverse-engineered model.

A parametric study of photogrammetric-reverse model has evaluated based on effect on accuracy by number of images. Various deviation values have been segmented and plotted as shown in Fig. 6.21. From this graph it is very much clear that, reducing number of images for reconstruction results into higher number of high deviation values. This means, higher number of images will produce higher accuracy reverse-engineered model. However, higher number of images increases computational time.

This research further needs the in-detailed study of the photogrammetry algorithm and parameters affecting its result. In this thesis, it can be concluded that meshroom provided logical close solution for creating photogrammetric workflow within single environment.

Through case studies, capability and procedure for reverse engineering from photogrammetric object reconstruction has been portrayed. This photogrammetric workflow closely matches with scan results. Hence considering higher investment cost for scanner, it can be concluded that this photogrammetry pipeline can be used as substitute.

It can also be concluded that, objectives like creating integrated workflow, parametric study and scan result comparison has been achieved.

8. FUTURE WORK

Limitations that have been found during this thesis research need to be addressed. Moreover, there is scope for improvement in the photogrammetry methodology as well as in other areas. Future work does include the following points

1. Advance add-in development with more freedom for choosing nodes' parameters.
2. Live reconstruction capability inside SOLIDWORKS.
3. Developement of quality inspection tool within SOLIDWORKS.
4. Detailed reserach about camera sensor calibration and choosing correct camera.
5. Preparation of the correct test setup in the mechanical industries with custom applications.
6. Research related to algorithm and computer vision so that computing cost can be reduced significantly.
7. Quality of 3D printed parts from reconstructed 3D objects from image acquisition.

REFERENCES

REFERENCES

[1] S. Slater and H. Childs, "Photorealistic rendering utilizing close-range photogrammetry," Ph.D. dissertation, University of Oregon, 2016.

[2] J. Braybon, "Traversing the unsw campus using terrestrial photogrammetry," Ph.D. dissertation, UNSW School of Surveying and Spatial Information Systems, Sydney , 2011.

[3] P. Pejić, S. Krasić, H. Krstić, M. Dragović, and Y. Akbiyik, "3d virtual modelling of existing objects by terrestrial photogrammetric methods-case study of barutana," *Tehnički vjesnik*, vol. 24, no. 1, pp. 233–239, 2017.

[4] H. J. Koelman, "Application of a photogrammetry-based system to measure and re-engineer ship hulls and ship parts: An industrial practices-based report," *Computer-Aided Design*, vol. 42, no. 8, pp. 731 – 743, 2010.

[5] S. Syring and J. Nylund, "Introducing uas and photogrammetry for surveying and surveilling new project sites," Ph.D. dissertation, YAKESHOGSKOLAN NOVIA, 2018.

[6] T. Liu, A. W. Burner, T. W. Jones, and D. A. Barrows, "Photogrammetric techniques for aerospace applications," *Progress in Aerospace Sciences*, vol. 54, pp. 1 – 58, 2012.

[7] A. Abdelhafiz, "Integrating digital photogrammetry and terrestrial laser scanning," Ph.D. dissertation, Assiut University, February 2009.

[8] C. Duke, "Evaluation of photogrammetry at different scales," Ph.D. dissertation, Auburn University, 2018.

[9] S. Gerbino, M. Martorelli, F. Renno, D. Speranza *et al.*, "Cheap photogrammetry versus expensive reverse engineering techinques in 3d model acquisition and shape reconstruction," in *DS 32: Proceedings of DESIGN 2004, the 8th International Design Conference, Dubrovnik, Croatia*, 2004, pp. 749–754.

[10] X. Jing, C. Zhang, Z. Sun, G. Zhao, and Y. Wang, "The technologies of close-range photogrammetry and application in manufacture," in *3rd International Conference on Mechatronics, Robotics and Automation.* Atlantis Press, 2015, pp. 1–7.

[11] O. C. Martin, S. Robson, A. Kayani, J. E. Muelaner, V. Dhokia, and P. G. Maropoulos, "Comparative performance between two photogrammetric systems and a reference laser tracker network for large-volume industrial measurement," *The Photogrammetric Record*, vol. 31, no. 155, pp. 348–360, 2016.

[12] LINEARIS3D, "Photogrammetry-system [portable 3d- coordinate measuring system," 2016. [Online]. Retrieved August, 2019 from: http://www.linearis3d.com/common/downloads/Linearis3D_Photogrammetry_EN.pdf

[13] I. HORVTH and J. S. M. VERGEEST, "Natural representation of shapes with singularities," *International Journal of Shape Modeling*, vol. 03, no. 03–04, pp. 127–139, 1997.

[14] F. Remondino and S. El-Hakim, "Image-based 3d modelling: a review," *The Photogrammetric Record*, vol. 21, no. 115, pp. 269–291, 2006.

[15] C. Kidson and M. Manton, "Assessment of coastal change with the aid of photogrammetric and computer-aided techniques," *Estuarine and Coastal Marine Science*, vol. 1, no. 3, pp. 271–283, 1973.

[16] S. F. El-Hakim, P. Boulanger, F. Blais, and J. A. Beraldin, "System for indoor 3d mapping and virtual environments," in *Videometrics V*, vol. 3174. International Society for Optics and Photonics, 1997, pp. 21–36.

[17] S. F. El-Hakim, C. Brenner, and G. Roth, "An approach to creating virtual environments using range and texture," *International Archives of Photogrammetry and Remote Sensing*, vol. 32, pp. 331–338, 1998.

[18] C. Ogleby, "Olympia-home of the ancient and modern olympic games a virtual reality three dimensional experience," *International Archives of Photogrammetry and Remote Sensing*, vol. 34, no. 5/W1, pp. 97–102, 2001.

[19] T. Luhmann, "Close range photogrammetry for industrial applications," *ISPRS journal of photogrammetry and remote sensing*, vol. 65, no. 6, pp. 558–569, 2010.

[20] K. B. Atkinson, "Close range photogrammetry and machine vision," Whittles Publishing, Tech. Rep., 2001.

[21] M. Rodrguez-Martn, S. Lagela, G. Gonzlez-Aguilera, and P. Rodrguez-Gonzlvez, "Procedure for quality inspection of welds based on macro-photogrammetric three-dimensional reconstruction," *Optics and Laser Technology*, vol. 73, pp. 54–62, 2015.

[22] W. Bösemann, "Industrial Photogrammetry - Accepted Metrology Tool or Exotic Niche," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pp. 15–24, Jun. 2016.

[23] J. Armesto, I. Lubowiecka, C. Ordez, and F. Rial, "Fem modeling of structures based on close range digital photogrammetry," *Automation in Construction*, vol. 18, pp. 559–569, 08 2009.

[24] G. Kortaberria, U. Mutilba, E. Gomez-Acedo, A. Tellaeche, and R. Minguez, "Accuracy evaluation of dense matching techniques for casting part dimensional verification," in *Sensors*, vol. 18, no. 9, September 2018, pp. 30–74.

[25] M. G. Guerra, "Analysis of a 3d optical scanner based on photogrammetry suitable for industrial applications in close and micro-range," Ph.D. dissertation, Politecnico di Bari, 08 2018.

[26] F. Grski, R. Kuczko, W.and Wichniarek, and P. Zawadzki, "Application of close-range photogrammetry in reverse engineering," in *7th International DAAAM Baltic Conference*, vol. 1, 2010, pp. 30–35.

[27] H. E.-D. Fawzy, "The accuracy of mobile phone camera instead of high resolution camera in digital close range photogrammetry," *International Journal of Civil Engineering & Technology (IJCIET)*, vol. 6, no. 1, pp. 76–85, 2015.

[28] M. von bel, "16 best photogrammetry software tools in 2019," 2019. [Online]. Retrieved August, 2018: https://all3dp.com/1/best-photogrammetry-software/

[29] D. P. L. FALKINGHAM, "Free photogrammetry software review: 2017," 2017. [Online]. Retrieved August, 2019 from: https:// peterfalkingham.com/2017/12/17/free-photogrammetry-software-review-2017/

[30] P. Shaffner, L. Krosley, and J. Kottenstette, "Us bureau of r eclamation digital photogrammetry r esearch r eport," 2004.

[31] MicMac,"Micmac photogrammetry," 2017. [Online]. Retrieved August, 2019 from: https://micmac.ensg.eu/index.php/Gravillonstutorial

[32] R. Feldman,"List of free photogrammetry software,"2019.[Online]. Retrieved August,2019 from: https://www.3dbeginners.com/free-photogrammetry-software

[33] A. Meshroom, "Meshroom documentation," 2018. [Online]. Retrieved August, 2019 from: https://docs.google.com/document/ d/17HYtYS1tvx053k3nO6Z2GnP2R3cXMlGMN1WIe3kJEeditheading=h.av4e0ukv6

[34] Wikipedia, "Json format," 2018. [Online]. Retrieved August, 2019 from: https://en.wikipedia.org/wiki/JSON

[35] SOLIDWORKS, "Solidworks," 2018. [Online]. Retrieved August, 2019 from: https://en.wikipedia.org/wiki/SolidWorks

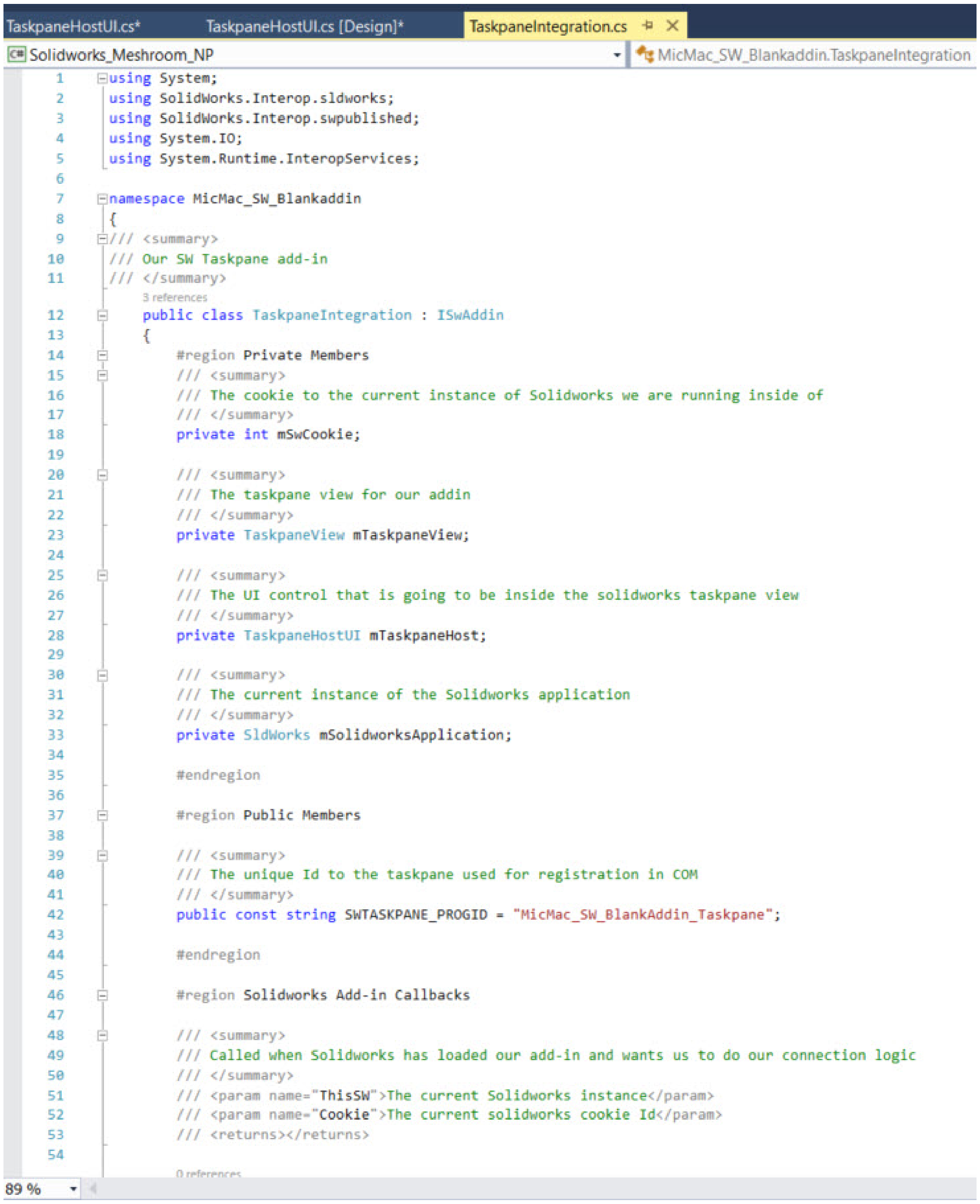
[36] CSharp, "Csharp," 2018. [Online]. Retrieved August, 2019 from: https://en.wikipedia.org/wiki/CSharp(programminglanguage)

[37] SOLIDWORKS, "Solidworksapi," 2018. [Online]. Retrieved August, 2019 from: http://help.solidworks.com/2019/english/api/sldworksapiprogguide/welcome.ht ml

[38] P. Christensson, "Dll definition," 2006. [Online]. Retrieved August, 2019 from: https://techterms.com/definition/dll)

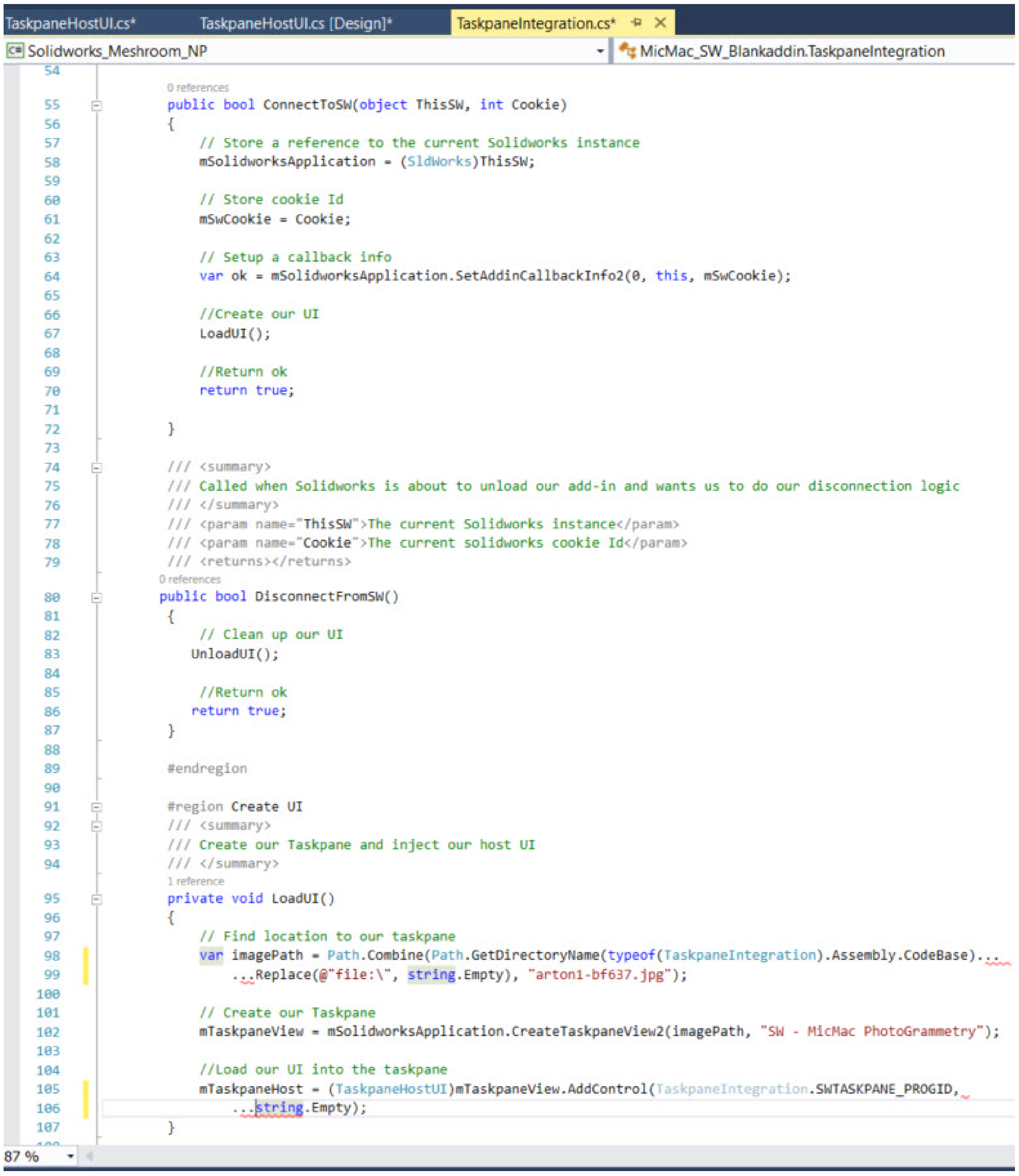
APPENDICES

A. SOURCE CODE


TaskpaneHostUI.cs*
TaskpaneHostUI.cs [Design]*
TaskpaneIntegration.cs
Solidworks_Meshroom_NP
MicMac_SW_Blankaddin.TaskpaneIntegration
89 %

```
using System;
using SolidWorks.Interop.sldworks;
using SolidWorks.Interop.swpublished;
using System.IO;
using System.Runtime.InteropServices;

namespace MicMac_SW_Blankaddin
{
/// <summary>
/// Our SW Taskpane add-in
/// </summary>
    3 references
    public class TaskpaneIntegration : ISwAddin
    {
        #region Private Members
        /// <summary>
        /// The cookie to the current instance of Solidworks we are running inside of
        /// </summary>
        private int mSwCookie;

        /// <summary>
        /// The taskpane view for our addin
        /// </summary>
        private TaskpaneView mTaskpaneView;

        /// <summary>
        /// The UI control that is going to be inside the solidworks taskpane view
        /// </summary>
        private TaskpaneHostUI mTaskpaneHost;

        /// <summary>
        /// The current instance of the Solidworks application
        /// </summary>
        private SldWorks mSolidworksApplication;

        #endregion

        #region Public Members

        /// <summary>
        /// The unique Id to the taskpane used for registration in COM
        /// </summary>
        public const string SWTASKPANE_PROGID = "MicMac_SW_BlankAddin_Taskpane";

        #endregion

        #region Solidworks Add-in Callbacks

        /// <summary>
        /// Called when Solidworks has loaded our add-in and wants us to do our connection logic
        /// </summary>
        /// <param name="ThisSW">The current Solidworks instance</param>
        /// <param name="Cookie">The current solidworks cookie Id</param>
        /// <returns></returns>

        0 references
```

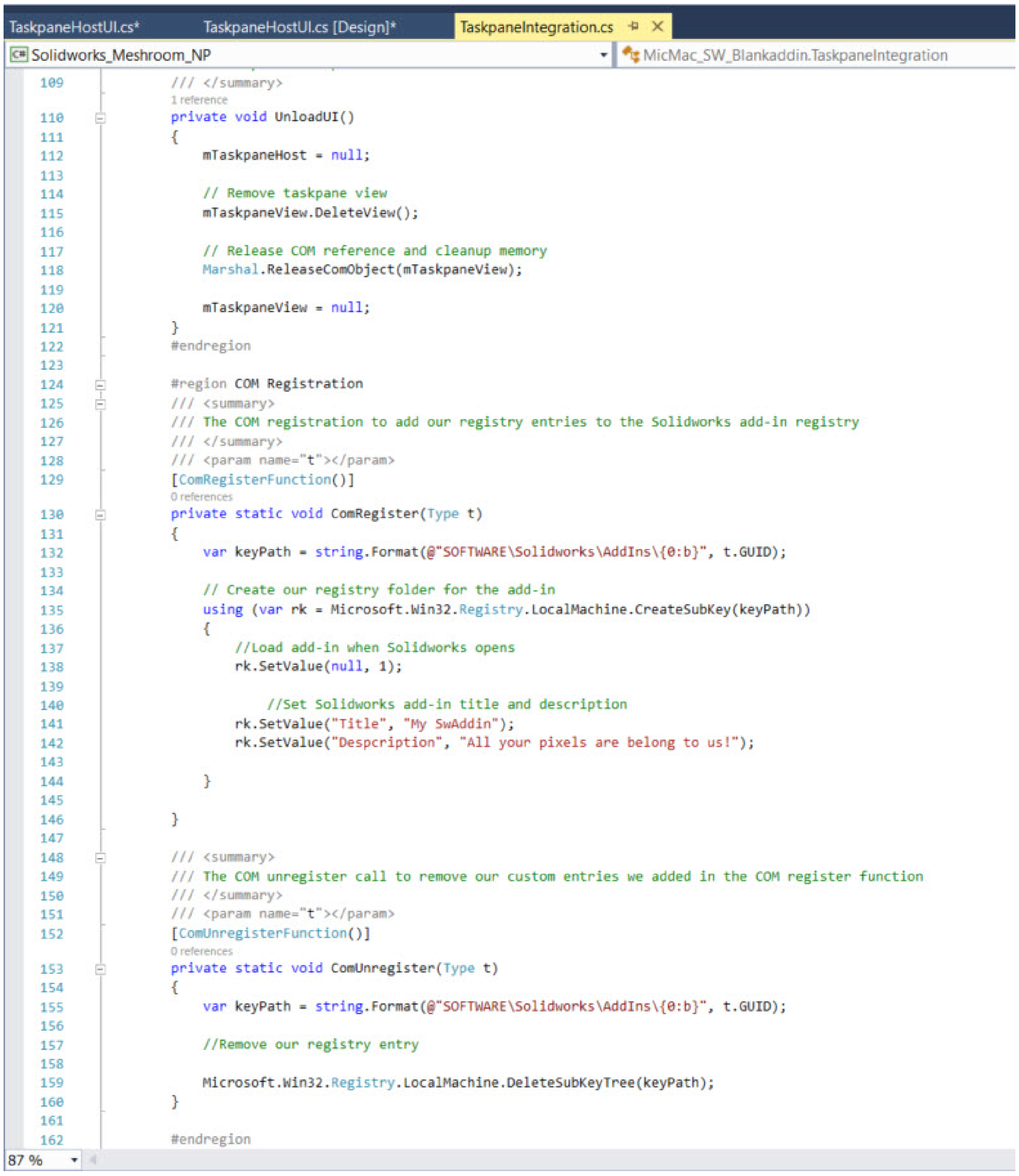
Fig. A.1. Add-in Source code 1of3



TaskpaneHostUI.cs*
TaskpaneHostUI.cs [Design]*
TaskpaneIntegration.cs*
Solidworks_Meshroom_NP
MicMac_SW_Blankaddin.TaskpaneIntegration
87 %

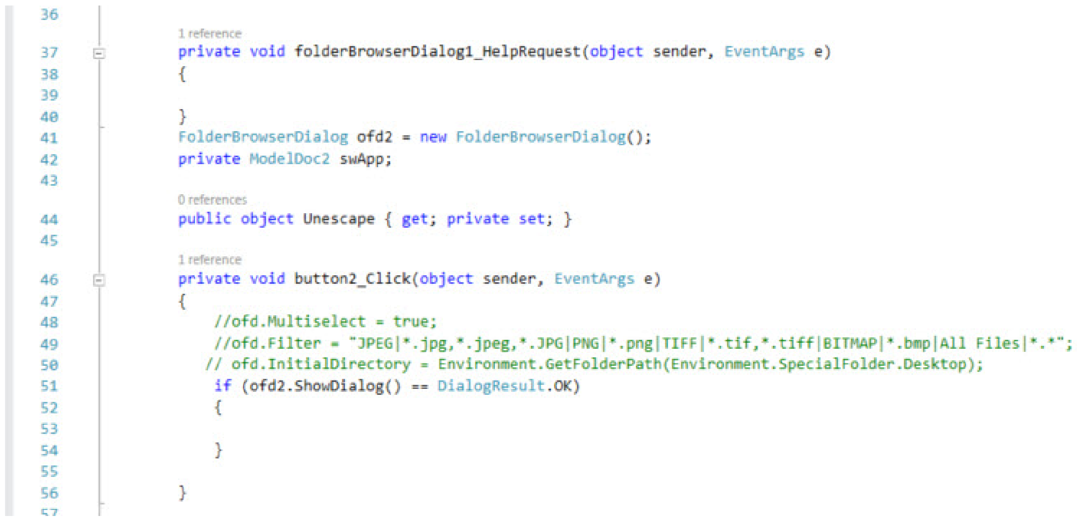
```
        0 references
        public bool ConnectToSW(object ThisSW, int Cookie)
        {
            // Store a reference to the current Solidworks instance
            mSolidworksApplication = (SldWorks)ThisSW;

            // Store cookie Id
            mSwCookie = Cookie;

            // Setup a callback info
            var ok = mSolidworksApplication.SetAddinCallbackInfo2(0, this, mSwCookie);

            //Create our UI
            LoadUI();

            //Return ok
            return true;

        }

        /// <summary>
        /// Called when Solidworks is about to unload our add-in and wants us to do our disconnection logic
        /// </summary>
        /// <param name="ThisSW">The current Solidworks instance</param>
        /// <param name="Cookie">The current solidworks cookie Id</param>
        /// <returns></returns>
       0 references
       public bool DisconnectFromSW()
        {
            // Clean up our UI
           UnloadUI();

            //Return ok
           return true;
        }

        #endregion

        #region Create UI
        /// <summary>
        /// Create our Taskpane and inject our host UI
        /// </summary>
        1 reference
        private void LoadUI()
        {
            // Find location to our taskpane
            var imagePath = Path.Combine(Path.GetDirectoryName(typeof(TaskpaneIntegration).Assembly.CodeBase)...
                ...Replace(@"file:\", string.Empty), "arton1-bf637.jpg");

            // Create our Taskpane
            mTaskpaneView = mSolidworksApplication.CreateTaskpaneView2(imagePath, "SW - MicMac PhotoGrammetry");

            //Load our UI into the taskpane
            mTaskpaneHost = (TaskpaneHostUI)mTaskpaneView.AddControl(TaskpaneIntegration.SWTASKPANE_PROGID,
                ...string.Empty);
        }
```

Fig. A.2. `Add-in Source code 2of3`


TaskpaneHostUI.cs*
TaskpaneHostUI.cs [Design]*
TaskpaneIntegration.cs
Solidworks_Meshroom_NP
MicMac_SW_Blankaddin.TaskpaneIntegration
87 %

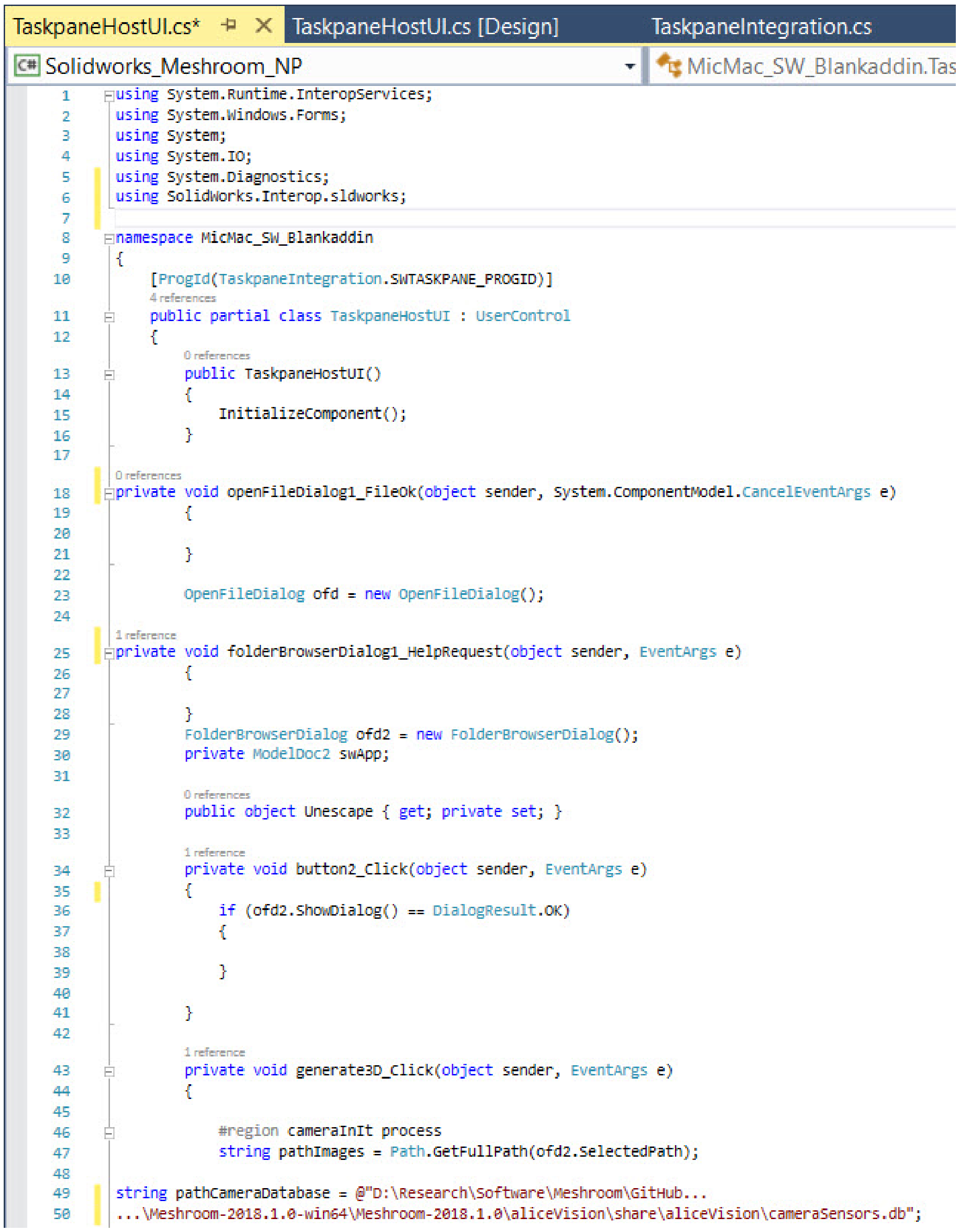
```
        /// </summary>
        1 reference
        private void UnloadUI()
        {
            mTaskpaneHost = null;

            // Remove taskpane view
            mTaskpaneView.DeleteView();

            // Release COM reference and cleanup memory
            Marshal.ReleaseComObject(mTaskpaneView);

            mTaskpaneView = null;
        }
        #endregion

        #region COM Registration
        /// <summary>
        /// The COM registration to add our registry entries to the Solidworks add-in registry
        /// </summary>
        /// <param name="t"></param>
        [ComRegisterFunction()]
        0 references
        private static void ComRegister(Type t)
        {
            var keyPath = string.Format(@"SOFTWARE\Solidworks\AddIns\{0:b}", t.GUID);

            // Create our registry folder for the add-in
            using (var rk = Microsoft.Win32.Registry.LocalMachine.CreateSubKey(keyPath))
            {
                //Load add-in when Solidworks opens
                rk.SetValue(null, 1);

                    //Set Solidworks add-in title and description
                rk.SetValue("Title", "My SwAddin");
                rk.SetValue("Despcription", "All your pixels are belong to us!");

            }

        }

        /// <summary>
        /// The COM unregister call to remove our custom entries we added in the COM register function
        /// </summary>
        /// <param name="t"></param>
        [ComUnregisterFunction()]
        0 references
        private static void ComUnregister(Type t)
        {
            var keyPath = string.Format(@"SOFTWARE\Solidworks\AddIns\{0:b}", t.GUID);

            //Remove our registry entry

            Microsoft.Win32.Registry.LocalMachine.DeleteSubKeyTree(keyPath);
        }

        #endregion
```

Fig. A.3. Add-in Source code 3of3



```
        1 reference
        private void folderBrowserDialog1_HelpRequest(object sender, EventArgs e)
        {

        }
        FolderBrowserDialog ofd2 = new FolderBrowserDialog();
        private ModelDoc2 swApp;

        0 references
        public object Unescape { get; private set; }

        1 reference
        private void button2_Click(object sender, EventArgs e)
        {
            //ofd.Multiselect = true;
            //ofd.Filter = "JPEG|*.jpg,*.jpeg,*.JPG|PNG|*.png|TIFF|*.tif,*.tiff|BITMAP|*.bmp|All Files|*.*";
           // ofd.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
            if (ofd2.ShowDialog() == DialogResult.OK)
            {

            }

        }
```

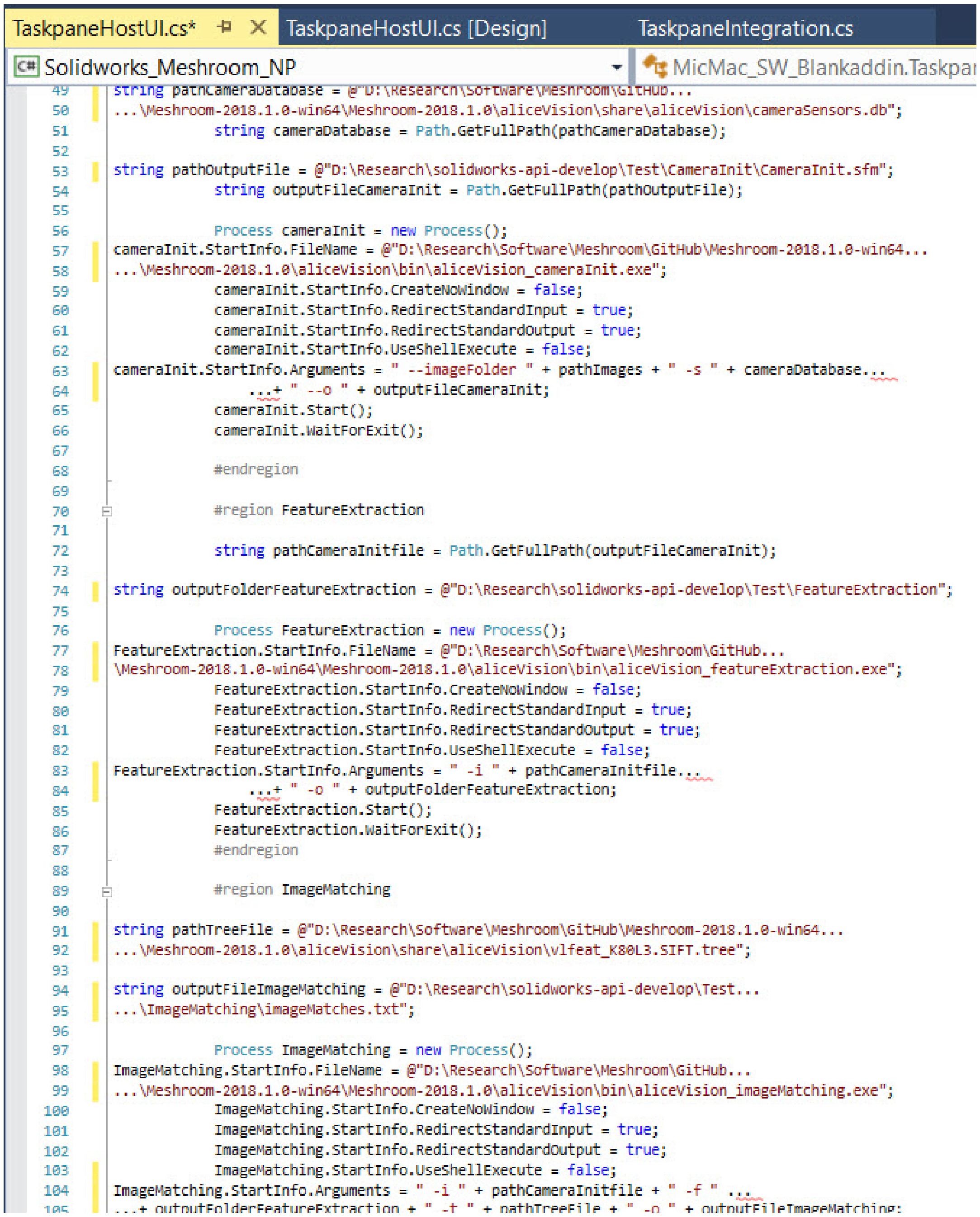
Fig. A.4. Open Folder Browser Dialog Code


TaskpaneHostUI.cs* TaskpaneHostUI.cs [Design] TaskpaneIntegration.cs

Solidworks_Meshroom_NP MicMac_SW_Blankaddin.Tas

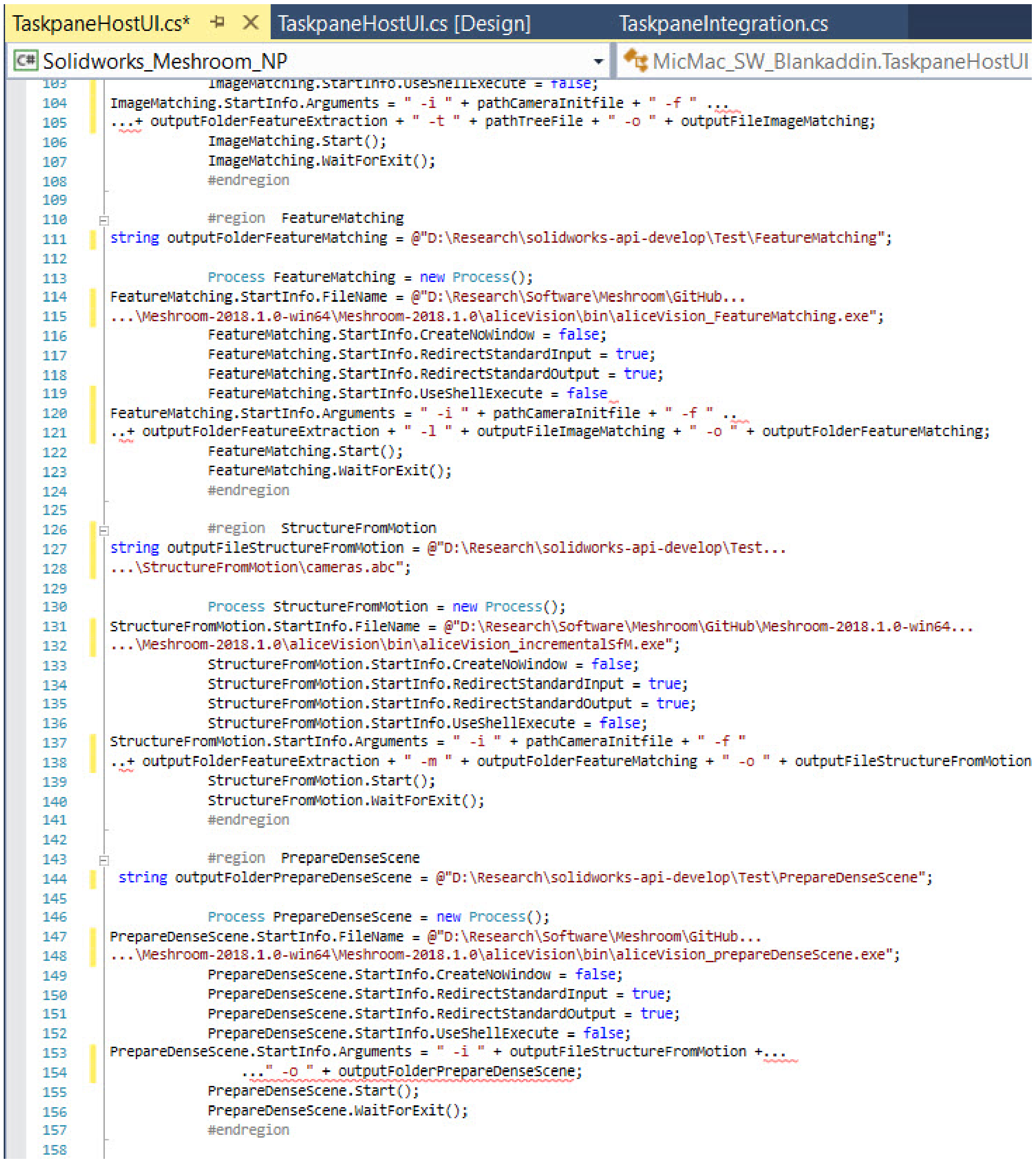
```
using System.Runtime.InteropServices;
using System.Windows.Forms;
using System;
using System.IO;
using System.Diagnostics;
using SolidWorks.Interop.sldworks;

namespace MicMac_SW_Blankaddin
{
    [ProgId(TaskpaneIntegration.SWTASKPANE_PROGID)]
    4 references
    public partial class TaskpaneHostUI : UserControl
    {
        0 references
        public TaskpaneHostUI()
        {
            InitializeComponent();
        }

0 references
private void openFileDialog1_FileOk(object sender, System.ComponentModel.CancelEventArgs e)
        {

        }

        OpenFileDialog ofd = new OpenFileDialog();

1 reference
private void folderBrowserDialog1_HelpRequest(object sender, EventArgs e)
        {

        }
        FolderBrowserDialog ofd2 = new FolderBrowserDialog();
        private ModelDoc2 swApp;

        0 references
        public object Unescape { get; private set; }

        1 reference
        private void button2_Click(object sender, EventArgs e)
        {
            if (ofd2.ShowDialog() == DialogResult.OK)
            {

            }

        }

        1 reference
        private void generate3D_Click(object sender, EventArgs e)
        {

            #region cameraInIt process
            string pathImages = Path.GetFullPath(ofd2.SelectedPath);

string pathCameraDatabase = @"D:\Research\Software\Meshroom\GitHub...
...\Meshroom-2018.1.0-win64\Meshroom-2018.1.0\aliceVision\share\aliceVision\cameraSensors.db";
```

Fig. A.5. Photogrammetry Pipeline code 1 of 5



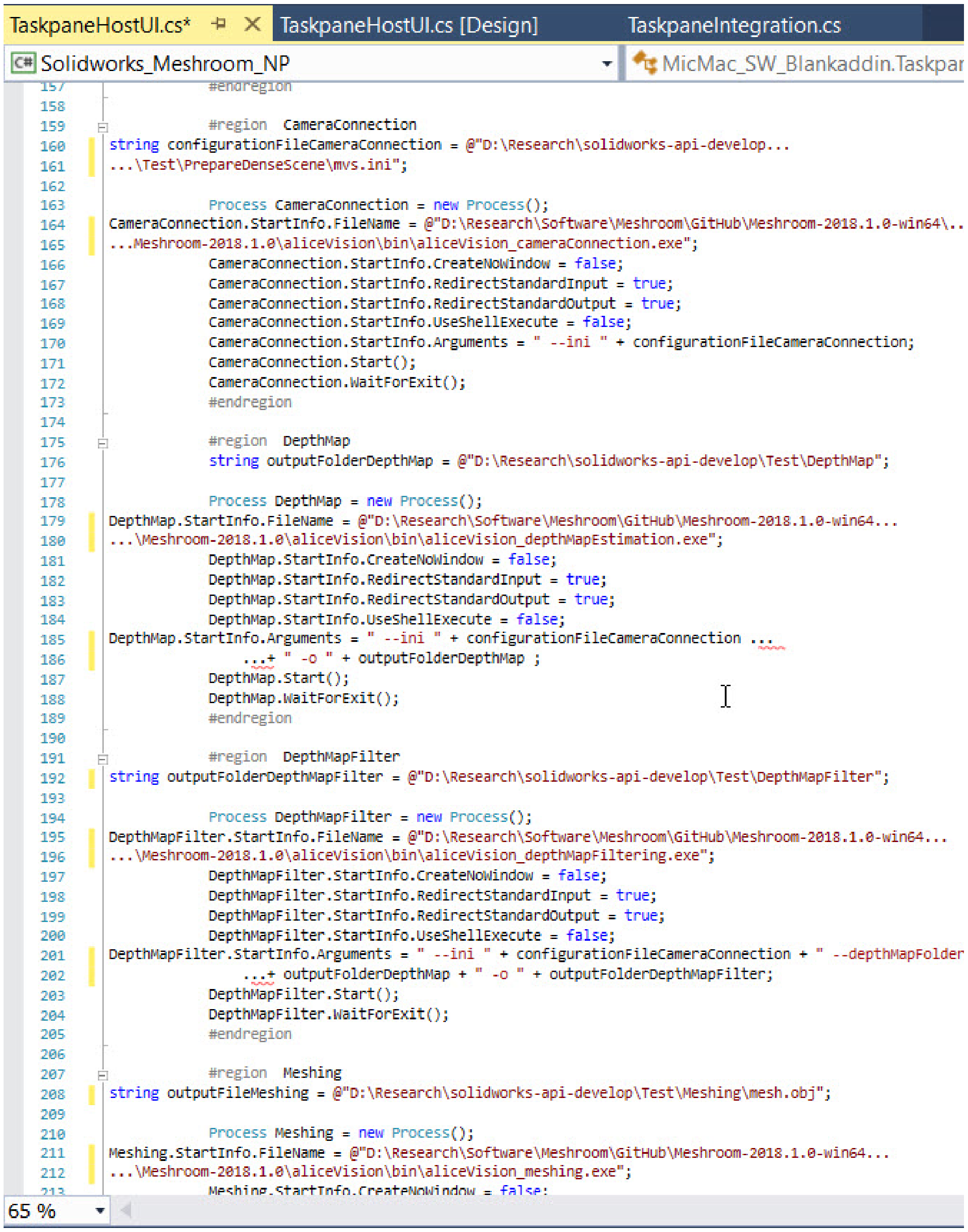
```
string pathCameraDatabase = @"D:\Research\Software\Meshroom\GitHub...
...\Meshroom-2018.1.0-win64\Meshroom-2018.1.0\aliceVision\share\aliceVision\cameraSensors.db";
            string cameraDatabase = Path.GetFullPath(pathCameraDatabase);

string pathOutputFile = @"D:\Research\solidworks-api-develop\Test\CameraInit\CameraInit.sfm";
            string outputFileCameraInit = Path.GetFullPath(pathOutputFile);

            Process cameraInit = new Process();
cameraInit.StartInfo.FileName = @"D:\Research\Software\Meshroom\GitHub\Meshroom-2018.1.0-win64...
...\Meshroom-2018.1.0\aliceVision\bin\aliceVision_cameraInit.exe";
            cameraInit.StartInfo.CreateNoWindow = false;
            cameraInit.StartInfo.RedirectStandardInput = true;
            cameraInit.StartInfo.RedirectStandardOutput = true;
            cameraInit.StartInfo.UseShellExecute = false;
cameraInit.StartInfo.Arguments = " --imageFolder " + pathImages + " -s " + cameraDatabase...
                ...+ " --o " + outputFileCameraInit;
            cameraInit.Start();
            cameraInit.WaitForExit();

            #endregion

            #region FeatureExtraction

            string pathCameraInitfile = Path.GetFullPath(outputFileCameraInit);

string outputFolderFeatureExtraction = @"D:\Research\solidworks-api-develop\Test\FeatureExtraction";

            Process FeatureExtraction = new Process();
FeatureExtraction.StartInfo.FileName = @"D:\Research\Software\Meshroom\GitHub...
\Meshroom-2018.1.0-win64\Meshroom-2018.1.0\aliceVision\bin\aliceVision_featureExtraction.exe";
            FeatureExtraction.StartInfo.CreateNoWindow = false;
            FeatureExtraction.StartInfo.RedirectStandardInput = true;
            FeatureExtraction.StartInfo.RedirectStandardOutput = true;
            FeatureExtraction.StartInfo.UseShellExecute = false;
FeatureExtraction.StartInfo.Arguments = " -i " + pathCameraInitfile...
                ...+ " -o " + outputFolderFeatureExtraction;
            FeatureExtraction.Start();
            FeatureExtraction.WaitForExit();
            #endregion

            #region ImageMatching

string pathTreeFile = @"D:\Research\Software\Meshroom\GitHub\Meshroom-2018.1.0-win64...
...\Meshroom-2018.1.0\aliceVision\share\aliceVision\vlfeat_K80L3.SIFT.tree";

string outputFileImageMatching = @"D:\Research\solidworks-api-develop\Test...
...\ImageMatching\imageMatches.txt";

            Process ImageMatching = new Process();
ImageMatching.StartInfo.FileName = @"D:\Research\Software\Meshroom\GitHub...
...\Meshroom-2018.1.0-win64\Meshroom-2018.1.0\aliceVision\bin\aliceVision_imageMatching.exe";
            ImageMatching.StartInfo.CreateNoWindow = false;
            ImageMatching.StartInfo.RedirectStandardInput = true;
            ImageMatching.StartInfo.RedirectStandardOutput = true;
            ImageMatching.StartInfo.UseShellExecute = false;
ImageMatching.StartInfo.Arguments = " -i " + pathCameraInitfile + " -f " ...
...+ outputFolderFeatureExtraction + " -t " + pathTreeFile + " -o " + outputFileImageMatching;
```

Fig. A.6. Photogrammetry Pipeline code 2 of 5



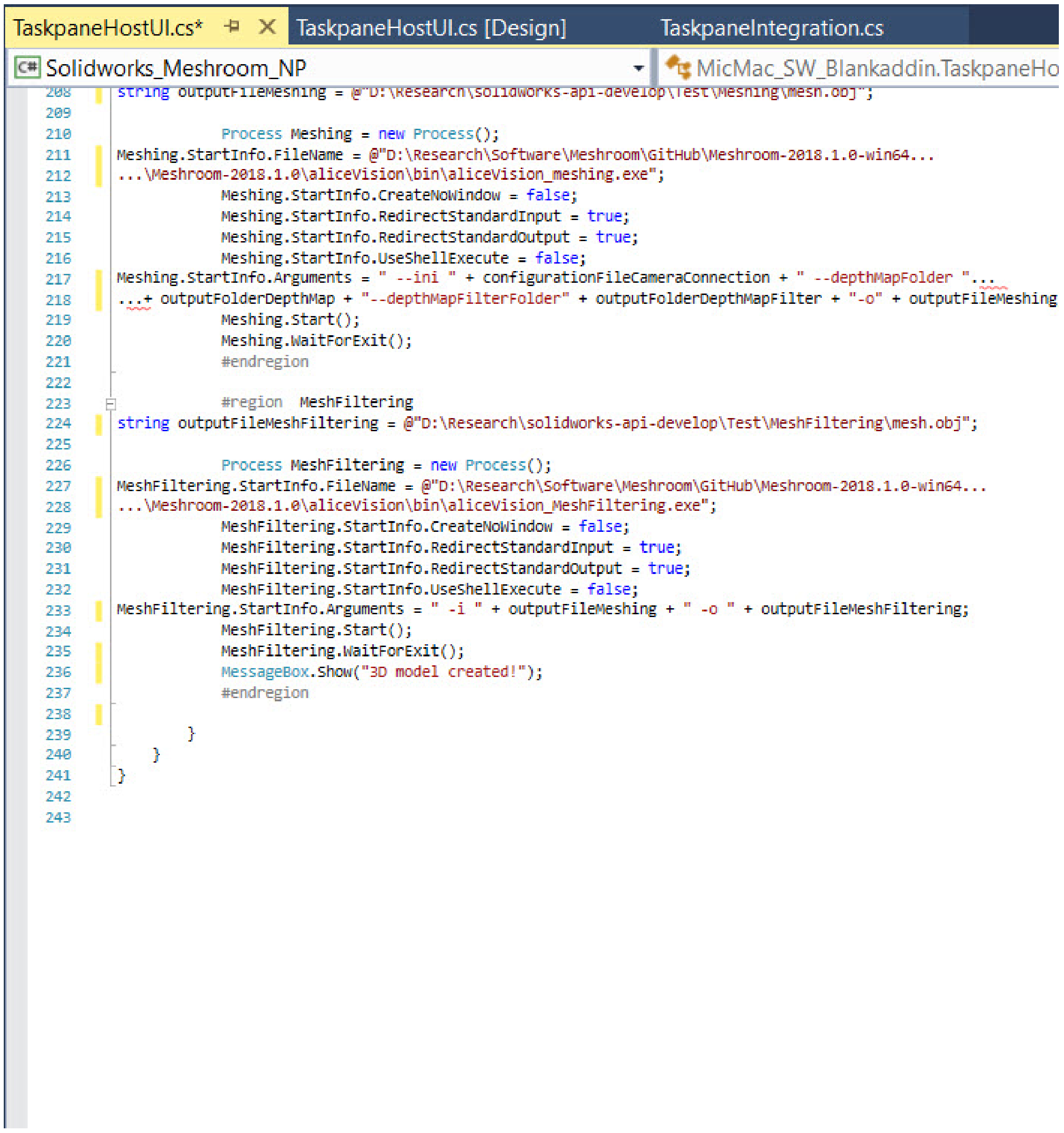
```
            ImageMatching.StartInfo.UseShellExecute = false;
ImageMatching.StartInfo.Arguments = " -i " + pathCameraInitfile + " -f " ...
...+ outputFolderFeatureExtraction + " -t " + pathTreeFile + " -o " + outputFileImageMatching;
            ImageMatching.Start();
            ImageMatching.WaitForExit();
            #endregion

            #region  FeatureMatching
string outputFolderFeatureMatching = @"D:\Research\solidworks-api-develop\Test\FeatureMatching";

            Process FeatureMatching = new Process();
FeatureMatching.StartInfo.FileName = @"D:\Research\Software\Meshroom\GitHub...
...\Meshroom-2018.1.0-win64\Meshroom-2018.1.0\aliceVision\bin\aliceVision_FeatureMatching.exe";
            FeatureMatching.StartInfo.CreateNoWindow = false;
            FeatureMatching.StartInfo.RedirectStandardInput = true;
            FeatureMatching.StartInfo.RedirectStandardOutput = true;
            FeatureMatching.StartInfo.UseShellExecute = false
FeatureMatching.StartInfo.Arguments = " -i " + pathCameraInitfile + " -f " ..
..+ outputFolderFeatureExtraction + " -l " + outputFileImageMatching + " -o " + outputFolderFeatureMatching;
            FeatureMatching.Start();
            FeatureMatching.WaitForExit();
            #endregion

            #region  StructureFromMotion
string outputFileStructureFromMotion = @"D:\Research\solidworks-api-develop\Test...
...\StructureFromMotion\cameras.abc";

            Process StructureFromMotion = new Process();
StructureFromMotion.StartInfo.FileName = @"D:\Research\Software\Meshroom\GitHub\Meshroom-2018.1.0-win64...
...\Meshroom-2018.1.0\aliceVision\bin\aliceVision_incrementalSfM.exe";
            StructureFromMotion.StartInfo.CreateNoWindow = false;
            StructureFromMotion.StartInfo.RedirectStandardInput = true;
            StructureFromMotion.StartInfo.RedirectStandardOutput = true;
            StructureFromMotion.StartInfo.UseShellExecute = false;
StructureFromMotion.StartInfo.Arguments = " -i " + pathCameraInitfile + " -f "
..+ outputFolderFeatureExtraction + " -m " + outputFolderFeatureMatching + " -o " + outputFileStructureFromMotion
            StructureFromMotion.Start();
            StructureFromMotion.WaitForExit();
            #endregion

            #region  PrepareDenseScene
 string outputFolderPrepareDenseScene = @"D:\Research\solidworks-api-develop\Test\PrepareDenseScene";

            Process PrepareDenseScene = new Process();
PrepareDenseScene.StartInfo.FileName = @"D:\Research\Software\Meshroom\GitHub...
...\Meshroom-2018.1.0-win64\Meshroom-2018.1.0\aliceVision\bin\aliceVision_prepareDenseScene.exe";
            PrepareDenseScene.StartInfo.CreateNoWindow = false;
            PrepareDenseScene.StartInfo.RedirectStandardInput = true;
            PrepareDenseScene.StartInfo.RedirectStandardOutput = true;
            PrepareDenseScene.StartInfo.UseShellExecute = false;
PrepareDenseScene.StartInfo.Arguments = " -i " + outputFileStructureFromMotion +...
                ..." -o " + outputFolderPrepareDenseScene;
            PrepareDenseScene.Start();
            PrepareDenseScene.WaitForExit();
            #endregion

```

Fig. A.7. Photogrammetry Pipeline code 3 of 5



```
            #endregion

            #region  CameraConnection
string configurationFileCameraConnection = @"D:\Research\solidworks-api-develop...
...\Test\PrepareDenseScene\mvs.ini";

            Process CameraConnection = new Process();
CameraConnection.StartInfo.FileName = @"D:\Research\Software\Meshroom\GitHub\Meshroom-2018.1.0-win64\..
...Meshroom-2018.1.0\aliceVision\bin\aliceVision_cameraConnection.exe";
            CameraConnection.StartInfo.CreateNoWindow = false;
            CameraConnection.StartInfo.RedirectStandardInput = true;
            CameraConnection.StartInfo.RedirectStandardOutput = true;
            CameraConnection.StartInfo.UseShellExecute = false;
            CameraConnection.StartInfo.Arguments = " --ini " + configurationFileCameraConnection;
            CameraConnection.Start();
            CameraConnection.WaitForExit();
            #endregion

            #region  DepthMap
            string outputFolderDepthMap = @"D:\Research\solidworks-api-develop\Test\DepthMap";

            Process DepthMap = new Process();
DepthMap.StartInfo.FileName = @"D:\Research\Software\Meshroom\GitHub\Meshroom-2018.1.0-win64...
...\Meshroom-2018.1.0\aliceVision\bin\aliceVision_depthMapEstimation.exe";
            DepthMap.StartInfo.CreateNoWindow = false;
            DepthMap.StartInfo.RedirectStandardInput = true;
            DepthMap.StartInfo.RedirectStandardOutput = true;
            DepthMap.StartInfo.UseShellExecute = false;
DepthMap.StartInfo.Arguments = " --ini " + configurationFileCameraConnection ...
                ...+ " -o " + outputFolderDepthMap ;
            DepthMap.Start();
            DepthMap.WaitForExit();
            #endregion

            #region  DepthMapFilter
string outputFolderDepthMapFilter = @"D:\Research\solidworks-api-develop\Test\DepthMapFilter";

            Process DepthMapFilter = new Process();
DepthMapFilter.StartInfo.FileName = @"D:\Research\Software\Meshroom\GitHub\Meshroom-2018.1.0-win64...
...\Meshroom-2018.1.0\aliceVision\bin\aliceVision_depthMapFiltering.exe";
            DepthMapFilter.StartInfo.CreateNoWindow = false;
            DepthMapFilter.StartInfo.RedirectStandardInput = true;
            DepthMapFilter.StartInfo.RedirectStandardOutput = true;
            DepthMapFilter.StartInfo.UseShellExecute = false;
DepthMapFilter.StartInfo.Arguments = " --ini " + configurationFileCameraConnection + " --depthMapFolder
                ...+ outputFolderDepthMap + " -o " + outputFolderDepthMapFilter;
            DepthMapFilter.Start();
            DepthMapFilter.WaitForExit();
            #endregion

            #region  Meshing
string outputFileMeshing = @"D:\Research\solidworks-api-develop\Test\Meshing\mesh.obj";

            Process Meshing = new Process();
Meshing.StartInfo.FileName = @"D:\Research\Software\Meshroom\GitHub\Meshroom-2018.1.0-win64...
...\Meshroom-2018.1.0\aliceVision\bin\aliceVision_meshing.exe";
            Meshing.StartInfo.CreateNoWindow = false;
```

Fig. A.8. Photogrammetry Pipeline code 4 of 5



TaskpaneHostUI.cs* | TaskpaneHostUI.cs [Design] | TaskpaneIntegration.cs

Solidworks_Meshroom_NP | MicMac_SW_Blankaddin.TaskpaneHo

```
string outputFileMeshing = @"D:\Research\solidworks-api-develop\Test\Meshing\mesh.obj";

                Process Meshing = new Process();
Meshing.StartInfo.FileName = @"D:\Research\Software\Meshroom\GitHub\Meshroom-2018.1.0-win64...
...\Meshroom-2018.1.0\aliceVision\bin\aliceVision_meshing.exe";
                Meshing.StartInfo.CreateNoWindow = false;
                Meshing.StartInfo.RedirectStandardInput = true;
                Meshing.StartInfo.RedirectStandardOutput = true;
                Meshing.StartInfo.UseShellExecute = false;
Meshing.StartInfo.Arguments = " --ini " + configurationFileCameraConnection + " --depthMapFolder "...
...+ outputFolderDepthMap + "--depthMapFilterFolder" + outputFolderDepthMapFilter + "-o" + outputFileMeshing
                Meshing.Start();
                Meshing.WaitForExit();
                #endregion

                #region  MeshFiltering
string outputFileMeshFiltering = @"D:\Research\solidworks-api-develop\Test\MeshFiltering\mesh.obj";

                Process MeshFiltering = new Process();
MeshFiltering.StartInfo.FileName = @"D:\Research\Software\Meshroom\GitHub\Meshroom-2018.1.0-win64...
...\Meshroom-2018.1.0\aliceVision\bin\aliceVision_MeshFiltering.exe";
                MeshFiltering.StartInfo.CreateNoWindow = false;
                MeshFiltering.StartInfo.RedirectStandardInput = true;
                MeshFiltering.StartInfo.RedirectStandardOutput = true;
                MeshFiltering.StartInfo.UseShellExecute = false;
MeshFiltering.StartInfo.Arguments = " -i " + outputFileMeshing + " -o " + outputFileMeshFiltering;
                MeshFiltering.Start();
                MeshFiltering.WaitForExit();
                MessageBox.Show("3D model created!");
                #endregion

            }
        }
}
```

Fig. A.9. `Photogrammetry Pipeline code 5 of 5`