Master's thesis

2023

Md ashiqul alam khan

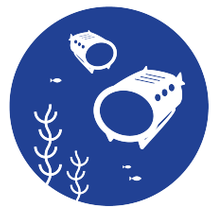
NTNU
Norwegian University of
Science and Technology
Faculty of Engineering
Department of Marine Technology

Md ashiqul alam khan

Underwater Machine Vision for Long Term Operation of Robotic Platforms

June 2023

MIR MARINE & MARITIME INTELLIGENT ROBOTICS



NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Underwater Machine Vision for Long Term Operation of Robotic Platforms

Md ashiqul alam khan

Erasmus Mundus Joint Masters Degree in Marine and Maritime Intelligent Robotics
Submission date: June 2023
Supervisor: Asgeir Johan Sørensen
Co-supervisor: Oscar Pizarro

Norwegian University of Science and Technology
Department of Marine Technology

ABSTRACT

This master thesis focuses on improving underwater geo-localization by identifying the most effective method of data pre-processing and point cloud registration. We explore multiyear 3D point cloud dataset obtained through Structure from Motion (SfM) and compare different registration techniques. Our study reveals that combining global registration method, RANSAC with local point-to-plane ICP registration yields promising results. However, by dividing the point cloud into smaller regions and applying per region RANSAC+ICP registration, we achieve even higher number of image feature correspondence with greater quality. The selected registration method is integrated into our underwater geo-localization dataset framework, demonstrating accurate correspondences in dynamic underwater environments. This research contributes to the understanding of point cloud registration for improved underwater geo-localization, highlighting the potential region based RANSAC+ICP registration in enhancing accuracy and reliability.

SAMMENDRAG

Sammendrag Denne masteroppgaven fokuserer på å forbedre geolokalisering under vann ved å identifisere den mest effektive metoden for forbehandling av data og registrering av punktskyer. Vi undersøker et flerårig 3D-punktsky-datasett innhentet gjennom Structure from Motion (SfM) og sammenligner ulike registreringsteknikker. Studien viser at kombinasjonen av den globale registreringsmetoden RANSAC og lokal ICP-registrering fra punkt til plan gir lovende resultater. Ved å dele punktskyen inn i mindre regioner og bruke RANSAC+ICP-registrering per region, oppnår vi imidlertid enda høyere antall bildefunksjonskorrespondanser med bedre kvalitet. Den valgte registreringsmetoden er integrert i vårt rammeverk for geolokaliseringsdatasett under vann, og viser nøyaktige korrespondanser i dynamiske undervannsmiljøer. Denne forskningen bidrar til forståelsen av punktskyregistrering for forbedret geolokalisering under vann, og fremhever den potensielle regionbaserte RANSAC+ICP-registreringen for å øke nøyaktigheten og påliteligheten.

PREFACE

Undertaking this thesis has been an incredible journey of exploration, innovation, and learning in underwater geo-localization. The research aimed to address challenges posed by significant temporal changes in dynamic underwater environments.

This thesis concludes my Erasmus Mundus Joint Masters degree in Marine and maritime Intelligent Robotics at the Norwegian University of Science and Technology, (NTNU), Norway and Universite de Toulon(UTLN), France. It has been carried out during the spring of 2023 and has been supervised by Asgeir Johan Sørensen and Oscar Pizarro from Department of Marine Engineering.

The work presented in this thesis is a continuation of the work performed in the specialization project during the autumn of 2022.

The sections of the thesis contains **Chapter 1 Introduction:** problem statement, literature review and brief overview of the whole thesis, **Chapter 2 Theory:** Theoretical background for the proposed methods. **Chapter 3 Methods:** Methodology of the thesis in details, **Chapter 4 Result:** Comparison and visualizing the result and findings of the thesis, and finally **Chapter 5:** Detailing what we achieved through this thesis and what are the future works that can be done.

I would like to thank my supervisor Asgeir Johan Sørensen for all guidance and good advice throughout the process. I would specially like to thank my co-supervisor Oscar Pizarro for answering all my questions, giving me the necessary insight and knowledge, and for good discussions on the development of the system. Finally, I would like to thank my family and friends for their endless support.

Contents

List of Tables

List of Figures

ABBREVIATIONS

List of all abbreviations in alphabetic order:

- **NTNU** Norwegian University of Science and Technology.
- **GPS** Global Positioning System.
- **AUV** Autonomous Underwater Vehicle.
- **USBL** Ultra Short Baseline.
- **SBL** Short Baseline.
- **LBL** Long Baseline.
- **SLAM** Simultaneous Localization and Mapping.
- **SfM** Structure From Motion.
- **SIFT** Scale Invariant Feature Transform.
- **MSE** Mean Squared Error.
- **RANSAC** Random Sample Consensus.
- **NCC** Neural Code Comprehension.
- **ONN** Orientation Normalization Network.
- **APR** Absolute Pose Regression.
- **RGB-D** Red Green Blue - Depth.
- **MVS** Multi View Stereo Matching.
- **ICP** Iterative Closet point.
- **RL** Reinforcement Learning.
- **ReLU** Rectified Linear Unit.
- **GAN** Generative Adversarial Network.
- **SGD** Stochastic Gradient Descent.

- **BBA** Bundle Block Adjustment.
- **DEM** Digital Elevation Models.
- **EO/IO** Exterior Orientation/Interior Orientation.
- **KNN** K-nearest neighbors.
- **FPFH** Fast Point Feature Histogram.
- **ICP** Iterative Closest Point.
- **IRLS** Iteratively Reweighted Least-Squares.
- **ORB** Oriented FAST and Rotated BRIEF.
- **FAST** Features from Accelerated Segment Test.
- **BRIEF** Binary Robust Independent Elementary Features

CHAPTER

ONE

INTRODUCTION

Ocean covers almost 70% of Earth's surface. The oceanic system has an immense effect on the on going climate change problem. It produces at least 50% of the planet's oxygen while absorbing 30% of human produced carbon dioxide. It is also an inhabitant of around 240,000 species with great diversity. In addition, the ocean serves as the largest heatsink on the planet, absorbing a staggering 90% of the heat generated by climate change.

Economically speaking, according to a report by the Organization for Economic Co-operation and Development (OECD) around 40 million people (about twice the population of New York) will be directly employed by ocean-based industries by 2030. These sectors include, among others, shipping, fishing, tourism, and offshore oil and gas extraction businesses. In addition, they also have a considerable amount of indirect impact on other economic sectors particularly in the supply chain of different industries. A sustainable development of such coastal areas will also benefit economic growth and improve the quality of life of the nearby population. Moreover, it is of utmost importance to prevent man-made disasters caused by system failures, such as oil rig malfunctions and explosions, which can result in irreparable damage to the ocean environment.

Still with all these possible benefits, around 90% of the big-fish population along with 50% coral reef has already undergone massive destruction. What is left is endangered by frequent human intervention and rapid climate change. Thus, it has become a topic of immense importance. And we must bring together scientific and technological means towards revitalizing the ocean environment and restoring natural balance.

Our project will provide scientific means to tackle some parts of this global problem. Through providing method for data creation and pre-processing for temporal change detection in coral reef to better understand and study climate change effect on ocean space. In addition, it will also provide tools for building more sustainable underwater structures (Oil rig, Gas pipeline, windmills, electricity transportation, underwater waste deposition etc.) through continuous monitoring of system failure and degradation over time. Such sustainable and safe structures will prevent any future environmental disaster caused by unprecedented natural or artificial

calamities. Our project will also influence how we localize autonomous systems underwater without the availability of GPS (Global Positioning System) systems. This will provide underwater vehicles with better collision avoidance capabilities and most importantly greater navigational accuracy for long term missions.

1.1 Motivation

Motivation for this project can be divided in two challenges:

- Data Collection and Pre-processing: Geo-localizing areas with significant temporal changes requires robust data collection and pre-processing methods. We aim to improve the detection of reliable features in dynamic environments by focusing on feature based method for dataset curation.
- Robust Correspondence between Images: Establishing accurate correspondences between images captured at different times in dynamic environments is essential for precise geo-localization. Our goal is to develop methods that can reliably establish correspondences using 3D point cloud information despite some part of the image having temporal changes.

Both of these solutions will have tremendous effect in understanding the dynamic nature of underwater habitat, causes and effect of environmental pollution, offshore structure monitoring, aqua culture growth, navigation and seabed mapping.
Temporal change detection underwater is quite different compared to traditional geo localization approaches in remote sensing domain. The dynamic characteristics of ocean, in addition to lower light performance, lack of standard communication system (radio-wave, GPS) coupled with transitional change in water salinity, temperature, phytoplankton presence provides significant complications in creating a robust underwater vision system. In addition, change of seafloor due to ocean current, sediment deposition, seasonal changes will introduce more difficulty in revisiting the same area after a certain period. Most importantly, our project will focus on processing dataset and to create robust correspondence for detecting temporal changes in benthic habitats underwater. The change in such habitat is directly influenced by water quality, seasonal influence, and climate change. And such temporal change detection requires accurate geo localization to compare with previous surveys.

Benthic habitat observation is generally conducted at a height of 3-5 meters from the ground. One significant challenge in this context is the potential impact of parallax errors on the geolocalization model, even with relatively small motions. Parallax errors arise due to the close proximity of the observation height to the ground. As the camera moves or tilts slightly, the resulting parallax effect can significantly affect the accuracy of the geolocalization model. This issue is particularly relevant in benthic habitat observation scenarios, where precise and reliable geolocalization is essential for studying and monitoring underwater ecosystems.
In addition, for navigation and geo-localization, it is particularly challenging in underwater domain due to limited GPS accessibility for depth beyond 20 centimeters (about 7.87 in). In addition, radio signal attenuates at a much higher rate

underwater. On the other hand, acoustics sensors also suffer from many shortcomings. For example: small bandwidth, low data rate, high latency, variable salinity and water temperature and data loss. Which hinders the real time operation of underwater vehicles and positioning of dynamic underwater structures.

Other frequent approaches such as USBL (Ultra Short Baseline), SBL (Short Baseline), and LBL (Long Baseline) systems have their own drawbacks. The main problem of these systems is they require a pre-deployed and localized infrastructure. Which limits the operational area and involves larger operational cost. In addition, communication speed between AUV (Autonomous Underwater Vehicle) and localized structure can suffer from acoustic sensor complications as mentioned before. Dead Reckoning/Inertia based localization approach uses accelerometers and gyroscopes for vehicle orientation and motion estimation. These systems produce cumulative error resulting unbounded errors over time.

Another popular solution is to use SLAM (Simultaneous Localization and Mapping) systems. These systems are very much capable of providing good estimates of the map and vehicle positioning if The onboard sensors are perfectly accurate. Which is not true in real life, sensors are never accurate enough and the underwater environment is very dynamic. Different versions of SLAM are available for underwater domain but with their own limitations. Real performance of a slam system mostly depends on accurate loop closure. A single wrong loop closure will catastrophically affect the localization and mapping performance. To accurately use loop closure, SLAM system must collect robust features and do accurate feature matching.

1.2 Literature Review

Underwater Geo-localization is a challenging task that has garnered significant research attention. While most geo-localization research has focused on aerial images, there is a need to adapt and apply similar techniques to underwater environments. In this literature review, we will explore influential research works in this domain and identify their limitations and dependencies.

One notable research paper, A seasonally invariant deep transform for visual terrain-relative navigation by Fragoso et al. [1], proposes a deep learning model that generates deep features from temporal image pairs. The authors train two separate U-net models using datasets that only differ in their time stamps. They evaluate the model using two loss functions: Area-based registration (NCC) [2] and Feature-based registration (SIFT) [3]. The paper highlights that more training data improves the network's performance, even when from different datasets. However, limitations include the reliance on NCC for SIFT matching, orientation problems, and the detection of small features and unreliable deep features in heavily shaded areas.

In a similar vein, the paper "Robust Multi-Temporal Underwater Mosaic Matching

and Registration based on Deep Siamese Neural Network" introduces a Siamese neural network for matching and registration [4]. The authors train the network using positive and negative datasets, employing RANSAC for image registration. Challenges addressed include orientation changes and obtaining accurate ground truth from different timelines. However, significant changes can impact the quality and quantity of SIFT features, and establishing robust underwater SIFT feature matching remains a challenge.

To address the general orientation problem in geo-localization, Tian et al. [5] develop an Orientation Normalization Network (ONN) that aligns query and reference imagery to a canonical orientation. This allows deep local features to depend on patch overlap rather than alignment alone. The authors use a convolutional neural network for deriving deep local features and employ a spatial transformer for reorientation. Limitations include unwanted global features, ambiguity in spatial transformer due to varying overlap, and challenges posed by homogenous areas and high temporal differences between query and reference images.

A different approach to geo-localization is presented in the paper "Are These from the Same Place? Seeing the Unseen in Cross-View Image Geo-Localization" by Rodrigues et al. [6]. The authors propose a semantically driven data augmentation technique for cross-view image geo-localization. Their model employs a multiscale attentive embedding network and contrastive loss for training. While the approach achieves promising results, it is not suitable for the underwater domain due to the lack of ground view and the difficulty of creating class masks.

Sattler et al. [7] discuss the limitations of CNN-based camera pose regression. They find that CNN-based methods do not perform as well as structure-based methods, even with substantial training data. The paper emphasizes that structure-based localization remains the current standard and highlights the limitations of camera-based absolute pose regression.

In summary, to address the limitations of CNN-based approaches and improve feature detection, it is crucial to combine 2D and 3D information. This involves creating a dataset that establishes robust SIFT-based matched features based on descriptor distance and filters false matches using their 3D properties. By focusing on detecting only robust features that satisfy structural constraints, we aim to reduce false matching to zero. the reviewed literature highlights the challenges and limitations of underwater geo-localization. By learning from existing research, we can adapt data processing techniques, such as deep learning models and feature matching algorithms, to overcome these challenges.

1.3 Project description

Our project is divided into two parts:

- Exploring Underwater Data: Develop effective techniques to process and prepare the data for analysis.

- Robust feature matching between image pairs with significant temporal changes.

1.3.1 Exploring Underwater Data: Develop effective techniques to process and prepare the data for analysis.

In the Data Collection section, our objective is to obtain a suitable dataset for our problem. To leverage the previous data and their 3D characteristics, we require 3D point clouds of the same area from multiple years. We utilized Structure from Motion (SfM) to generate the point clouds and reconstruct the scene, along with the respective camera poses. For this purpose, we obtained a multiyear 3D dataset from the University of Sydney, which encompasses all the necessary element to take advantage of 3D information for underwater geo-localization. But the 3D dataset gathered across multiple year suffers from error based on GPS georeferencing. Thus it is very important to register the dataset over a time period so that we have an accurate alignment of identical structures.

The subsequent step involves pre-processing the dataset. Although we possess multiyear data, it is crucial to determine if they originated from the same area in subsequent years. To address this, we employed geo-reference points. Despite using geo-reference points, there may still be errors in the calculation of the point clouds for each year. Thus, we employed point cloud registration techniques to align the position of the point clouds and camera poses to a common coordinate system. By doing so, we can validate their true positions based on a common global coordinate system.

We need positive image pair and negative image pair for training the deep learning models to learn robust features. For such reason, the next task is to create positive and negative pairings of images across multiple years.

It is also essential to understand the coordinate systems used in each point cloud. Notably, the point cloud and camera coordinate systems may differ in Metashape, the software used for generating the point clouds. To address this, we meticulously tracked all the coordinated changes and converted the positions of the point clouds and camera poses to a single global coordinate system across the years. In our case, we adopted the world coordinate system from dataset of 2014 as the global coordinate system for all datasets. The final dataset utilized in this study consists of positive image pairs, accompanied by their respective camera poses and 2D correspondences, all based on reprojection errors. The training dataset includes image pairs comprising one image from 2014 and its corresponding image from 2015. Conversely, the test dataset comprises image pairs from 2014 and 2016, allowing for a comprehensive evaluation of the proposed method across multiple years.

1.3.2 Robust feature matching between image pairs with significant temporal changes

Robust correspondence points are crucial in establishing links between images captured from different viewpoints. These points represent the same physical location in the scene but appear differently in each image. To identify these correspondences, we employ SIFT feature matching, selecting a total of 80,000 points from multiple octave layers. This comprehensive selection ensures that no correspondences are missed due to Gaussian noise response. We calculate SIFT descriptors and perform Brute force matching with a variable threshold (ranging from 0.74 to 0.9) to account for varying match quantities. Outliers are removed, and only robust matches based on corresponding 3D points are considered.

To obtain the 3D points of the matched 2D points, we follow a series of steps. First, we transform camera matrices from the local coordinate system to the global coordinate system. Next, using the camera matrices for each image, we determine the 3D points of each 2D point. we further transform the 3D points using the transformation matrices obtained from point cloud registrations for the years 2015 and 2016. To ensure the robustness of our 3D points, we compare corresponding 3D points from image one to image two in the positive image pairs.

By carefully selecting this subset of points with minimal differences, we ensure that our dataset comprises highly reliable and accurate 2D correspondences based on their 3D characteristics. These robust points, immune to temporal changes, are essential for training deep learning models for geo-localization.

1.4 Real world Applications

The potential for real world implementation for this work is immense. This section will only highlight some the most important aspects:

1.4.1 Environmental Survey

Climate change has an enormous impact on the ocean life cycle. The change in climate also changes how varied species live underwater. As most of the earth is covered with ocean, it is particularly important to understand the temporal behavior of our ocean with respect to climate change. Our proposed work will provide great assistance in understanding such oceanic changes over time, particularly how seabed habitat is evolving through time.

1.4.2 Seabed Mapping

The ability to determine what and where is safe is made possible by seabed mapping, which is a vital tool for searching underwater resources, controlling extraction, and laying down equipment. Seafloor maps also allow ships to safely navigate around both naturally occurring and man-made structures on the ocean floor. Such seabed mapping with accurate temporal information provided by our work

will result in better understanding of where and how to build man-made structures (port, aqua culture, oil rig, gas rig, wind turbine etc.). The design can be made more robust to temporal changes to seabed with this additional information, which will reduce risk and increase the sustainability of those systems.

1.4.3 Structure Monitoring

Preventive maintenance is very crucial for underwater structures. It helps to prevent unwanted disasters. It includes visual evidence of impending failure such as water seepage, cracks, surface degradation. In addition, underwater electric cables and optical fiber cables monitoring is also important for safe operation of offshore structures. Most importantly, underwater gas and oil pipeline, oil drilling structure maintenance must be precise and accurate to safely guard against any substantial environmental disaster. Undoubtedly, robust temporal change detection will be decisive in avoiding such catastrophe. Temporal monitoring can also validate maintenance work done on the structure by comparing it with the initial structural condition.

1.4.4 Benthic habitat study

Main motivation of our work is to understand how benthic habitat changes with time. This understanding of changes will help researcher to develop technologies in restoring natural balance. On top of that, temporal change detection can also validate the effectiveness of these technologies through continuous monitoring of concerned areas affected by those technologies. Benthic monitoring can provide early warning indicators of decreasing fish stocks. Additionally, it can help to ensure that fish will always be available as a food and income source.

1.4.5 Pollution Detection

The quantity of marine litter in the oceans is gigantic. The amount of research done to evaluate and address the environmental impact of marine litter has increased tremendously in recent years. But much of this research is concentrated on floating or stranded litter. Whereas there is little knowledge on marine litter on the seafloor. Through temporal monitoring of seabed near shore, we can identify which part of the country is contributing more to ocean pollution. It will immensely help the government to implement necessary restrictions and develop innovative technologies in those areas to tackle the growing problem.

1.4.6 Path following

If we can geo-localize our system accurately, then we can also follow predefined path traversed before easily. Even with temporal changes, we will not require new positioning modality to follow the same path before. Through matching robust descriptors between image sets of different timelines we can accurately follow the same motion pattern or generate new motion trajectories based on our needs.

CHAPTER

TWO

THEORY

2.1 Theory

3D computer vision deals with objects described in 3D space. In this section, we will talk about topics of 3D computer vision directly relevant to this project. Many of the fundamental concepts will not be discussed here. We will not go in to details in this theory section but will highlight some basic concepts on what we are working with.

2.1.1 3D Geometry

Three-dimensional geometry facilitates the depiction of a line or plane in a three-dimension using the x, y, and z axes. The x, y, and z axes, which are all perpendicular to one another and share the same units of length on all three axes resulting in a three-dimensional cartesian coordinate system. In three-dimensional geometry, every point's coordinates have three coordinates (x, y, z). These three coordinates represent a point in 3D space. We can easily construct a 3D space if we have x, y, z coordinates of all the elements of the scene. This is called point cloud. Using RGB-D images we can easily create such point clouds. The resultant point cloud will be more sparse structure compared to grid aligned RGB-D images. If we want to fit the point cloud in a 3D grid, we can simply use voxel of our preferred size and associate each voxel with group of points belonging to that voxels area. Voxels are 3D equivalent of pixels. Voxel-based methods perform well in deep-learning tasks. But they do lose a certain amount of information during the point cloud to voxel transformation process.

2.1.2 Point Cloud from 2D images

There are multiple ways to capture 3D point cloud of an object. For example, Active stereo vision, Laser triangulation, Structured light and Time-of-Flight and mostly used. For this project we are concerned with image-based point cloud creation, particularly stereo vision. The stereo vision method records two 2D images of the target item from two distinct viewpoints using two cameras. With known camera relative pose and camera intrinsic, we can identify the disparity between two same points residing at different pixel location for each camera's

2D image plane through triangulation method. Which will give us the x, y, and z coordinate of that pixel. it is only possible to use triangulation if we have same point present in those image pairs. Thus, correct point correspondence between images has a significant effect on point cloud accuracy. In addition, smooth featureless surface provides little means to establish point correspondence between image pairs. That causes difficulty in creating point clouds for that featureless area which results in a sparsely populated point cloud.

2.1.3 Structure from Motion (SfM)

SfM is a popular method used to create a 3D point cloud from multiple 2D images of different camera poses. The main principle is the same as stereo vision. In this case the corresponding images in sequence act as stereo pairs. When we have the matching locations of multiple points on two or more photos, there is usually just one mathematical solution for where the photos were taken. Therefore, a singular step known as “bundle adjustment" can be used to calculate individual camera poses (positions and orientations), camera intrinsic (focal lengths, principal point etc.), relative position corresponding features. The final optimization produces a near accurate structure of the scene. Taking SfM points as ground control along with known camera parameters, we can produce a dense model. All pixels of all the images can be used to increase the resolution of the structure. This process is also known as multi view stereo matching (MVS). A final georectification is done to convert the local coordinate system of the point cloud into a georeferenced coordinate system either using ground control points (known geographical coordinates) or with actual camera position and focal lengths. It has some important advantages over stereo image methods. SfM can be used with images of different ranges of distance and angles without any prior knowledge of locations or orientations. In addition, it does not require a stable platform with fixed elevation (reduced operational cost). To increase the accuracy of point cloud, it is important to have multiple views of same area from different camera poses and significant overlap between image pairs for greater point correspondence.

2.1.4 Point cloud registration

Point cloud registration is a process of transforming one point cloud to correctly align itself with another point cloud. For our case, we can simply say that we are looking for an optimal homogenous transformation matrix that can convert coordinates of all the points of our source point cloud to be similar with the coordinates of the target point cloud. Traditional methods for registration are based on minimizing geometric projection error, which is achieved by correspondence searching and transformation estimation. Currently, there are multiple Deep Learning based solutions. For our project we are more interested in traditional methods compared to deep learning-based methods. The reason for this will be discussed in the report’s result section. Instead of using RANSAC by Fischler and Bolles [4] based global registration, fast global registration by Zhou, Park, and Koltun [8] is implemented to get an initial transformation matrix estimate with much lower computational cost. Now using this initial estimate, we can leverage on the ICP (Iterative Closest Point)Rodrigues and Tani [6] based local registration

to get an extremely high registration accuracy. We choose Point-to-Plane byChen and Medioni [9] ICP which uses the following objective function:

$$E(T) = \sum_{(p,q)\in K} ((p - Tq) \cdot n_p)^2 \tag{2.1}$$

Where n_p is the normal point of p.
In addition, point-to-plane ICP has faster convergence speed than point-to-point ICP. To remove outliers in the registration process, we can use different robust kernels. which results in better registration performance as shown in Babin, Giguère, and Pomerleau [10]. The main idea of a robust loss is to lower large residuals resulting from outliers. This is achieved by optimizing $E(T)$ as:

$$E(T) = \sum_{(p,q)\in K} ((p - Tq) \cdot n_p) = \sum_{i=1}^{N} \rho(r_i(T)) \tag{2.2}$$

Where, $\rho(r)$ is the robust loss or robust kernel

2.1.5 Machine Learning and Deep Learning Techniques

Machine learning helps computer to learn from data or experience and make predictions based on that learning. Machine learning provides an intelligent data driven approach with out hard coding the solution for a particular problem. There are mainly three types of machine learning domain depending how they leverage on provided data.

- Supervised Learning: In supervised learning, Machine Learning algorithm trains it self using labels data. The task is to map each inputs to their corresponding true labels or ground truth. These systems are data hungry. And often its expensive, time consuming to hand label those huge amount of data. Efficiency lies on how well and accurately the data is labeled.

- Unsupervised Learning: Its a opposite notion to supervised learning. No need of labeled data is required here to train the model. They often cluster the data based on maximum similarity . Most of the traditional clustering algorithms are based on unsupervised learning.

- Reinforcement Learning: RL or reinforcement learning learns from experience. Given an environment, it applies all possible actions and get rewards based on put goal. The idea is to maximize the reward. With maximizing the reward, system learns to take best actions in each possible state based on previous experience. Thus the RL system provides optimal state action pair for achieving the ultimate goal.

Self-supervised learning is an emerging domain in machine learning community. Generally. this models are fed with unstructured data, and trained to generate data labels automatically. Using these generated labels as ground truth , the model trains it self.

2.1.5.1 Deep Learning:

Deep Learning is a sub field of Machine Learning inspired from human brain analogy. It is composed of closely connected neurons with variable weights. Which significantly increases the number of parameters. This in general provides better generalization of the objective function. Following 2.1.1 and 2.1.2 illustrates such analogy:

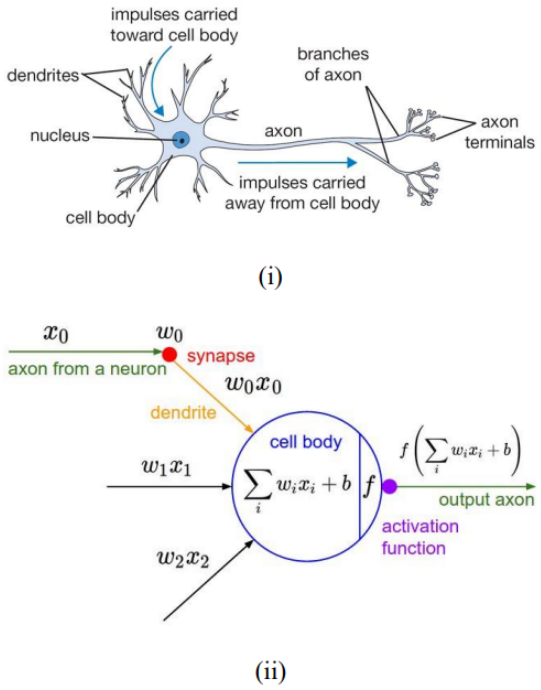


Figure 2.1.1: (i) Human neuron (ii) Mathematical analogy of Neuron

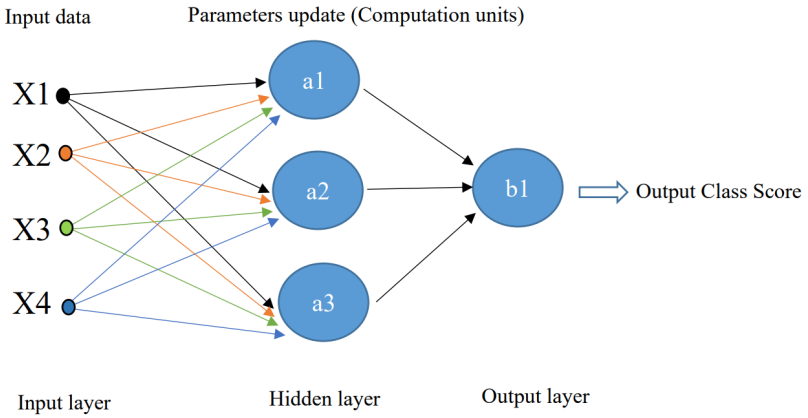


Figure 2.1.2: Simplified Deep Learning model based on brain analogy

Each deep learning model has following sets of basic function to train, update, optimize and test the input data:

- Forward Propagation: Forward propagation is the process of computing and storing intermediate variables (including neural network outputs) for the models in the order from input layer to output layer. Lets assume Z is a linear function at i^{th} computation node. Then

$$Z^i = W^i x + b^i \tag{2.3}$$

  Where W is a randomly initialized weight, x is the input matrix and b is a randomly initialized bias vector. Now, we multiply the linear function with a non liner activation function σ which results to output:

$$a^i = \sigma(Z^i) \tag{2.4}$$

- Loss Function: Neural networks contain a crucial component called the loss function that measures the discrepancy between the predicted value and the ground truth. It is a non-negative variable where the robustness of the model grows as the loss function's value decreases.
  - Cross entropy Loss:
  - Contrastive Loss:
  - Reprojection error based loss:
  - Pose based Loss:
- Back Propagation: Back propagation from the work of Rumelhart, Hinton, and Williams [11] refers to the method of calculating the gradient of neural network parameters,in general, back propagation calculates and stores the intermediate variables of an objective function related to each layer of the neural network and the gradient of the parameters in order of the output layer to the input layer according to the chain rule in calculus
- Parameter Update: Updating procedure of the network now utilizes such gradients achieved in previous step. It updates the weight and bias variable using following analogy:

$$W^n = W^n - lr * dW^n \tag{2.5}$$

$$b^n = b^n - lr * db^n \tag{2.6}$$

  Where lr is the learning rate
- Activation Function: The artificial neural networks' activation functions are a crucial component. In essence, they decide whether or not to activate a neuron. The activation function is the non-linear transformation that we do over the input signal. Some frequently used activation functions:
  - Sigmoid: A sigmoid function is a mathematical curve that looks like the letter "S." as shown in figure 2.1.3 In machine learning, we often use the term "sigmoid function" to talk about the logistic sigmoid function. The main job of a sigmoid function is to take input values and squash

them into a smaller range. This makes it easier to interpret the values as probabilities.

$$S(x) = 1/(1 + e^{-x}) \tag{2.7}$$

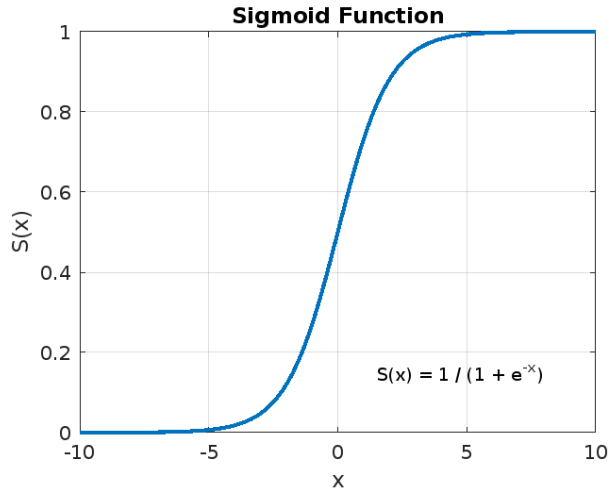


Figure 2.1.3: Sigmoid Function

– Tanh: The hyperbolic tangent function, tanh, shares a similar shape with the sigmoid function. However, unlike sigmoid, tanh maps negative values to even more negative values and brings zero values closer to zero. Equation 2.7 provides the description of a specific type tanh activation function.

$$tanh(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x}) \tag{2.8}$$

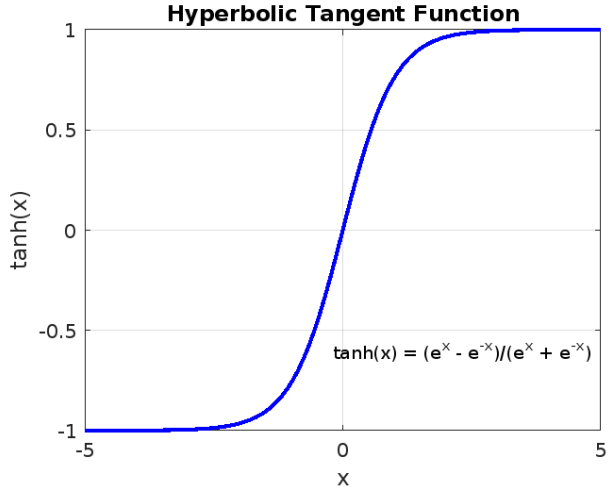


Figure 2.1.4: Tanh Function

– ReLU: The ReLU (Rectified Linear Unit) function is an activation function commonly used in deep learning. It has gained popularity due to its simplicity and effectiveness in neural networks.It returns the input

value x if it is greater than or equal to zero, and returns zero otherwise. The ReLU function introduces non-linearity to the network, which is essential for learning complex patterns and making the network more expressive. 2.1.5 function introduces non-linearity to the network, allowing it to model and learn complex relationships between inputs and outputs. Linearity alone in the network's activation functions would result in a linear combination of linear functions, limiting its capacity to represent more intricate patterns.

$$ReLU(x) = max(0, x) \tag{2.9}$$

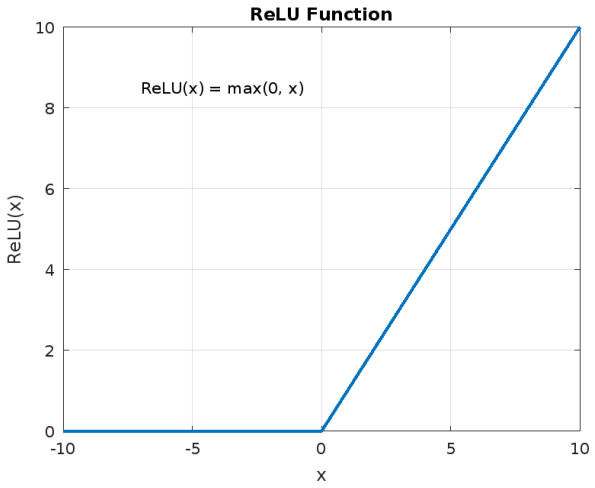


Figure 2.1.5: ReLU Function

– Leaky ReLU: The Leaky ReLU (Rectified Linear Unit) function is a variations of the ReLU function that addresses one of its limitations, which is the "dying ReLU" problem. In comparison to the ReLU function, the LeakyReLU function allows a small gradient for negative inputs, preventing the neurons from becoming completely inactive. This helps to alleviate the "dying ReLU" problem and promotes better learning in deep neural networks.

$$LeakyReLU(x) = max(x, alpha * x) \tag{2.10}$$

where alpha = 0.1

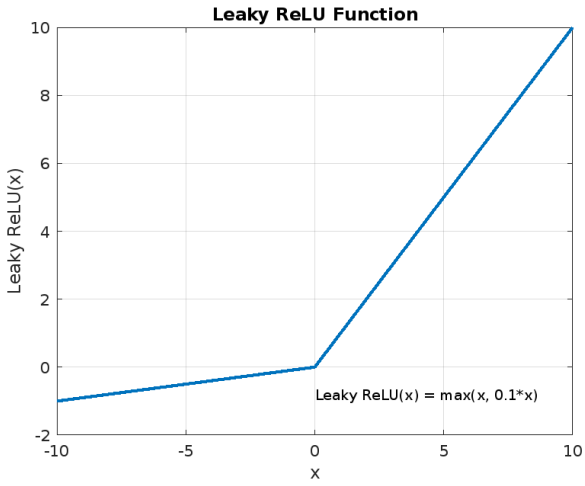


Figure 2.1.6: Leaky ReLU Function

Some common properties of these activation functions:

- Sigmoid functions and their combinations generally work better in the case of binary classifiers
- Sigmoids and tanh functions are sometimes avoided due to the vanishing gradient problem
- ReLU function is a general activation function and is used in most cases these days
- If we encounter a case of dead neurons in our networks the leaky ReLU function is the best choice
- Always keep in mind that ReLU function should only be used in the hidden layers
- As a rule of thumb, you can begin with using ReLU function and then move over to other activation functions in case ReLU doesn't provide with optimum results

- Optimization Algorithms: Optimization algorithms helps us to minimize (or maximize) an Objective function (another name for Error function), which is simply a mathematical function dependent on the Model's internal learnable parameters which are used in computing the target values from the set of predictors used in the model. Some of the most used optimizer are listed below:

  - Gradient Descent : It is an iterative first-order optimisation algorithm used to find a local minimum/maximum of a given function. This method is commonly used in machine learning (ML) and deep learning(DL) to minimise a cost/loss function. Gradient Descent Algorithm iteratively calculates the next point using gradient at the current position, scales it (by a learning rate) and subtracts obtained value from the current position (makes a step). It subtracts the value because we want to minimise the function (to maximise it would be adding). This

process can be written as:

$$\theta := \theta - \alpha * \nabla J(\theta) \tag{2.11}$$

θ represents the parameters of the neural network (weights and biases), α (alpha) is the learning rate, which determines the step size taken in the direction of the gradients. It is a hyperparameter that needs to be set manually. $\nabla J(\theta)$ represents the gradient of the loss function J with respect to the parameters θ. The gradient indicates the direction and magnitude of the change needed to minimize the loss. The smaller learning rate the longer GD converges, or may reach maximum iteration before reaching the optimum point. If learning rate is too big the algorithm may not converge to the optimal point (jump around) or even to diverge completely.

– Adam: Adam is an optimization algorithm that serves as an alternative to the conventional stochastic gradient descent (SGD) procedure for iteratively updating network weights based on training data. Unlike SGD, which uses a fixed learning rate throughout training, Adam incorporates adaptive learning rates and other techniques to enhance the optimization process.

$$m = \beta_1 * m + (1 - \beta_1) * g \tag{2.12}$$

$$v = \beta_2 * v + (1 - \beta_2) * (g^2) \tag{2.13}$$

Where m represents the first moment estimate, which is the exponentially decaying average of past gradients, v represents the second moment estimate, which is the exponentially decaying average of past squared gradients.

$$\hat{m} = \frac{m}{1 - \beta_1^t} \tag{2.14}$$

$$\hat{v} = \frac{v}{1 - \beta_2^t} \tag{2.15}$$

Where $\hat{m}$ and $\hat{v}$ are bias-corrected versions of m and v, β_1 and β_2 are hyperparameters that control the exponential decay rates for the moment estimates ($0 < \beta_1 < 1$ and $0 < \beta_2 < 1$) and t represents the time step or iteration number.

$$w = w - \frac{\eta * \hat{m}}{\sqrt{(\hat{v})} + \epsilon} \tag{2.16}$$

Where w is model weights,η is the step size that determines the magnitude of weight updates. ϵ is a small constant added to the denominator for numerical stability.

– RMS Prop : It is an optimization algorithm commonly used in deep learning for updating network weights during training. It is an extension of the classical gradient descent algorithm that addresses some of its limitations, such as slow convergence and sensitivity to learning rates.

$$v(t) = \rho * v(t-1) + (1 - \rho) * (grad^2) \tag{2.17}$$

Where v(t) is the current value of the first moment estimate (squared gradients). ρ is the decay rate that controls the contribution of previous estimates. Typically, it is set to a value like 0.9. grad is the gradient of the network weights at the current iteration. w(t) is the updated value of the network weights.

$$w(t) = w(t-1) - \frac{\eta}{\sqrt{(v(t) + \epsilon)}} * grad \tag{2.18}$$

η is the learning rate that determines the step size for weight updates. epsilon is a small constant (e.g., 1e-8) added for numerical stability to avoid division by zero.

Some common terms used in Deep Learning:

- Training, Test and Validation set: Division of the data set into training and validation set for training phase, and testing set for testing phase. There is no overlapping of sample in between the sub data sets.
- Batches: Instead of delivering the complete data set as an input at once when training a neural network, we randomly partition the input into many pieces of equal size feed it to the network. Model trained on these batches of data is more generalized compared to full data set.
- Epochs: An epoch is defined as a single training iteration of all batches in both forward and back propagation. This means 1 epoch is a single forward and backward pass of the entire input data.
- Regularization: Method generally used for removing over-fitting. Regularization can be described as change we make to a learning system that is designed to lower its generalization error but not its training error.
- Dropout: Randomly dropping some neurons in the hidden layer to protect against over-fitting. Dropout is one type of regularization technique.
- Batch Normalization:t, batch normalization is used to ensure that distribution of data is the same as the next layer hoped to get.
- Data Augmentation: Process of changing orientation, or color of images to create more data sample or class sample with higher variance for better system generalization ability.
- Vanishing Gradient: When the gradient of the activation function is very small, the vanishing gradient problem occurs. The weights are multiplied with these small gradients during back propagation, thus they have a tendency to get smaller and "vanish" as they move deeper into the network. As a result, the long-term dependency is forgotten by the neural network. Exploding gradient is just opposite of vanishing gradient
- Bias Variance Trade off:Bias is the difference between the average prediction of our model and the correct value which we are trying to predict. Model with high bias oversimplifies the model. It always leads to

high error on training and test data.Variance is the variability of model prediction for a given data point or a value which tells us spread of our data. Model with high variance does not generalize on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.

- Transfer Learning: Utilizing already learned model and train it in a new data set. It can be either feature extraction: only last layers are updated during training. or Fine tuning: all the layers are updated during training on new dataset. Transfer learning lowers training time as its weight are already at a good initialized position.

2.1.5.2 Deep Learning Applications:

- Computer Vision: Deep Learning technique has revolutionized the field of computer vision. With the introduction of convolutional neural network, we achieved significant advancement in the field of computer vision. There a diversified application in this domain. Some of the most important ones will be mentioned here.

  - Feature Extraction: Extracting useful information from an image or set of images or even a video. Useful information can be defined by the end user. Like edge detection, Feature point selection or color detection.
  - Image Classification: Classifying an image. It can be single image call or bounded box multi-class classification in a single image. For example, detection human, car, tree in a photo and locate them in pixel space.
  - Continuous Monitoring and Tracking: The use case is vast. From structural monitoring to security. Offshore structure monitoring, Manufacturing Fault detection, System failure detection, Bird tracking or eye tracking for auto focusing in camera system, security against intruders and anomaly detection.
  - Neural Rendering: Fast Deep learning based graphics rendering is becoming very popular. They are fast and fairly accurate compared to traditional rendering techniques.
  - Generative Adversarial Network(GAN): A generative model's objective is to analyze a set of training examples and discover the probability distribution that produced them. The estimated probability distribution is then used by Generative Adversarial Networks (GANs) from Goodfellow et al. [12] to produce more realistic instances similar to the training example. Deep Fake, Style transfer, Image enhancement are some of the most common implementations of GANs.
  - Image Enhancement: Image resolution up scaling, down scaling, image quality enhancement, noise removal, color restoration are most used deep learning based techniques for image enhancement.
  - Localization and Mapping: Localization of an object or vehicle based on visual surroundings, Mapping on the environment based on overlapping visual features captured by cameras in motion.

  - Semantic Segmentation: Segmenting classified object and its are in the images, detecting fault though segmentation of images.
  - Pose Estimation and Action Recognition: Estimating human pose in sports, or action detection in security cameras or human tracking system.
  - Change Detection: Determining temporal changes in diverse types of objects for preventive monitoring and accident alarm.

- Robotics: The filed of robotics has been heavily influenced by deep learning, particularly by deep reinforcement learning Mnih et al. [13]. Some of the most used cases will be illustrated below:
  - Motion Planing: Planning of vehicle motion based on changes in surroundings. In addition industrial robot motion for doing particular dynamic job.
  - Path Planing: Dynamic environment path planning based on continuous obstacle detection collision avoidance.
  - Localization and Mapping: Mapping the environment and localize it self in the environment by using multi modal(IMU, LASER, IMAGE) SLAM.
  - Job task optimization: Optimal state-action selection for achieving a particular goal for each jobs.

- Data Analytic: The world is full of data. Intelligent data understanding and utilizing will significantly improve human lifestyle and also technological advancement and increased safety.
  - Market Analysis and Stock exchange prediction: Trained on market and stock exchange data, Deep learning can predict future outcome.
  - Weather prediction: Weather prediction model can be greatly improved by deep learning based prediction
  - Biotechnology: Genome understanding and development of new bio material can be done through deep learning based analysis.
  - System monitoring: Predicting failure based on system and/or sensor data.

- Natural Language Processing:
  - Chatbot: Developing chatbot for intelligent machine conversation. for example chatGPT
  - Summarizing: Summarizing a topic or book through AI.
  - Language Translation: Translating languages based on Deep Learning models. For example, English to Norwegian.
  - Speech Recognition: Understanding speech signature for bio metric identification. Speech to word transformation
  - Sentiment Analysis: Predicting human sentiment through words or speech analysis.

- Text Generation: AI generated text based on human given topics

Here only an overview of Deep Learning practices has been illustrated. It will be beyond our scope to discuss each topic in deep due to vastness of the domain. In depth description of loss function, activation function and optimization techniques will be discussed in next chapter. That's why a detailed description is skipped.

CHAPTER

THREE

METHODS

The focus of our thesis has primarily been on the dataset creation procedure, specifically on dataset cleaning and pre-processing methodology. We have dedicated a significant portion of our thesis to discuss these aspects, while comparatively less emphasis has been placed on how to utilize knowledge from the dataset for designing models and training it. Since limited research has been conducted on this topic, our thesis aims to provide valuable insights and tools for future researchers in processing the dataset and evaluating models.

There are two factors that contribute to the complexity of our thesis problem statement. Firstly, collecting and pre-processing the data is an expensive and time-consuming process. Secondly, even after obtaining clean data, it is challenging to establish similarities between places revisited in different years. This is often due to significant changes in image features over time and the presence of invasive species expansion over some areas. As a result, there may be no similar landmarks or reference points available between images from same area over different time period.

In this section, we will provide a detailed methodology for cleaning and pre-processing of the existing raw data. Additionally, we will also look into different architectures that can use our dataset for effectively solving geo-localization problem in underwater environments.

This chapter is divided into three principal components:

- Data collection and pre-processing
- Fine-tuning point cloud registration
- Utilizing dataset knowledge for designing deep learning-based geo-localization model

3.1 Data Collection and Pre-processing

In our thesis, the primary focus is on utilizing 3D information from the dataset to enhance the model's geo-localization capabilities. Traditional 2D image matching

models rely on finding similar features in images, which may not be applicable in real-life underwater scenes, especially in benthic habitats. These habitats have lower visibility and undergo significant changes over time due to variations in species abundance, invasive species impact, pollution, and climate change. To address these challenges, we aim to investigate the potential of leveraging 3D information to mitigate such issues.To utilize such multi year data data and their 3D characteristics, we require 3D point clouds of the same area from multiple years

Firstly, let's delve into the specific 3D information we are referring to. In our model, we employed Structure-from-Motion (SfM) techniques to construct a comprehensive 3D map of the underwater environment for a given year. This approach provided us with a detailed and reasonably accurate interpretation of the scene in three dimensions. Instead of relying solely on 2D information, which includes the x and y coordinates based on the camera frame, we now have a global representation of the underwater site. This representation encompasses x, y, and z values in a global coordinate system that remains consistent over time as oppose to image coordinate system. This global representation is crucial when comparing dataset from the same location across different time periods.

The dataset we are working with is collected as described in Pizarro et al. [14]. They have conducted multiple visits to the same location in 2014, 2015, and 2016, creating maps of the area. To establish precise positioning, they utilized prior pose and depth based on GPS data. By employing these ground references, they generated a 3D representation of the scene using full bundle adjustment.

Data Collection and pre-processing section consists of following subsection:

- 3D Model Creation
- Data Pre-processing
- Camera Pairing

An overview of each section is illustrated in the figure 3.1.1

3.1.1 3D Model Creation

The 3D model was constructed by capturing a series of overlapping images of the underwater environment. The dataset from 2014, 2015 and 2016 consists of 1600, 1234 and 1348 images respectively. To achieve a dense and accurate reconstruction, the Full Bundle Adjustment technique was applied using Metashape. This technique involves optimizing the camera parameters and 3D structure to minimize the reprojection error and ensure precise alignment of the images in the model.

The Metashape works primarily based on the following principle of 3D reconstruction:

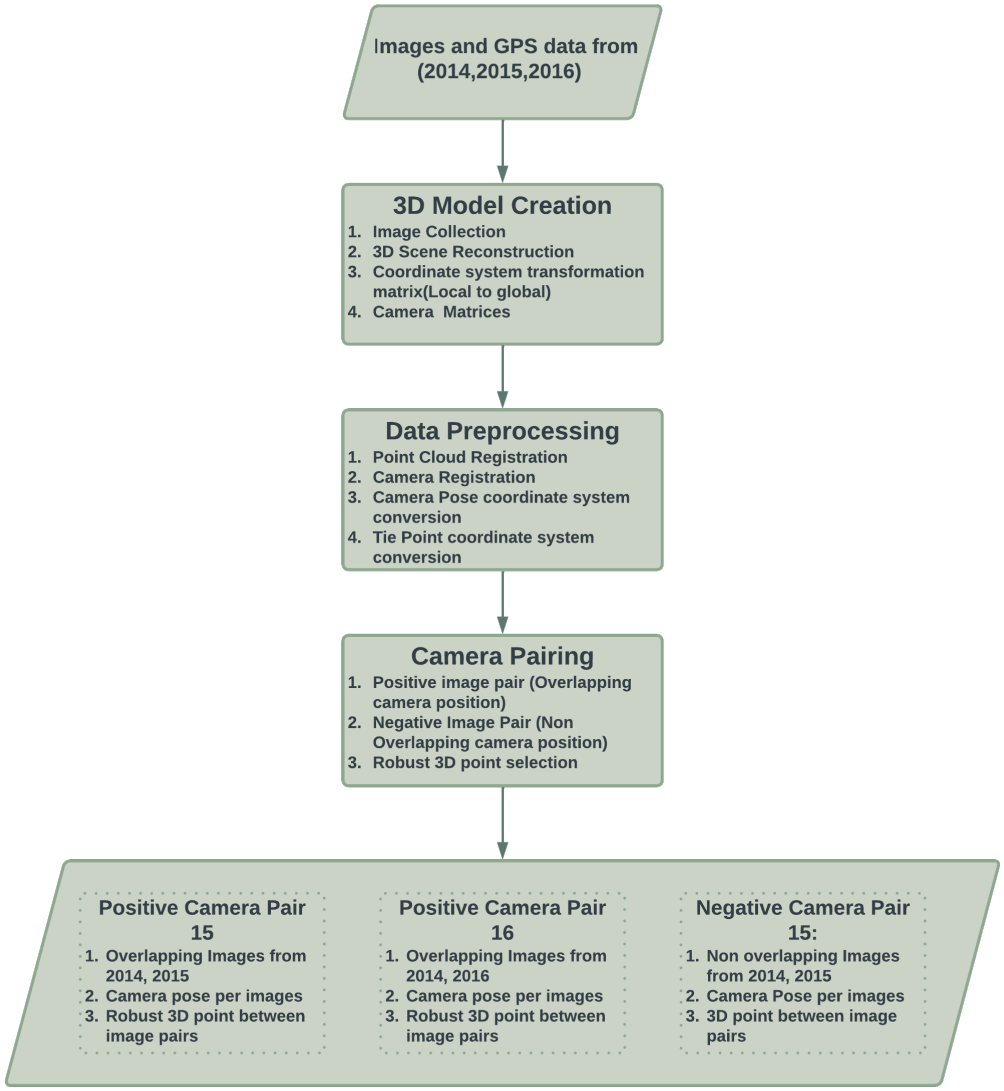


Figure 3.1.1: Data collection and pre-processing flow chart

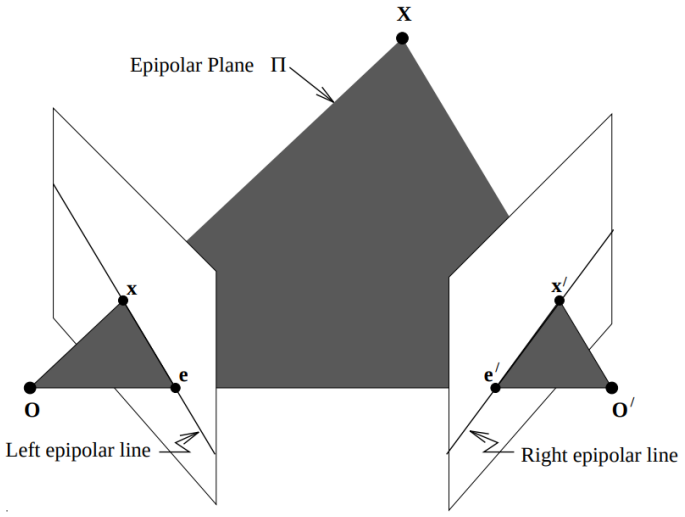


Figure 3.1.2: 3D point triangulation

- Computation of Fundamental Matrix using corresponding points: The Fundamental matrix is computed by considering corresponding points in the images. It satisfies the equation 3.1 for all points involved. With the known values of x, $x^{'}$, The elements of the matrix F can be expressed as linear equations. When there are at least 8 point correspondences, it is possible to solve the F matrix values linearly, up to a scale factor. In cases where there are more than 8 correspondences, a least squares solution can be determined.

$$xFx^{'} = 0 \tag{3.1}$$

- Computation of camera matrix from the fundamental matrix: The camera matrices(internal and external) can be derived by decomposing the essential matrix.

- Computing 3D Points: The 3D points in space are computed based on corresponding matched points between two images using triangulation. The intersection of back-projected lines from these points, which lie on the common epipolar plane, determines the 3D point. However, 3D point correspondence cannot be established for points that lie in the baseline between two images, as they are co-linear.

In Metashape, these processes are automated and user-friendly. The steps followed to create a dense reconstruction from given images are as follows:

- Step One (Alignment):This step involves triangulation and Bundle Block Adjustment (BBA). Metashape generates tie points, which are matched feature points across the images. It also calculates the camera pose for each image, including the estimation and refinement of camera orientation parameters. This step results in a sparse point cloud model and per-camera pose estimation, which are essential for dense map creation and further 3D reconstruction in Metashape. This was done using Metashape GUI software. First we chose the preferred longitude and units (Degree, Minute, second and

Coordinate system unit) from the Preference option at Tools menu. Then we add the image folder from the "Workflow" menu. Then we used "Aligned Photos" option from the same "Workflow" menu(takes around 8 hr on rtx 3070 GPU). This process finds similar features, create tie points and find camera matrices/orientations. We manually assigned markers in the images where ground control points are present. We edit the marker values based on the GPS data. Now based on the marker data, the model will provided a more accurate tie points.

- Step Two (Surface Creation): The second step focuses on creating 3D surfaces, such as meshes or 2.5D representations like Digital Elevation Models (DEMs). Metashape uses polygonal models (meshes) that can be textured for realistic digital representations. Tiled models can be generated for efficient visualization and smooth scene navigation. Metashape utilizes dense stereo matching techniques and the anticipated camera locations to create a dense point cloud, enhancing the level of detail in the model.W e can create the dense point cloud using Build Point cloud Method from the same menu.

- Step Three (Orthomosaic Generation): The final step involves generating an orthomosaic, which serves as a geo-referenced base layer for various types of maps. The images are projected onto a chosen surface (DEM or mesh) using their Exterior Orientation/Interior Orientation (EO/IO) data. By aligning and adjusting the images based on positional and orientation information, a high-resolution and distortion-free orthomosaic is generated. This geo-referenced orthomosaic is crucial for creating accurate and detailed maps, facilitating effective visualization and analysis of geographic data. In similar way as previous steps, we can Build DEM and Build orthomosaic using the workflow menu.

- Step Four (Saving Point Cloud in Appropriate Coordinate system): The generated point cloud from Metashape is in WGS 84 (EPSG::4326). So we first determine the projected coordinate system for our region. Based on the GPS longitude and Latitude, The underwater survey region falls under UTM zone 55S. Now we export the point cloud to WGS 84/UTM zone 55S (EPSG::32755) coordinate system. It is also important keep in mind to save x,y,z shifting value from the year 2014 point cloud and use them for year 2015 and 2016 point clouds. For us, the shifting value was [x, y, z] = [-332400, -8375600, 0] Finally, Metashape automatically does everything about the coordinate conversion based on the user input while exporting the point cloud. After the coordinate conversion, the point clouds for year 14,15,16 look like figure 3.1.3, figure 3.1.4 and figure 3.1.5 respectively:

- Step Five (Exporting Camera Pose, reference, Local to Global transformation Matrix, Camera Intrinsics): In this step we first converted the camera reference coordinate system to WGS 84/UTM zone 55S (EPSG::32755) coordinate system using the transformation pan in reference toolbox. After that we exported the camera in Metashape xml format. In the same time, we also exported the camera intrinsic in OpenCV format.

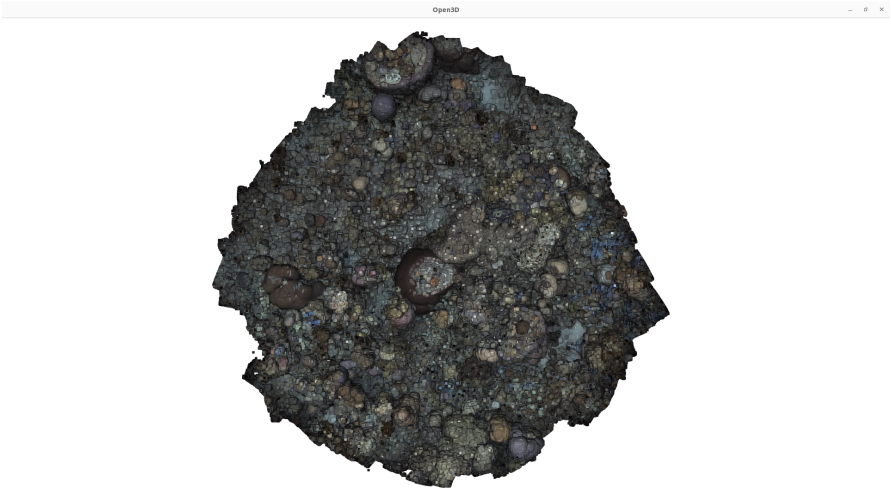


Figure 3.1.3: Dense Point cloud of dataset14

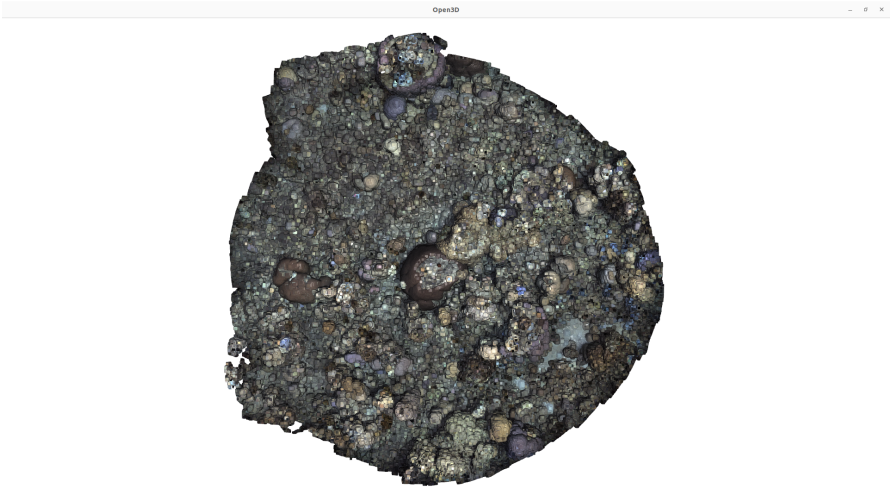


Figure 3.1.4: Dense Point cloud of dataset15

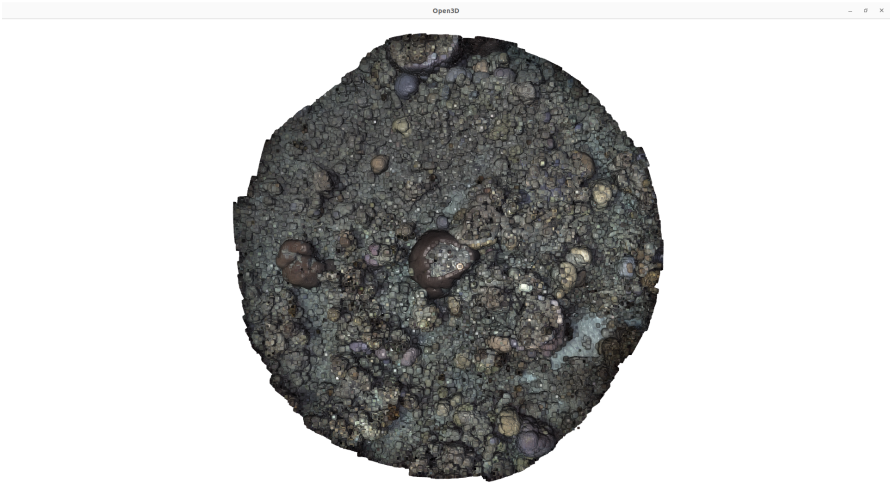


Figure 3.1.5: Dense Point cloud of dataset16

- Step Six (Tie Point selection): Tie points are the image pixels that has multiple correspondence among other images. These correspondences are done using SIFT feature matching inside the Metashape module when the alignment command is running. We have collected tie points per images using the Metashape python API.

After all these steps, now we have a complete 3D representation of the scene in following formats:

- A dense 3D point cloud of the Scene
- Tie points of each images
- Optimized Camera Intrinsic and Extrinsic (Internal and External Orientation)
- Known Coordinate Systems(Global, Local) used to define the cameras and point clouds
- Transformation matrix for conversion between Global and Internal coordinate system

3.1.2 Data Pre-processing:

Point cloud of the scene itself is not suitable to use for cross year evaluation. This is because of the following reasons:

- Misalignment: It is not possible to replicate the exact path or motion of the vehicle while travelling to the same place over different time periods. There will be differences in camera position, orientation of the cameras, path trajectory and depth. Just by observing how different numbers of photos (1600, 1234, 1348 images for the year 2014,2015 and 2016 respectively) were used to reconstruct the same scene from different years, we can validate this analogy.
- The local coordinate system or internal coordinate system that is used by Metashape to represent the camera pose is based on the size and shape of the point cloud. That means internal coordinate systems with also differ among 3D scenes from different years as the shape and size of the point clouds are not identical for each year.

Thus, to make the dataset usable for deep learning models for comparison through extracting features, it must be processed and cleaned. This will result in correct camera correspondence over multiple years. Now the main part of the thesis, that is the methods used to pre-process and clean the dataset will the discussed below in the following steps:

3.1.2.1 Step One (Point Cloud Registration):

Point cloud registration is a crucial process that involves aligning the source point cloud with a target point cloud to maximize their overlap. The objective is to find a transformation matrix that accurately maps one point cloud onto another,

minimizing the spatial differences between them. But one key assumption has to be made before using the registration process. The point cloud must have areas with same or non modified structures over the years. That means, the temporal changes must not be dominant. Several methods are available for point cloud registration, each with its own approach and characteristics. These methods can be broadly classified into two classes of algorithms:

- Global Registration: This class of Registration does not require any initial alignment. As a result, they produce coarse alignment between point clouds compared to the local registration method. They are well-suited for situations where the differences in camera position, orientation, and trajectory between the point clouds are significant.

- Local Registration: On the contrary, local registration methods utilize initial alignments and typically result in a much tighter alignment of the point clouds. These methods leverage the initial estimates to refine the registration iteratively, aiming for a more accurate alignment.

Our approach involves the combination of both local and global registration methods as in figure 3.1.6 due to the following reasons:

- Lack of Initial Alignment: The point clouds from different years, such as 2014, 2015, and 2016, do not have a good initial alignment. The camera positions, orientations, and paths may vary significantly between these time periods. By incorporating both local and global registration techniques, we can compensate for the lack of initial alignment and achieve a more accurate alignment.

- Improved Results with Combination: Combining global and local registration methods has been shown to yield better results in most cases. Global registration methods can provide a coarse alignment and help establish an initial transformation estimation. However, they may not capture fine details or handle significant local variations. On the other hand, local registration methods excel at refining the alignment locally and capturing intricate details. By combining both approaches, we can leverage their respective strengths and achieve a more comprehensive and precise alignment.

- Low computational cost: Both local and global registration methods can indeed be computationally efficient when optimized algorithms and techniques are employed. The computational time required for registration depends on factors such as the size and complexity of the point clouds and the specific registration algorithm used. In the context of our approach, it is important to note that there is no trade-off between computation time and precision (small point cloud and availability of powerful GPU). This means that we can achieve high precision without having a higher computational cost. This is advantageous because it allows us to obtain accurate alignment results while keeping the registration process efficient.

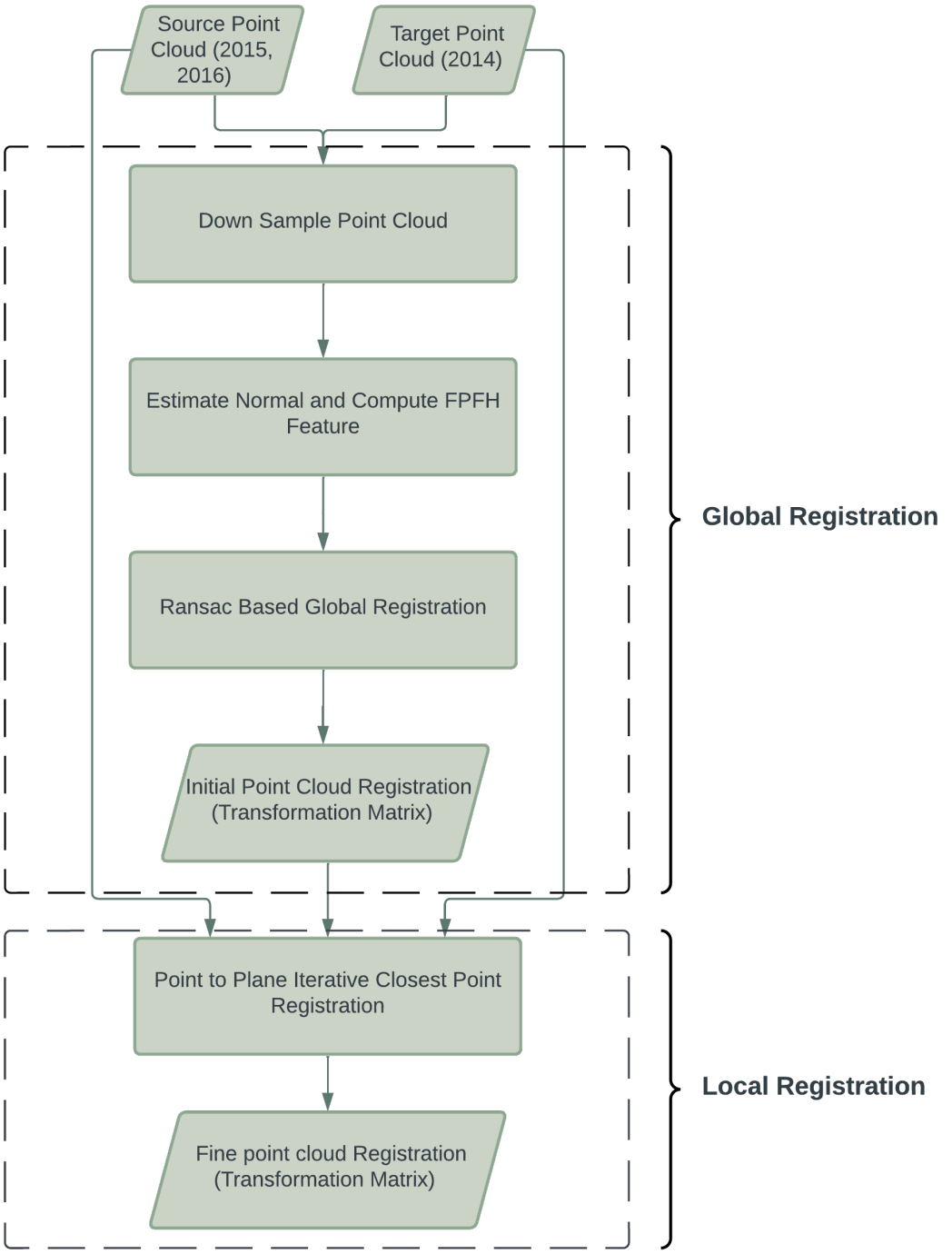


Figure 3.1.6: Point Cloud Registration Flow Chart

Firstly, in case of global registration, there are two widely used global registration methods: RANSAC-based global registration and FAST global registration by Zhou et al. (2016). The choice between these methods depends on our specific problem requirements.

FAST global registration offers good results with lower computational cost, making it an attractive option in many scenarios. However, it may not perform well when there are significant scale differences between the point clouds. In cases

where scale differences are prominent, FAST global registration may fail to provide accurate alignment.

On the other hand, RANSAC-based point cloud registration is known for its robustness to outliers and its effectiveness in dealing with scale differences. RANSAC handles outliers by iteratively selecting minimal sets of correspondences and estimating the transformation matrix that best aligns the point clouds based on these samples. This robustness makes RANSAC particularly suitable for noisy data, where variations and outliers may be present. Although RANSAC may require a bit more computational time (typically in minutes), the robustness it offers against outliers and, more importantly, scale differences is crucial for our multiyear data analysis.

Considering the robustness requirements of our problem, including the need to handle outliers and significant scale differences, RANSAC-based point cloud registration is the more suitable choice. Its ability to robustly align the point clouds despite these challenges outweighs the slightly higher computational cost. By selecting RANSAC-based global registration, we can ensure the accuracy and reliability of our registration process for the multiyear dataset. Following steps were taken to implement RANSAC based global registration using Open3D framework.

- Step One: Visualize unaligned point clouds: First, lets see the point clouds initial alignment. The figure 3.1.7 and figure 3.1.8 are the visual representation of the three point clouds from two different view point. We can see they have good amount of deviation in X, Y and some deviation in Z coordinate. As previously stated in step five of section 3.1.1 "3D Model Creation", all the point clouds are in global coordinate system WGS 84/UTM zone 55S (EPSG::32755) with same amount of shifting. Thus they are directly comparable without any need of coordinate system adjustment.

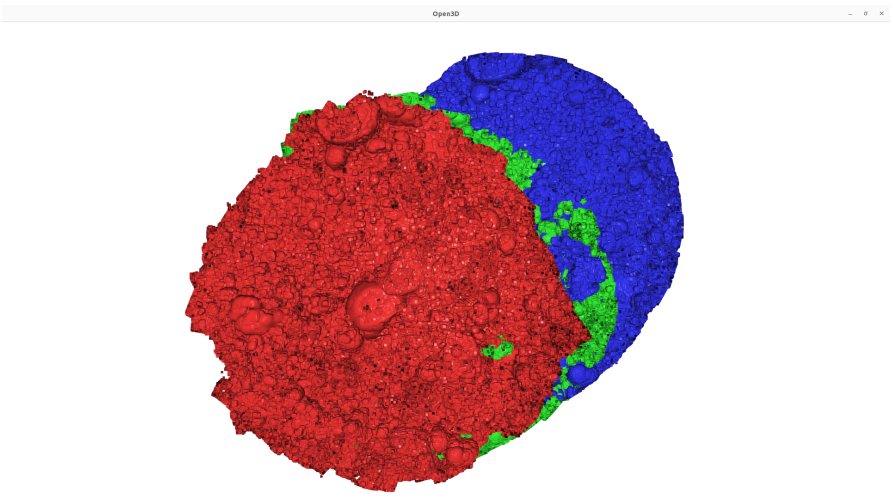


Figure 3.1.7: Top view of point clouds from years 2014(Red), 2015(Green), 2016(Blue)

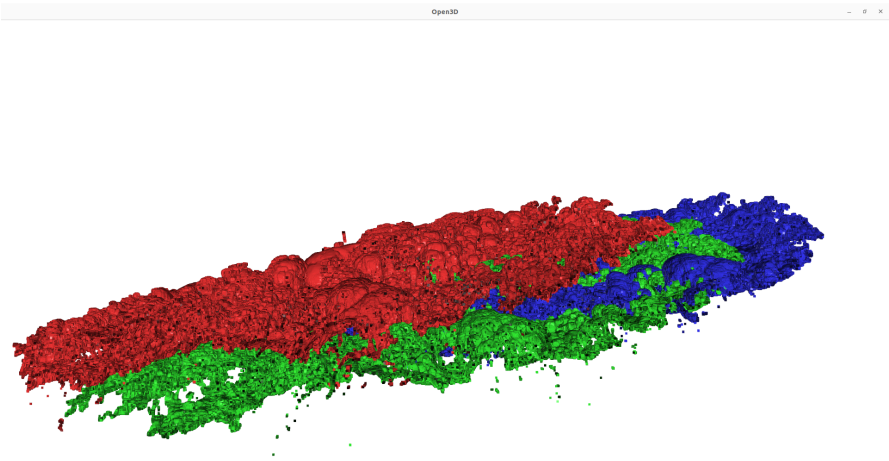


Figure 3.1.8: Side view of point clouds from years 2014(Red), 2015(Green), 2016(Blue)

- Step Two: Now, we will begin the process of down sampling the point cloud. The purpose of this down sampling step is to accelerate the RANSAC process while still achieving a satisfactory initialization for the registration. Additionally, down sampling also addresses scaling differences. For this purpose, we have chosen a voxel size of 0.025 meters which is small enough to provide a very good registration result without any significant increase in computation time. The figures below illustrate the down sampled point cloud. Moreover, we estimate the normals for this new down sampled point cloud using hybrid KDTsearch with radius of 0.05 m and maximum neighbour of 30 as the search parameter to enhance the registration process. By down sampling the point cloud and estimating normals, we can effectively speed up the subsequent RANSAC process while ensuring a good initialization for the registration.Following figures represent the down sampled point clouds.

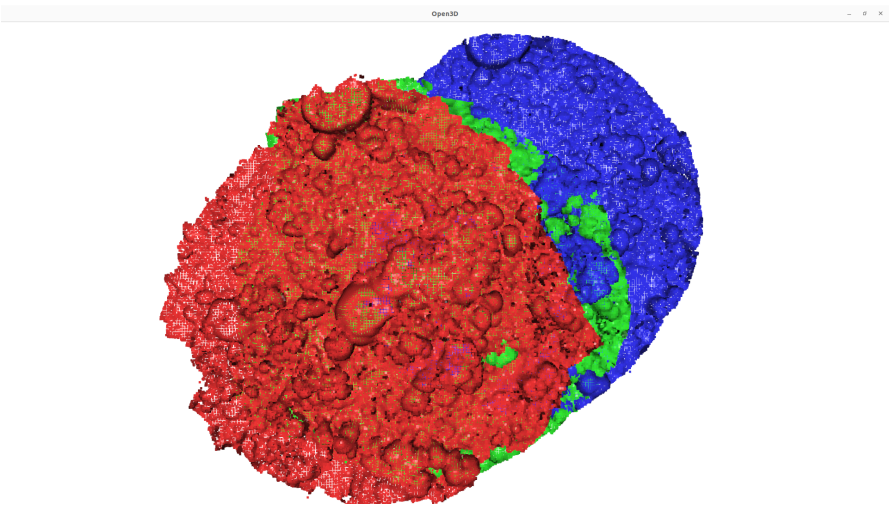


Figure 3.1.9: Top view of down sampled point clouds from years 2014(Red), 2015(Green), 2016(Blue)

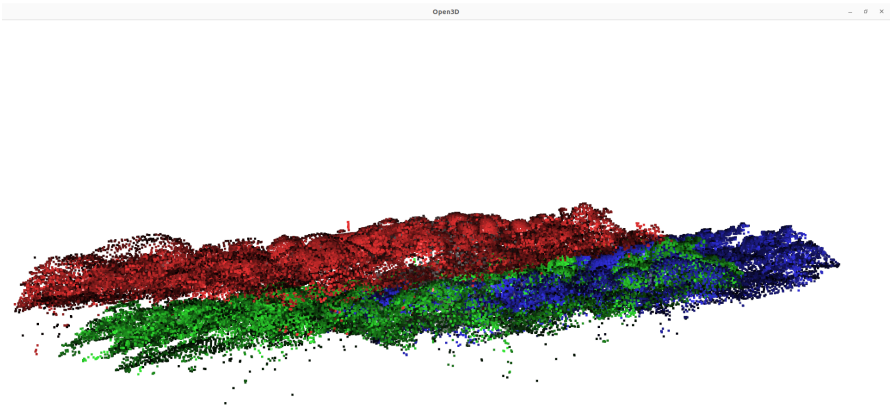


Figure 3.1.10: Side view of down sampled point clouds from years 2014(Red), 2015(Green), 2016(Blue)

- Step Three: We are using feature based RANSAC global registration. So we have to represent each voxel into a feature. A 33-dimensional FPFH feature space(Fast Point Feature Histogram by Rusu [15]) for each down-sampled voxel that has been created. We search and compute the FPFH. using hybrid KDTsearch with a radius of 0.25 m and neighbour size of 100 as the search parameter.
- Step Four: Now we will use RANSAC based registration on feature matching to approximate the transformation matrix for point cloud alignment.

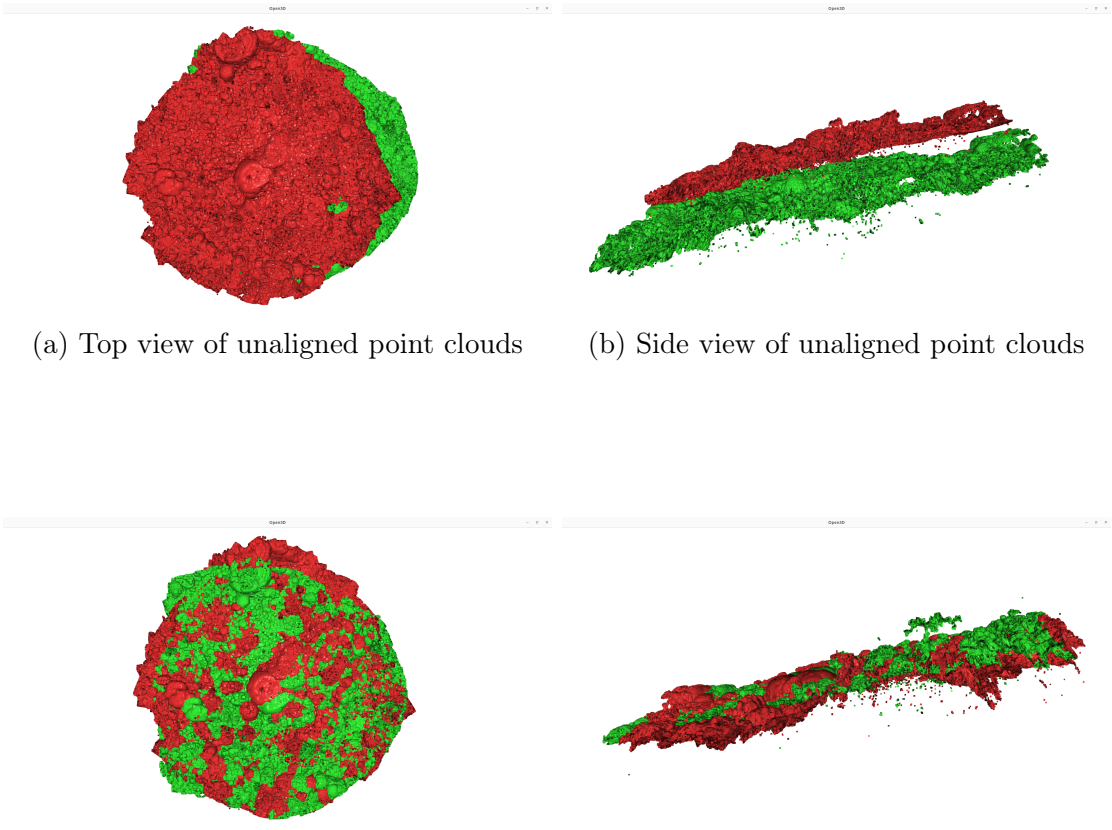
  In Open3D $registration\_ransac\_based\_on\_feature\_matching$ function takes following parameters:

  - source: Down-sampled source point cloud
  - target: Down-sampled target point cloud
  - source feature: Set of FPFH features of down sampled source point cloud
  - target feature: Set of FPFH features of down sampled target point cloud
  - mutual filter: Set to "True". We ensure that each point in the source point cloud has a corresponding point, which refers to itself, by setting the self-correspondence to True. This enables a more comprehensive correspondence set and guarantees that no point is missed during registration.
  - max_correspondence_distance: Maximum correspondence points-pair distance. We set a value of voxel_size*8.5 or 0.2125 m. We put a higher value to increase the search radius. this is due to using down-sampled version of the point cloud.
  - estimation_method: Here, "TransformationEstimationPointToPoint" method of Open3D is used to estimate the transformation matrix,

which is based on minimizing the point-to-point distances between corresponding points in two point clouds. We also set scaling factor "True" to take care of point cloud scaling difference. This method provides better result in the presence of scale difference between point clouds.

- ransac_n: We used the default value of 3. That means the model uses RANSAC with 3 correspondences.
- checker: To check if aligned point clouds are close (less than specified threshold). In our case, We used both correspondence checker based on edge length (threshold 0.9) and correspondence checker based on distance(voxel_size*8.5).
- criteria: We set max RANSAC iteration to 1000000, and confidence to .99999.

To visually demonstrate this, Figure 3.1.11 showcases the result of the RANSAC-based point cloud registration of the 2015 point cloud to the 2014 point cloud.



(a) Top view of unaligned point clouds

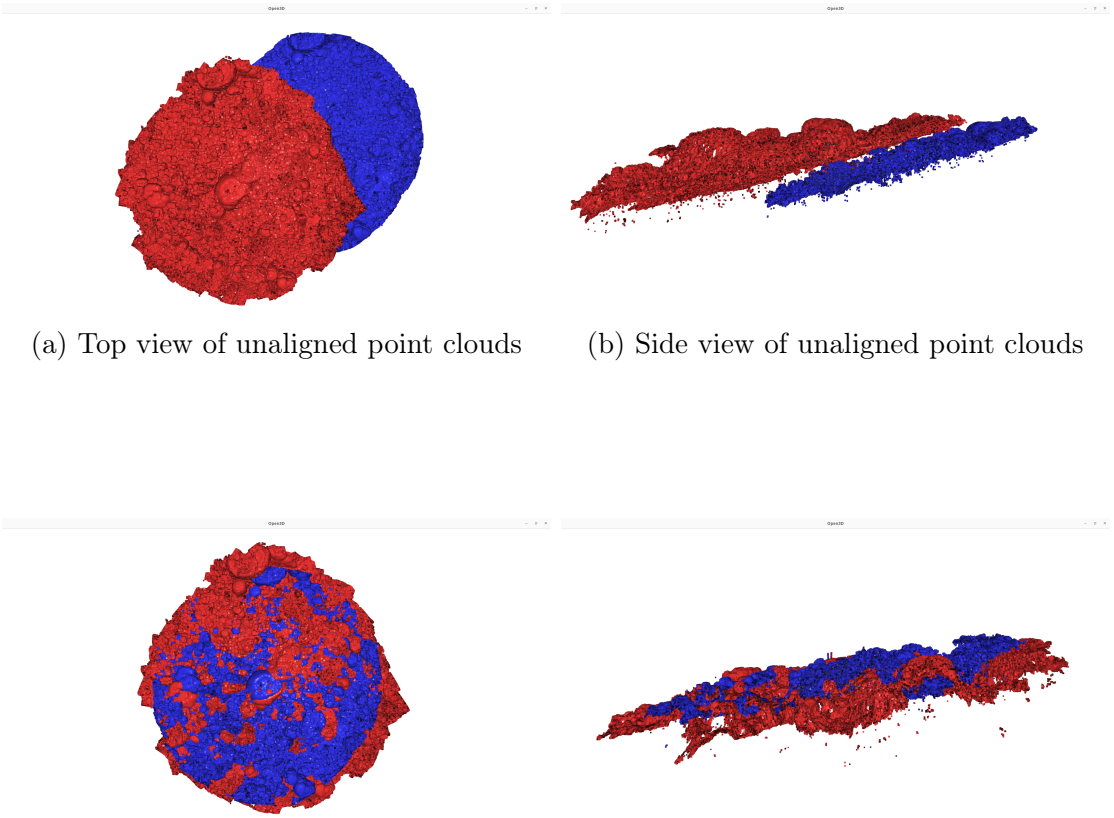
(b) Side view of unaligned point clouds

(c) Top view of aligned point clouds

(d) Side view of aligned point clouds

Figure 3.1.11: RANSAC based global registration from 2015 to 2014 (Red:2014, Green:2015)

Similarly, Figure 3.1.12 presents the corresponding result for the registration of the 2016 point cloud to the 2014 point cloud.



(a) Top view of unaligned point clouds

(b) Side view of unaligned point clouds

(c) Top view of aligned point clouds

(d) Side view of aligned point clouds

Figure 3.1.12: RANSAC based global registration from 2016 to 2014 (Red:2014, Blue:2016)

It is not possible ti understand which registration method is performing well. To compare them quantitatively, in the results section, we present a comprehensive performance comparison between two feature-based global registration algorithms: FAST and RANSAC. Upon conducting the initial registration step, it was observed that RANSAC produced highly similar point clouds even in the initialization step. By performing a detailed analysis of the registration outcomes, we found that RANSAC consistently outperformed FAST in terms of accuracy and robustness. RANSAC exhibited a higher degree of correspondence between the matched features in the two point clouds, leading to a more precise alignment. Furthermore, RANSAC demonstrated superior resistance to outliers and noise compared to FAST, resulting in more reliable and consistent registration results.

Now, after establishing an initial alignment between the point clouds using global registration, we can proceed to finely adjust and enhance the registration through the utilization of a local registration method. In this case, we will be using an effective and popular Iterative Closest Point (ICP) algorithm. ICP enables us to refine the alignment and achieve a more accurate registration of the point clouds. In general there are two versions of ICP, Point-to-point with objective function as in equation 3.2 and Point-to-plane with objective function in equation 3.3 . We will be using the extended version of ICP known as Point-to-plane ICP.

$$E(T) = \sum_{(p,q)\in K} ||p - Tq||^2 \tag{3.2}$$

$$E(T) = \sum_{(p,q)\in K} ((p - Tq).n_p)^2 \tag{3.3}$$

Where, $E(T)$ is the transformation matrix (T) estimation, p (target point) and q (source point) are from the correspondence set K. n_p is the normal of point p. Point-to-plane ICP improves registration accuracy compared to point-to-point ICP by considering the surface geometry of the point clouds. Here are a few reasons why we choose point-to-plane ICP rather than point-to-point ICP:

- Surface geometry: Point-to-plane ICP considers the local surface geometry by aligning points with respect to the tangent planes of the target point cloud. This is particularly beneficial when dealing with non-rigid deformations or surfaces with varying curvatures. By aligning points to the tangent planes, it can handle local deformations and non-uniform surface structures more effectively.

- Noise robustness: Point-to-plane ICP tends to be more robust to noise compared to point-to-point ICP. Since the distances are computed between points and tangent planes rather than point-to-point distances, it reduces the influence of noisy or outlier correspondences. By incorporating the normal information, point-to-plane ICP can discard outliers that do not conform to the local surface structure.

- Non-uniform point densities: Point-to-point ICP assumes a uniform distribution of points in the surfaces being aligned. However, in real-world scenarios, the density of points can vary across the surfaces. Point-to-plane ICP handles non-uniform point densities more gracefully because it aligns points based on the tangent planes, which can compensate for the variations in point densities.

- Convergence speed: In some cases, point-to-plane ICP may converge faster than point-to-point ICP. By using the orthogonal distances between points and tangent planes, the optimization problem can be solved more efficiently, leading to faster convergence and improved runtime performance.

For implementing point-to-plane ICP, we need a good initialization. We used the output of global registration as the initial alignment for the point-to-plane ICP algorithm to work on. Following steps and parameters were taken, provided we have the initial alignment from the global optimization.
We are using *multi_scale_icp* instead of a single scale icp. The *multi_scale_icp* function in Open3D offers notable advantages compared to single-scale ICP. By adopting a multiscale approach, the algorithm begins with a higher voxel size and progressively refines the registration by iteratively transitioning to lower voxel sizes. This approach brings two significant benefits to the registration process.

Firstly, the multiscale strategy employed by *multi_scale_icp* yields exceptional accuracy. Starting with a higher voxel size allows for a more coarse alignment,

effectively capturing large-scale correspondences between the point clouds. As the algorithm proceeds to lower voxel sizes, it refines the alignment on finer scales, capturing intricate details and achieving highly accurate registration results. This multiscale progression ensures that the registration process adapts to the varying levels of details present in the point clouds, leading to improved overall accuracy.

Secondly, the use of progressively lower voxel sizes significantly reduces the computational time required for registration. Rather than solely relying on a single, fine-grained voxel size, the algorithm leverages the coarse-to-fine approach to efficiently converge towards the optimal alignment. By starting with larger voxel sizes, which encompass larger point neighborhoods, the algorithm can initially identify correspondences at a faster pace. As the voxel size decreases, the algorithm shifts its focus to smaller point neighborhoods, narrowing down the search space and refining the alignment. This hierarchy of voxel sizes allows for a more efficient use of computational resources, enabling accurate registration without using excessive processing time.

The *multi_scale_icp* takes following parameters:

- Source: Source point cloud (point clouds from the year 2015, 2016)
- Target: Target point cloud (point clouds from the year 2014)
- Voxel_size: The hierarchical approach of the voxel sizes used in the registration process begins with an initial voxel size of 0.1. Subsequently, the voxel sizes are successively halved in two additional steps, resulting in $[0.1, 0.05, 0.025]$.
- Criteria_list: Similarly, the hierarchical criteria list comprises of three different values corresponding to the voxel sizes: relative_fitness = [0.000001, 0.0000001, 0.00000001], relative_rmse = [0.000001, 0.0000001, 0.00000001], and max_iter = [50, 20, 10].
- Max_correspondence_distances = [.1, .05, 0.025] meters
- Initial transformation matrix: We used the output of our global registration as the initial transformation matrix.
- Estimation: To estimate the transformation, we utilized the TransformationEstimationPointToPlane class with a robust kernel. For computing the residuals and Jacobian matrices of the objective function, we employed the TransformationEstimationPointToPoint class. To enhance the algorithm's robustness against outliers, we incorporated the tukey loss as suggested by Open3D. This robust kernel works by down-weighting large residuals that are likely caused by outliers. By reducing their influence on the solution, the tukey loss optimizes the objective function E(T) to mitigate the impact of outliers. If $r_i(T)$ is i^{th} the residual, for a given pair of correspondences $(p, q) \in K$ we can convert the objective function as in equation 3.3 to equation 3.4

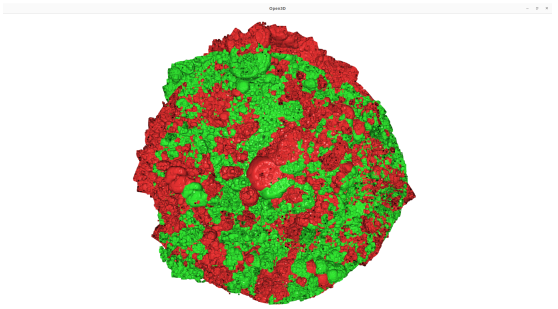
$$E(T) = \sum_{(p,q)\in K} \rho((p - Tq).n_p)^2 = \sum_{i=1}^{N} \rho(r_i(T)) \tag{3.4}$$

Where $\rho(r)$ is the robust kernel.

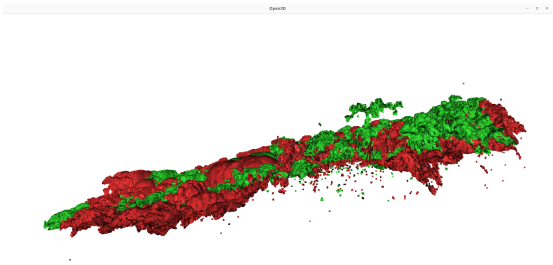
The optimization problem above can also be solved by using the iteratively reweighted least-squares (IRLS) approach, which solves a sequence of weighted least squares problems as:

$$E(T) = \sum_{i=1}^{N} \frac{1}{r_i(T)} \rho^{'}(r_i(T))^2 \tag{3.5}$$

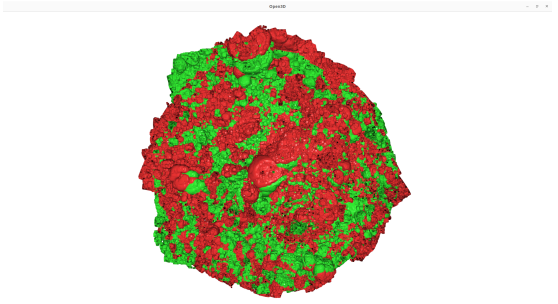
By using the aforementioned steps, we increased the precision of our point cloud alignment through a synergistic blend of Global and Local point cloud registration algorithms. The combination of these two methods produces a refined outcome, which can be seen in the following figures 3.2.2 and 3.1.14. Using both global and local registration methods results in more consistent alignment of point clouds. From the figure we can observe a more homogeneous distribution of colors/point clouds compared to relying solely on local registration. While the visual improvements may not be immediately obvious, the statistical findings also demonstrate a slight increase in registration accuracy when employing the combined local and global registration approach.



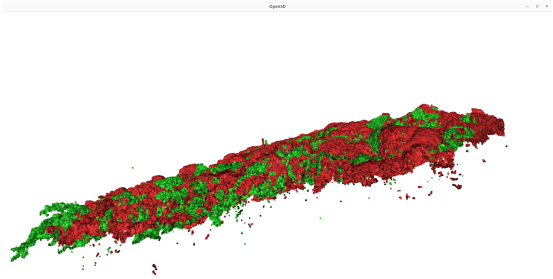
(a) Top view of aligned point clouds using RANSAC



(b) Side view of aligned point clouds using RANSAC

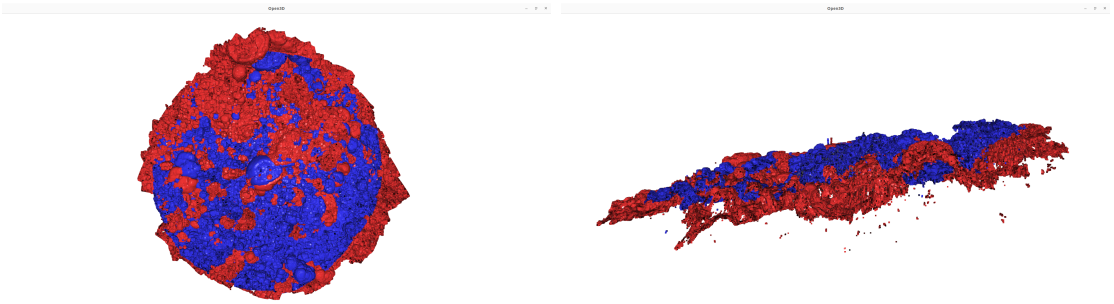


(c) Top view of aligned point clouds using RANSAC+ICP



(d) Side view of aligned point clouds using RANSAC+ICP

Figure 3.1.13: RANSAC based global registration (a, b) and RANSAC based Global + Point-to-plane ICP based Local registration (c, d) from 2015 to 2014 (Red:2014, Green:2015)



(a) Top view of aligned point clouds using RANSAC

(b) Side view of aligned point clouds using RANSAC

(c) Top view of aligned point clouds using RANSAC+ICP

(d) Side view of aligned point clouds using RANSAC+ICP

Figure 3.1.14: RANSAC based global registration (a, b) and RANSAC based Global + Point-to-plane ICP based Local registration (c, d) from 2016 to 2014 (Red:2014, Blue:2016)

3.1.2.2 Step Two (Camera Registration and Pose transformation):

Once we have obtained a transformation matrix to align all the point clouds to a single coordinate system, it is necessary to align the cameras based on their orientation in each respective coordinate system to a unified reference. However, this process is more complex than simply using the transformation matrix obtained from the point cloud registration for aligning the camera orientations. This is due to the following reasons:

- The camera orientations are described in internal coordinate system. Internal coordinate systems are local coordinate system that depends on the point cloud characteristics. This means that the camera coordinate system differs from one point cloud to another.
- It is important to note that coordinate shifting was applied during the visualization and registration of the point clouds. However, the internal or local coordinate values of the cameras do not consider this shifting. Consequently, before applying the transformation obtained from point cloud registration, it is necessary to manually adjust the camera poses to account for the coordinate shifting.

To address these challenges, we begin by converting the camera orientations from the local coordinate systems to align with the point cloud's coordinate system. Additionally, a coordinate shift, similar to the one employed during the export process in Metashape, needs to be applied. Metashape conveniently provides a transformation matrix for converting the local coordinate system to the global coordinate system. The entire process involves the following steps:

- Local to global coordinate conversion: The camera orientation is described in local coordinate systems which depends on point cloud structural characteristics. To convert the local camera orientation to global coordinate system, we multiplied the camera pose with the coordinate transformation matrix provided in the Metashape camera.xml file.

- Shift coordinates: Now that we have the camera pose in Geographic coordinate system (WGS 84 (EPSG::4326)), we converted them to the projected coordinate system (WGS 84/UTM zone 55S (EPSG::32755) similar to point cloud conversion during Metashape point cloud exporting process. Then we shifted the coordinate by [x, y, z] = [-332400, -8375600, 0]

- Camera Position Alignment: we get an aligned camera orientation by simply multiplying the transformation matrix from point cloud registration. Figure 3.1.15 and 3.1.16 refer to initial and registered position of cameras for the year 2014, 2015 and 2016

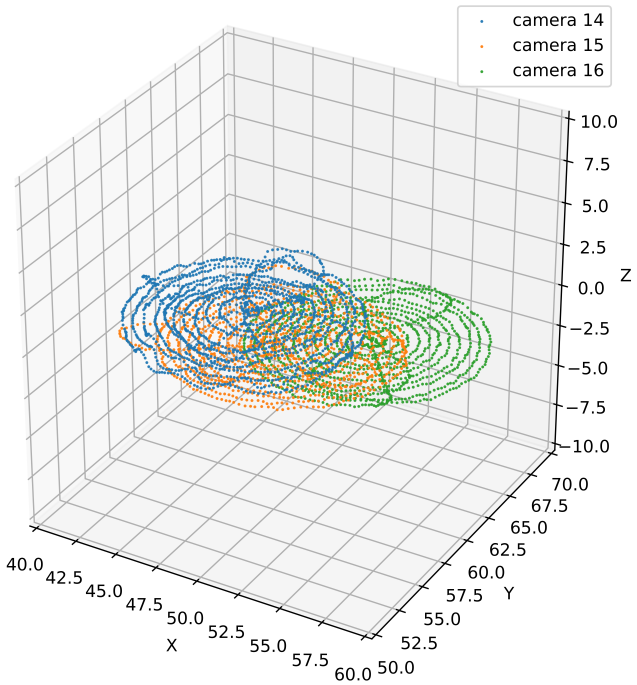


Figure 3.1.15: Initial unaligned camera position for the year 2014, 2015, 2016

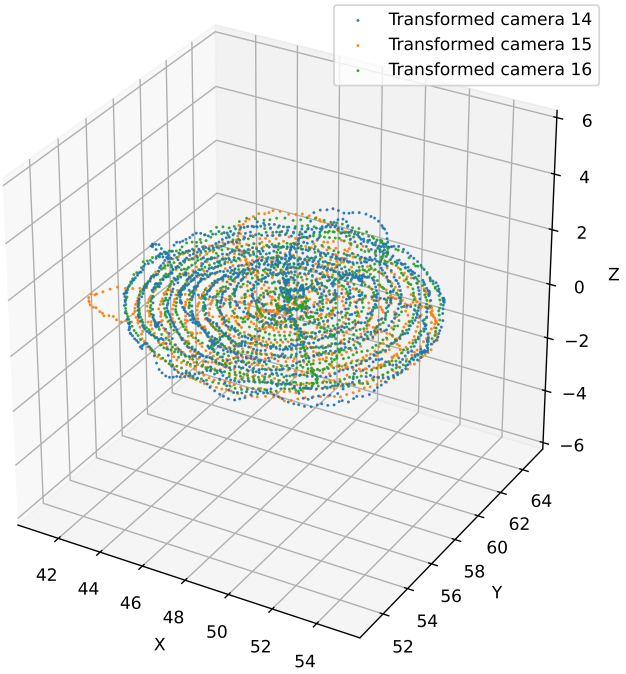


Figure 3.1.16: Aligned camera positions for the year 2014, 2015, 2016

3.1.2.3 Step Three (Tie Point coordinate system conversion):

In order to utilize the 3D information within deep learning model, we need to convert the coordinate system of the 3D points to an aligned point cloud coordinate system. This process is facilitated through the use of the Metashape software and the Pyproj library. The detailed steps for this transformation will be discussed in the Camera Pairing section.

3.1.3 Camera Pairing:

To finalize the dataset, we have to create positive image pair and negative image pair. Generally, negative image pair is only used to train the model. For such reason there is no need for creating negative image pair for our test dataset. There are three parts in this section. First, we have to pair images and secondly we have to find correspondence point between positive image pairs from two different years. Finally, we have to select the robust correspondence points based on their similarity in 3D space(Are they pointing to same or near 3D points or not?). We will not find any correspondence point/tie point in negative image pairs as they do not share any common 3D points between them.

3.1.3.1 Step One: Positive and Negative image pairs:

We carefully established pairs of positive and negative images by considering the differences in camera positions within a 3D space. The cameras were all looking

straight down, so we didn't have to worry about if they are looking upward or side wise. Instead, we focused on the variations in the x, y, and z values of the cameras.

When it comes to the z values, more higher above the surface provides a wider field of view compared to a lower position closer to the surface. We took this into account when setting a threshold for the x and y values, which determine whether the cameras overlap or not.

Initial $threshold\_train$ and $threshold\_test$ values are determined based on min max z values for all the 3D points of each year rather than two cameras. It helps to create balanced number of test and train image pairs. For determining this dataset balancing threshold we used equation 3.6, 3.7 and 3.8

$$ratio\_global = \frac{max\_z\_test - min\_z\_test}{max\_z\_train - min\_z\_train} \tag{3.6}$$

$$threshold\_train = 0.2 * (ratio\_global + 0.1) \tag{3.7}$$

$$threshold\_test = 0.2 \tag{3.8}$$

Where, max_z_test, min_z_test are maximum and minimum z values for point cloud 2016. Similarly, max_z_train, min_z_train are maximum and minimum z values for point cloud 2015

Now for comparing between two images, following equation 3.9 to 3.11 for determining x, y threshold based on z ratio was used for positive pairing between two cameras and 3D points present in them:

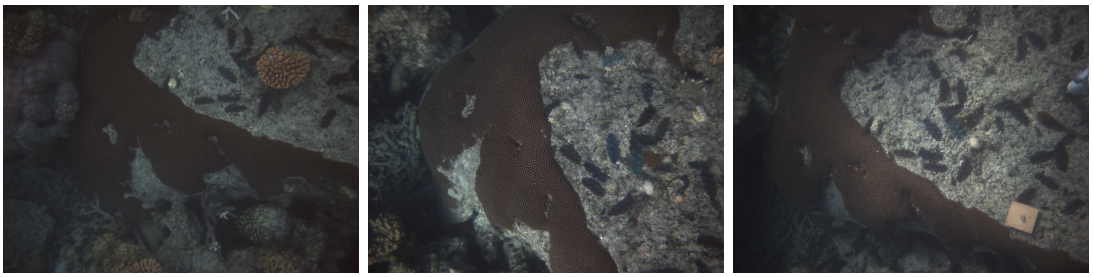
$$\begin{aligned} ratio\_image\_pair &= \frac{-0.10 * (max\_depth - z\_diff)}{max\_depth} for\ z\_diff < 0 \\ &= \frac{0.10 * (max\_depth - z\_diff)}{max\_depth} for\ z\_diff > 0 \end{aligned} \tag{3.9}$$

$$threshold\_train = 1.2 * threshold\_train - ratio\_image\_pair \tag{3.10}$$

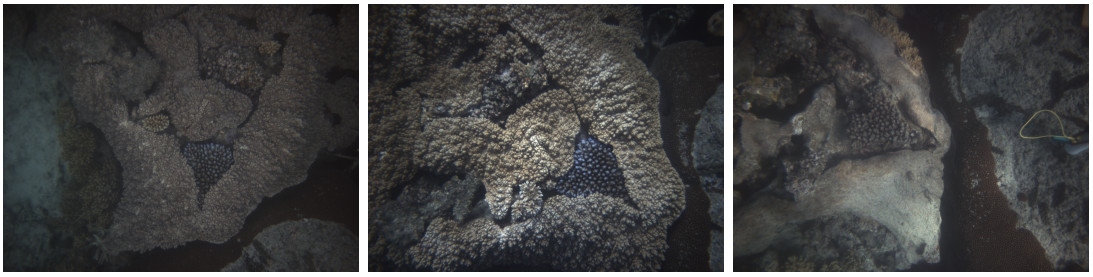
$$threshold\_test = threshold\_test - ratio\_image\_pair \tag{3.11}$$

Where, max_depth is the maximum depth among all the 3D points present in two cameras we are comparing, z_diff is the difference of z value between two cameras,one from 2014, and the other from 2015 for train set and one from 2014, and the other from 2016 for test set.

To create positive image pairs, we compared the positions of each camera from 2014 with the cameras from 2015. If the difference in the x and y values was lower than the threshold we determined, we considered them as a positive pair. We applied a similar approach to create positive image pairs for the test dataset, using cameras from 2014 and 2016.

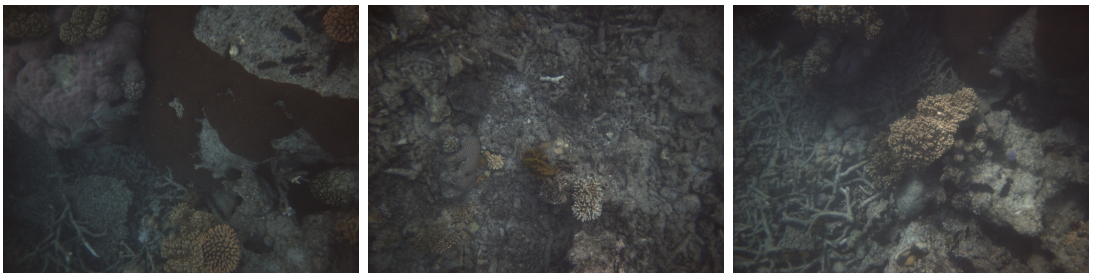


(a) Positive sample from 14 (b) Positive sample from 15 (c) Positive sample from 16



(a) Positive sample from 14 (b) Positive sample from 15 (c) Positive sample from 16

In the above figures, from the first row we can see that positive image sample from overlapping area have little temporal changes. Whereas, the second row positive image sample also from overlapping areas has significant temporal changes.
On the other hand, for negative image pairs, we wanted to select images that were far apart. To achieve this, we used a higher threshold value and only considered camera pairs where the differences in x and y values were greater than the threshold. For the negative dataset pairs, we only used cameras from 2014 and 2015, as they were specifically intended for training purposes.



(a) Negative sample from 14 (b) Negative sample from 15 (c) Negative sample from 16

The above illustrated figures show negative samples with non overlapping camera images.
By establishing both positive and negative image pairs, we aimed to create a comprehensive dataset that covers different camera positions and their relationships, enabling effective training and testing of our models.

3.1.3.2 Correspondence Point Selection:

- Correspondence Points: These are points that establish correspondences between two or more images captured by different cameras. They represent

the same physical point in the scene but appear differently in each image due to the varying viewpoints. By identifying these correspondence points, we can link the 2D image information to their corresponding 3D points.

- First, we have to obtain the 2D correspondence between the image pair using SIFT feature matching. We are selecting a total of 80000 points from 3 octave layers. The reason for taking so many points is to ensure that we are not missing any correspondence that may get neglected during point selection based on gaussian noise response. We then calculate the SIFT descriptor of all these points and do point correspondence using Brute force matching with threshold value from 0.74 to 0.9. This threshold value is set to an lower and upper limit rather than a constant value as number of matches can be very low for lower threshold depending on the image pairs. After we get the good point correspondence, we will remove the outliers and only take robust matches based on their corresponding 3D points.

- Now to get the 3D points of the matched 2D points we took following steps.

  - Transform camera matrices from local coordinate system to global coordinate system
  - Then we find the 3D points of each 2D points using camera matrices of each image respectively.
  - We determined the tie points (3D points of matched 2D points) coordinate system. We found out that it uses EPSG 4978 as tie point global coordinate system.
  - Transform 3D points from EPSG 4978 to EPSG 32755 (aligned point cloud's coordinate system) and apply coordinate shift of [x, y, z] = [-332400, -8375600, 0]
  - Finally, We transformed the 3D points using the transformation matrices we got from point cloud registrations for the year 2015, 2016

3.1.3.3 Robust 3D point selection:

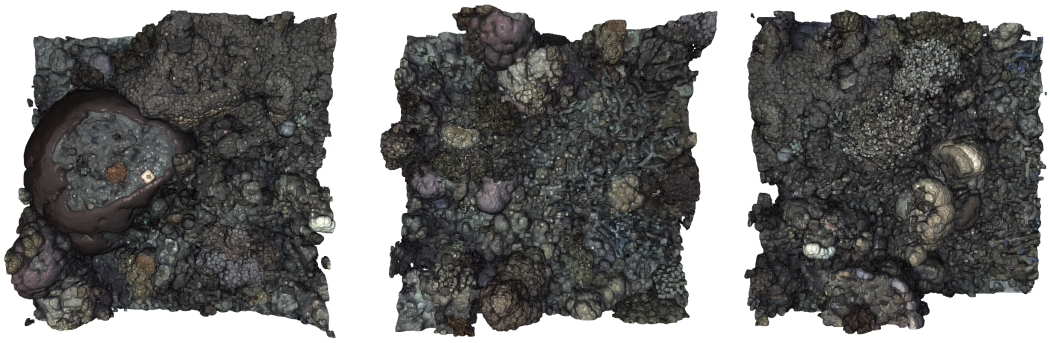
Creating robust 3D points is a straightforward process. We compare each corresponding 3D points obtained from previous section based on comparing image one to image two in the positive image pairs. By calculating the differences between these points, we sort them accordingly. We used a difference threshold of 10 cm x,y and z axis to declare that the point correspondence are robust. The reason to choose 10 cm is to compensate for error accumulated during 3D reconstruction, point cloud registration and coordinate conversion. We also chose the value 10 cm as the images are 1-1.3 meter in height and width. This 10 cm difference gives us with a good estimate of where the temporal changes are not present in the image

By carefully selecting these subset of points with minimal differences, we ensure that our dataset consists of highly reliable and accurate 2D correspondence based on their 3D characteristics . These robust points between positive image pairs are immune to temporal changes and essential for deep learning based geo-localization model training.

3.2 Fine-tuning point cloud registration:

Theoratically, the point cloud registration should provide us means to compare 3D points associated to images. from different years. But in reality, the transformation matrix obtained from the point cloud doesn't perform well in every part of the point cloud. And it significantly reduces number of robust 3D points. More importantly, it creates a lot of wrong correspondence which is what we are trying to avoid. To solve this, we repeated the registration process But this time, with the output of previous registration as its initial transformation matrix and most importantly the registration was done for a much smaller subsections of the point clouds or chunks. During doing this, There are two things to consider here:

- **Selection of chunks:** It is important to decide how we want to divide the point cloud into multiple chunks. More importantly we have to select the center/location and radius/area for such fragmentation. In our case, it is only beneficial that we have good registration for the image we are working on.So it is obvious that the center of each fragmentation should be the same location as each camera center. In addition, one image covers around 1.1 meter of area. thus we have taken a square of 2 meter length per side for creating a chunk of area 4 meter square. Now we have a total of 1234 chunks for 2015 and 1348 for 2016. This number was for the source point cloud. Similarly we have fragmented the target point cloud based on its closest point to the corresponding source point cloud's chunk center, also with an area of $4m^2$.



(a) Center chunk (b) Side chunk (c) Side chunk

Figure 3.2.1: Point cloud chunks of size $4m^2$ from 3 different position of the point cloud

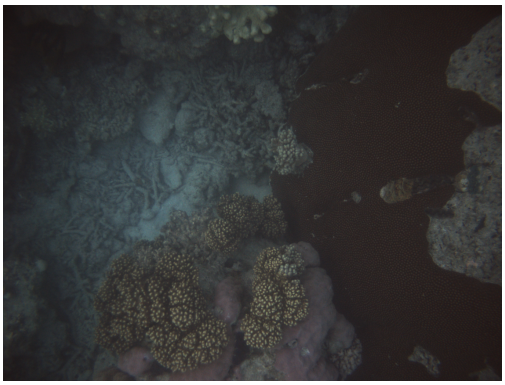
- **Registration of Chunks:** The registration process for the chunks followed a similar approach to the registration of the entire point cloud. First, we performed RANSAC-based global registration using the output from the previous registration process. Then, we refined the registration further using point-to-plane ICP for local registration. Given the large number of chunks, it is not feasible to show all of them. Instead, we will display a representative sample of the chunks, ensuring a homogeneous distribution across the point cloud.
- **Robust Correspondence based on chunk Registration:** Finally, we determine the 2D correspondence based on similar images from each camera

year. Then we perform 2D to 3D mapping and ultimately compare them by using chunk transformation matrix.

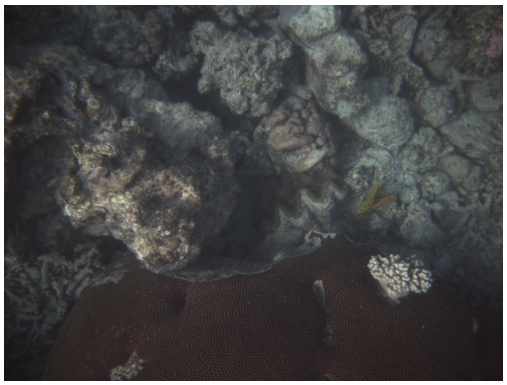
Most importantly, we found out that, using chunk registration we are getting higher number of robust matches and no wrong matches at all. This means, multiple chunk registration approach solves all the problem that raised due to the generalization error during whole point cloud registration.

Finally, we have a dataset that has:

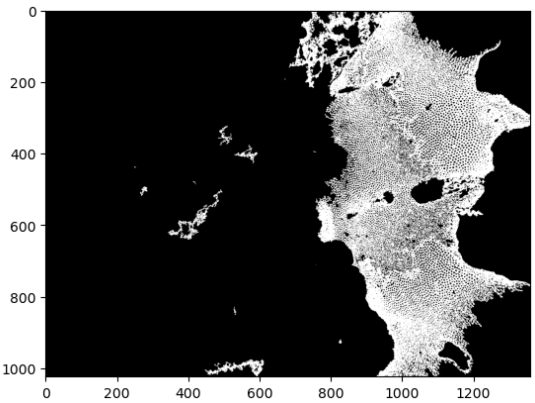
- Per Camera Point cloud Registration Matrix across multiple years
- Robust 2D and 3D correspondence with minimal temporal changes
- Image mask illustrating areas with minimal changes using region growing algorithm at robust 2D points. For example look at following figures:



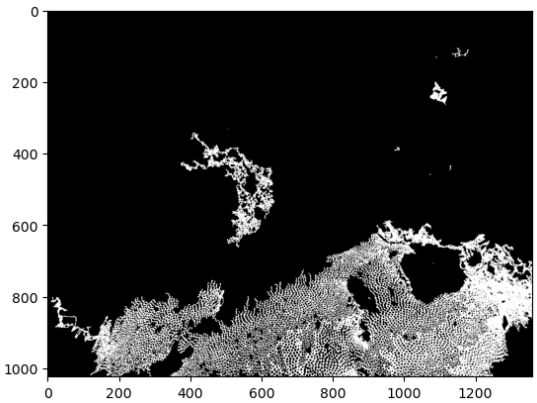
(a) Positive image from year 2014



(b) Positive image from year 2015



(c) Robust mask extracted from positive image from 2014



(d) Robust mask extracted from positive image from 2015

Figure 3.2.2: Top row: overlapping images from different years. Bottom row: masks with white regions indicating minimal temporal changes for the image (a) and (b) respectively

3.3 Utilizing our dataset knowledge for designing deep learning-based geo-localization model

Now, we will discuss what kind of deep learning models can best use our dataset:

- **Model with pose-based loss function:** In this model, our dataset will provide robust correspondence to establish accurate relative pose between two images without the need of using RANSAC loop for outlier rejection. It is more robust against outliers than RANSAC is.
- **Model based on semantic segmentation:** The mask created using region growing around robust key-points can work as a ground truth for models to produce image masks that only shows areas with minimal temporal changes. For example, it can be used in a UNET Based Siamese network for creating temporal image masks.
- **Model which needs 3D information as an input:** The close alignment of smaller chunks obtained by our proposed system can provided better and accurate 3D structural information for models that take images and their 3D models as inputs for image comparison tasks.

CHAPTER

FOUR

RESULTS

The primary motivation behind this thesis was to develop a system that enables researchers to leverage both 3D information and image data by preprocessing and cleaning the dataset, ensuring precise and accurate information for analysis. Our work focused on two key objectives:

- Comparing Point Clouds Across Multiple Years: We aimed to address the challenge of comparing point clouds with temporal changes over different years. This objective involved developing methods to effectively analyze and identify differences between point clouds acquired at different time intervals.
- Establishing Robust Correspondence Between Images: Another objective was to establish robust correspondences between pairs of images. We sought to develop techniques that enable accurate alignment and matching of features across images, facilitating subsequent analysis and comparison.

Additionally, we highlight specific systems or applications where our proposed methodology can excel.

Based on our comprehensive analysis and results, we have successfully achieved the intended objectives outlined above. In the following sections, we will discuss the results obtained for each step of the process, starting with the reconstruction phase.

4.1 3D Reconstruction:

Although 3D reconstruction was not the primary focus or objective of our problem statement, we will provide a brief overview in this section. For a more detailed understanding, interested readers can refer to the reconstruction report provided in the appendices with all the information parameters are given there including camera calibration matrix. During the reconstruction phase, we evaluated the performance using the following performance measures for each year:
Please note that while reconstruction results will be briefly discussed, the subsequent sections will primarily focus on addressing our main objectives of comparing point clouds from multiple years and establishing robust correspondence between images.

Table 4.1.1: Reconstruction Report

Year	N images	Tie points	Projections	Reprojection error	Area
2014	1,600	984,417	3,917,915	0.560 pix	189 m^2
2015	1,234	1,075,143	3,887,739	0.501 pix	206 m^2
2016	1,348	730,467	4,127,915	0.570 pix	185 m^2

4.2 Point Cloud Registration:

The main part of our thesis was totally devoted to this section. We have compared different registration methods and have already discussed in methodology section regarding our selection of the combination of both global and local registration. Or by name, RANSAC based Local and Point to Plane based ICP local registration. Following table illustrated our findings among all the registration methods tasted:

Table 4.2.1: Registration Result: Source Point cloud 2016 to Target Point cloud 2014

Method	Fitness	Max RMSE (m)	Min RMSE (m)	Avg RMSE (m)	Time (s)
RANSAC	0.9034	2.3425	0.00001951	0.09476	3641
FAST	0.8525	2.0265	0.0000185	0.1166	1074
RANSAC + ICP	0.5241	1.9757	0.0000189	0.0932	2802

Table 4.2.2: Registration Result: Source Point cloud 2016 to Target Point cloud 2014

Method	Fitness	Max RMSE (m)	Min RMSE (m)	Avg RMSE (m)	Time (s)
RANSAC	0.9402	1.0963	0.00001844	0.08476	4931
FAST	0.8887	1.0554	0.0000181	0.8913	1294
RANSAC + ICP	0.4084	1.064	0.0000242	0.05913	4781

It is hard to tell which model is creating the best alignment. Initially observing the table 4.2.1 and 4.2.2, Global RANSAC + Local ICP is better with lower average RMSE of 0.0932 and 0.05913 for point clouds from 2015 and 2016 respectively. In addition, it is to be noted that, we cannot decide each method's performance just by looking at fitness value as we are working with point clouds having temporal changes, the fitness value will be lower naturally. FAST is the worst performing among all the methods as it does not take scale difference in account during the registration process as expected.

There are few things to consider before you can say that the model works well:

- Its not our goal to consider all the points as our inlier. This is due to the fact that some of the area has naturally changed a lot thus they should have more spatial difference than the threshold we used for registration.
- Its is always good to lower the inlier RMSE value. But the inlier should be the places with minimal temporal changes.

The figures 4.2.1 and 4.2.2 illustrate the distance distribution of each point of source point cloud to the nearest point of target point cloud after RANSAC + ICP based global registration for point clouds from the the year 2014 and 2015.

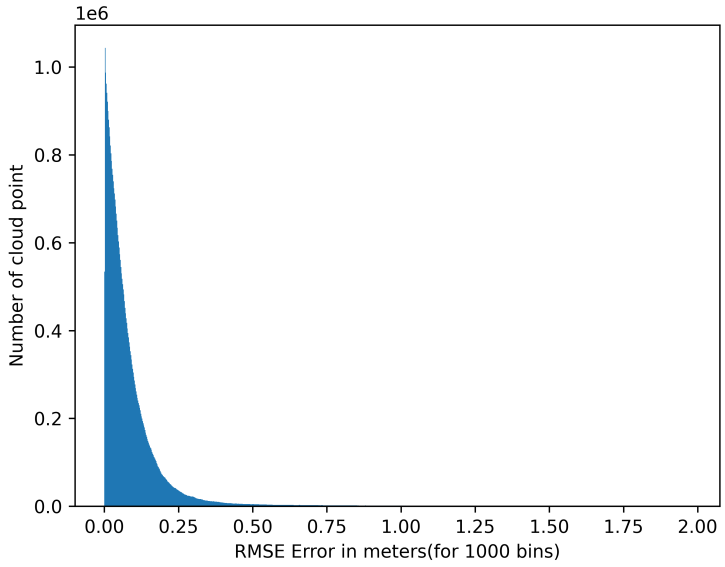


Figure 4.2.1: point cloud distance RMSE histogram for registration of point cloud 2015 to point cloud 2014(normal scale)

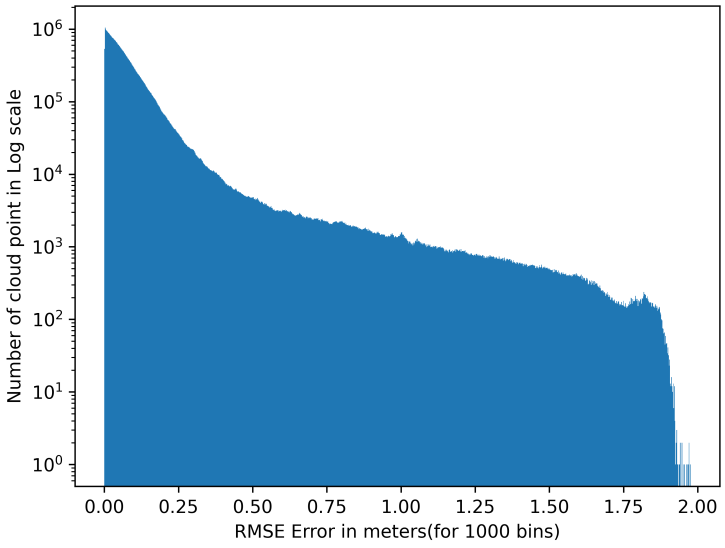


Figure 4.2.2: point cloud distance RMSE histogram for registration of point cloud 2015 to point cloud 2014(log scale)

Similarly, The figures 4.2.3 and 4.2.4 illustrate the distance distribution of each point of source point cloud to the nearest point of target point cloud after RANSAC + ICP based global registration for point clouds from the the year 2014 and 2016.

By looking at the histogram, particularly the log ones, we can see around 10000 points are not closely aligned having RMSE value more that 0.25 meters. In particular, we also observe that the histogram of 2016 is left shifted. That means, 2016 and 2014 point clouds are more closely aligned to each other compared to 2015 and 2014 ones. In addition, 2015 point cloud suffers from more misalignment with greater distance compared to 2016 point cloud (having bigger x axis values). This tells that 2016 and 2014 point cloud are more comparable and closely aligned that 2015 and 2014 ones.

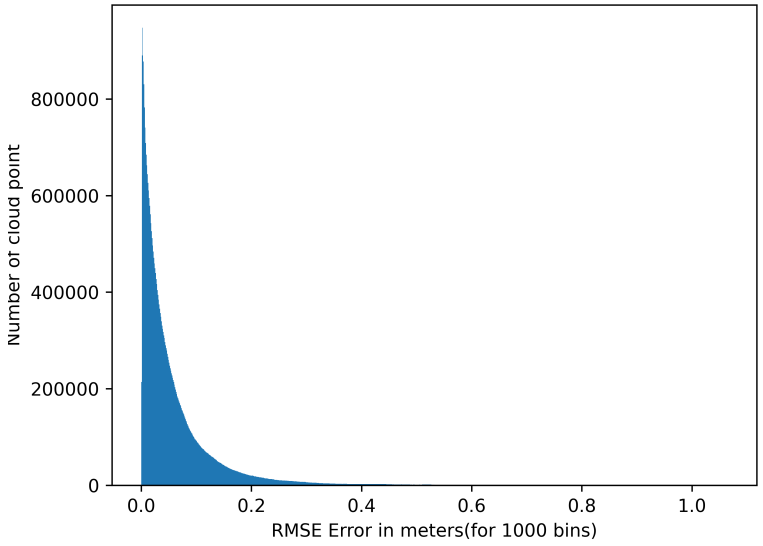


Figure 4.2.3: point cloud distance RMSE histogram for registration of point cloud 2016 to point cloud 2014(normal scale)

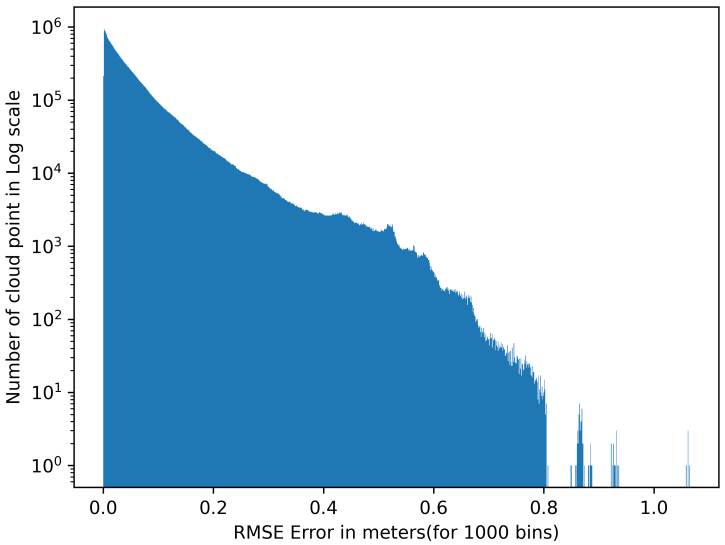


Figure 4.2.4: point cloud distance RMSE histogram for registration of point cloud 2016 to point cloud 2014(log scale)

We have shown the histogram for RANSAC + ICP registration method in this section. Now we will see, how many inliers and outliers are there for a threshold value of 0.05 RMSE based on distance between closet point between two point clouds.

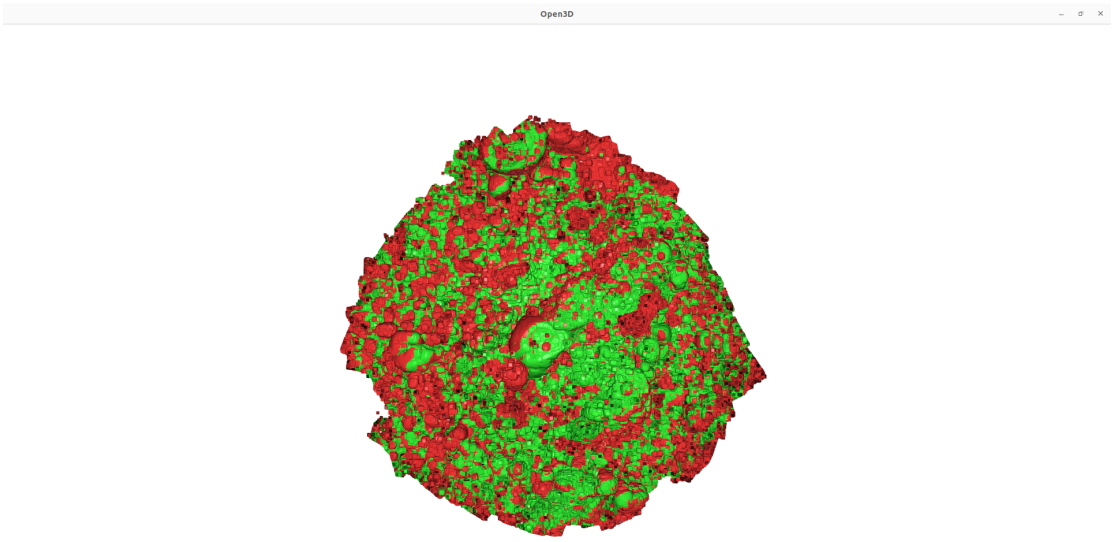


Figure 4.2.5: Inlier and outlier spatial distribution for point cloud 2015 and point cloud 2014 registration using RANSAC + ICP. (Green: Inliers, Red: Outliers)

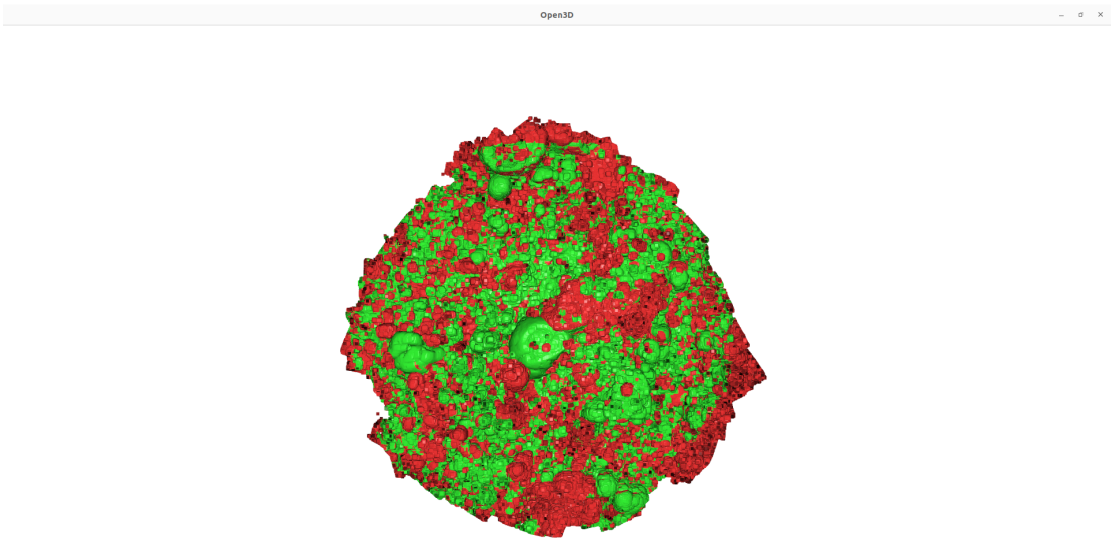


Figure 4.2.6: Inlier and outlier spatial distribution for point cloud 2016 and point cloud 2014 registration using RANSAC + ICP. (Green: Inliers, Red: Outliers)

4.2.1 Chunk Registration

In this section we will illustrate why per chunk registration outperforms RANSAC + ICP registration which is still showing best result up to this point. To start, lets talk about criteria we can best use to compare which method produces best point cloud alignment. One way to decide the accuracy of point cloud alignment is to look at how close robust correspondence are in 3D space. In this part we will also show how per chunk registration outperforms other methods. To do so, we took 5 cameras spatially distributed homogeneously and compare how many robust features (point correspondence) they have and if the RMSE is less than the threshold (10 cm) or not. The following table shows such results.

Table 4.2.3: Registration Result: Per chunk registration vs RANSAC+ICP registration

Chunk ID	**Method**	**Avg RMSE (m)**	**N Correspondence(Corr)**	**Corr. diff (m)**
Chunk one	Chunk	0.04026	116	0.0305
Chunk one	RANSAC+ICP	0.1405	54	0.0577
Chunk two	Chunk	0.0891	31	0.0568
Chunk two	RANSAC+ICP	0.0999	44	0.0685
Chunk three	Chunk	0.0791	37	0.0677
Chunk three	RANSAC+ICP	0.0791	26	0.0822
Chunk four	Chunk	0.0791	25	0.0626
Chunk four	RANSAC+ICP	0.1379	23	0.9982
Chunk five	Chunk	0.0791	23	0.0975
Chunk five	RANSAC+ICP	0.1015	28	0.0662

Finally, from table 4.2.3, we can clearly observe that in almost all cases we have much higher number of robust correspondence with lower difference between 3D representation of those points for the per chunk registration method (Green values). This confirms that, to create a dataset which can provide robust correspondence between multiyear data, per chunk based registration is the way to go.

CHAPTER

FIVE

DISCUSSION AND CONCLUSIONS

5.1 Discussion and Conclusion

In conclusion, this thesis has addressed the challenges of creatin underwater geo-localization in the presence of significant temporal changes. Through our exploration of data collection and pre-processing techniques, we have highlighted the importance of obtaining accurate and aligned datasets for robust underwater geo-localization models. Our robust feature matching approach has demonstrated promising results in establishing correspondences between images captured at different times, providing a foundation for precise geo-localization in dynamic underwater environments.

The evaluation of different point cloud registration methods has led to the identification of per-chunk RANSAC+ICP as a superior technique for achieving higher correspondence accuracy. This method has proven effective in mitigating the effects of temporal changes and improving the reliability of underwater geo-localization models. However, further research is needed to optimize and refine the registration process for more complex underwater scenes.

Overall, this thesis contributes to the advancement of underwater geo-localization methodologies by addressing the challenges posed by significant temporal changes. It provides valuable insights into data collection, pre-processing, feature matching, and point cloud registration techniques specific to underwater environments.

5.2 Future work

There are several avenues for future research and improvement in the field of underwater geo-localization. In this section, we outline some potential areas that can be explored to further enhance the accuracy and robustness of underwater geo-localization systems:

- Integration with State-of-the-Art Systems: An important future direction is to evaluate our dataset and methodologies using state-of-the-art underwater geo-localization systems. By comparing our results with established systems,

we can validate the effectiveness of our approach and identify areas for further improvement. However, due to the limited availability of such systems, we propose creating a similar setup as an underwater node and adapting it for our dataset. This would involve modifications and adaptations specific to underwater conditions. We plan to undertake this as a summer project this year to bridge the gap between our research and existing underwater geo-localization systems.

- Collaboration with Biologists: Collaborating with marine biologists can provide valuable insights into species behavior and their changing patterns in underwater environments. By incorporating biological knowledge into the data curation process, we can create more accurate and informative datasets. This collaboration can help refine our understanding of underwater ecosystems and contribute to the development of comprehensive underwater geo-localization models. In particular, species identification and tracking can be integrated into the process of mask creation for feature extraction.

- Validation on Different Datasets: To further validate the effectiveness of our methodologies, it would be beneficial to test our approach on different datasets captured from various underwater environments. This would involve collecting additional data from different locations with diverse conditions and underwater scenes. Comparing the performance of our methods across different datasets can provide insights into the generalizability and adaptability of our approach.

- Comparison with Ground-Based Data: In addition to validating our methods on different underwater datasets, it would be valuable to compare our results with ground-based data. This would involve collecting data from above-water or near-water perspectives using techniques such as aerial imagery or ground-based lidar. Comparing the geo-localization results from both underwater and above-water perspectives can provide a comprehensive understanding of the challenges and limitations specific to underwater geo-localization. It can also help identify potential areas of improvement and inspire the development of integrated multi-modal geo-localization systems.

- These future research directions have the potential to advance the field of underwater geo-localization and contribute to its practical applications in various domains such as marine research, underwater exploration, and underwater asset management. By addressing these areas, we can further enhance the accuracy, robustness, and applicability of underwater geo-localization models, ultimately facilitating a deeper understanding of underwater environments and their dynamic nature.

As we conclude this thesis, we recognize that the field of underwater geo-localization is dynamic and ever-evolving. Our work lays the groundwork for future studies to build upon, with the aim of achieving even higher accuracy and robustness in underwater geo-localization models. We hope that our findings contribute to the broader body of knowledge in this field and inspire researchers to continue pushing the boundaries of underwater geo-localization capabilities

REFERENCES

[1] Anthony T Fragoso et al. "A seasonally invariant deep transform for visual terrain-relative navigation". In: *Science Robotics* 6.55 (2021), eabf3320.

[2] B.S. Reddy and B.N. Chatterji. "An FFT-based technique for translation, rotation, and scale-invariant image registration". In: *IEEE Transactions on Image Processing* 5.8 (1996), pp. 1266–1271. DOI: 10.1109/83.506761.

[3] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[4] Martin A Fischler and Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[5] Yuxin Tian et al. "Cross-time and orientation-invariant overhead image geolocalization using deep local features". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2020, pp. 2512–2520.

[6] Royston Rodrigues and Masahiro Tani. "Are these from the same place? seeing the unseen in cross-view image geo-localization". In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 3753–3761.

[7] Torsten Sattler et al. "Understanding the limitations of cnn-based absolute camera pose regression". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 3302–3312.

[8] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. "Fast global registration". In: *European conference on computer vision*. Springer. 2016, pp. 766–782.

[9] Yang Chen and Gérard Medioni. "Object modelling by registration of multiple range images". In: *Image and Vision Computing* 10.3 (1992). Range Image Understanding, pp. 145–155. ISSN: 0262-8856. DOI: https://doi.org/10.1016/0262-8856(92)90066-C. URL: https://www.sciencedirect.com/science/article/pii/026288569290066C.

[10] Philippe Babin, Philippe Giguère, and François Pomerleau. "Analysis of Robust Functions for Registration Algorithms". In: *2019 International Conference on Robotics and Automation (ICRA)*. 2019, pp. 1451–1457. DOI: 10.1109/ICRA.2019.8793791.

[11] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[12] Ian Goodfellow et al. "Generative Adversarial Networks". In: *Commun. ACM* 63.11 (Oct. 2020), pp. 139–144. ISSN: 0001-0782. DOI: 10.1145/3422622. URL: https://doi.org/10.1145/3422622.

[13] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[14] Oscar Pizarro et al. "A simple, fast, and repeatable survey method for underwater visual 3D benthic mapping and monitoring". In: *Ecology and Evolution* 7.6 (2017), pp. 1770–1782.

[15] Radu Bogdan Rusu. "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments". PhD thesis. Computer Science department, Technische Universitaet Muenchen, Germany, Oct. 2009.

APPENDICES

All the codes can be found here:

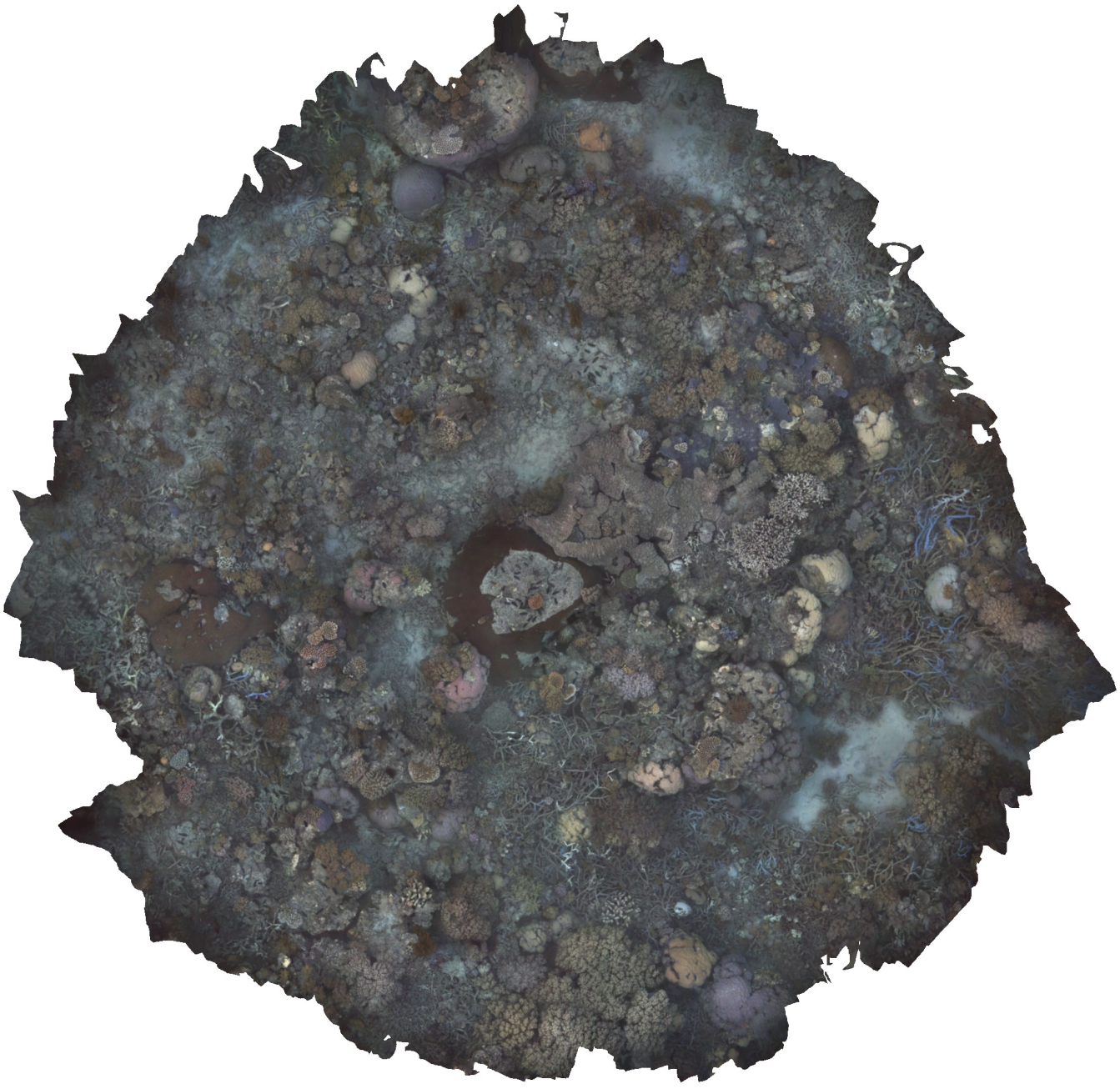
Github repository link

- `https://github.com/ashiqulalamkhan/long_term_underwater_vision`

Agisoft Metashape

Processing Report

21 June 2023



Survey Data

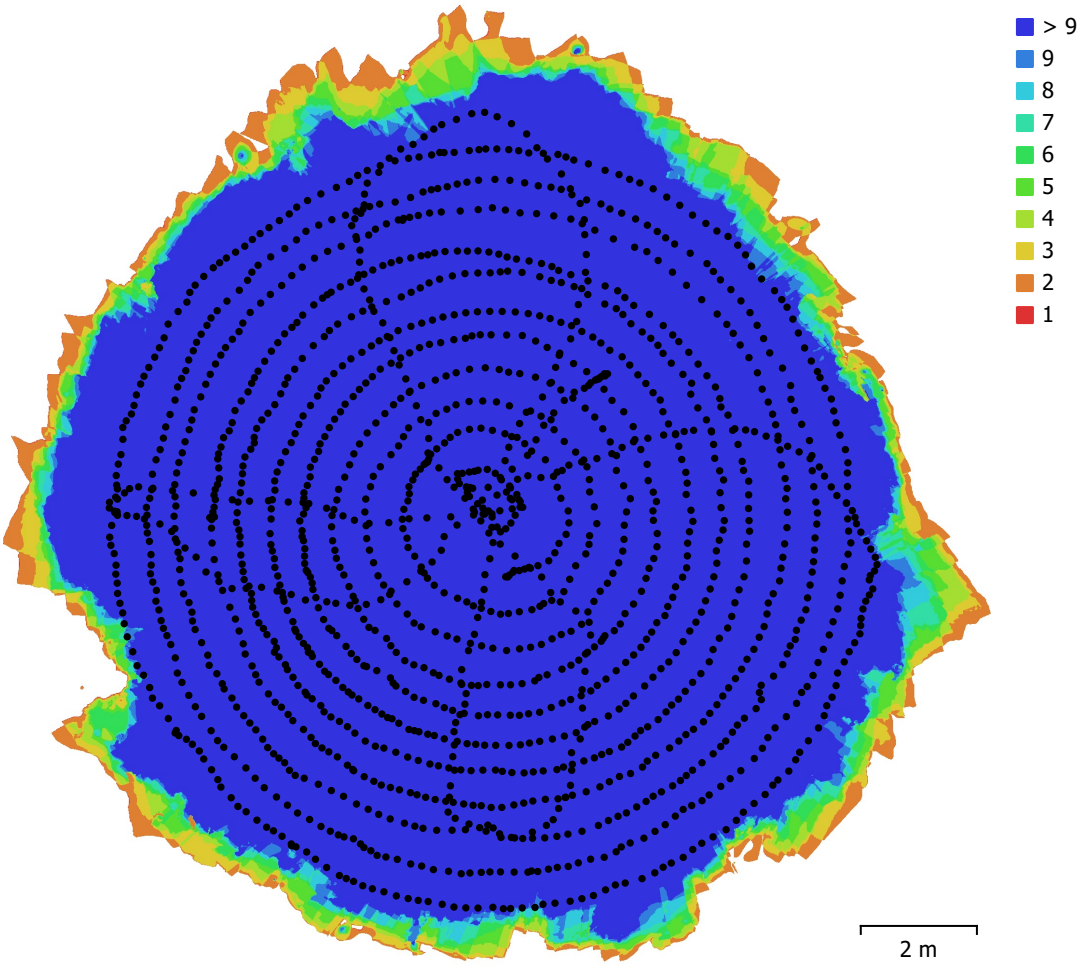


Fig. 1. Camera locations and image overlap.

Number of images:	1,600	Camera stations:	1,600
Flying altitude:	2.46 m	Tie points:	984,417
Ground resolution:	1.55 mm/pix	Projections:	3,917,915
Coverage area:	189 m²	Reprojection error:	0.56 pix

Camera Model	Resolution	Focal Length	Pixel Size	Precalibrated
unknown	1360 x 1024	10.67 mm	6.45 x 6.45 µm	Yes

Table 1. Cameras.

Camera Calibration

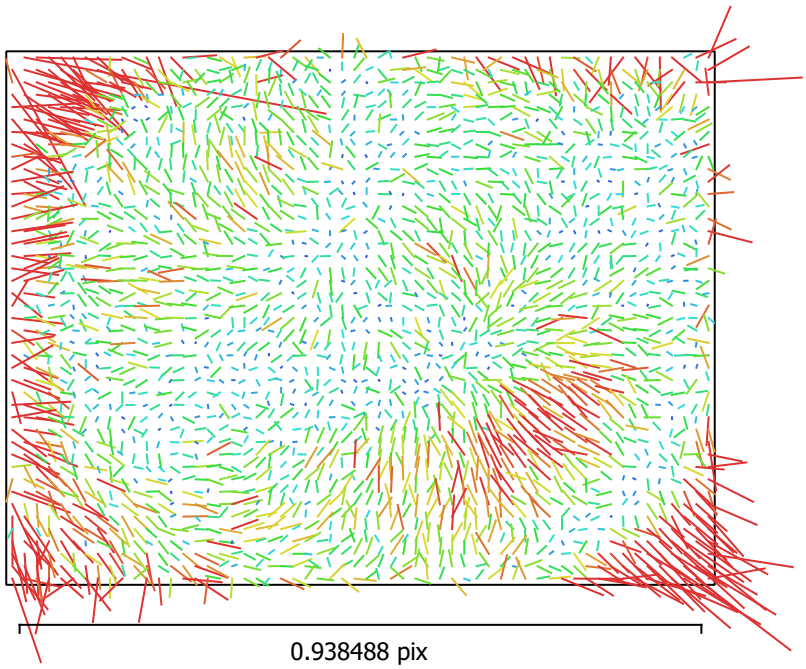


Fig. 2. Image residuals for unknown.

unknown

1600 images, precalibrated

Type	Resolution	Focal Length	Pixel Size
Frame	**1360 x 1024**	**10.67 mm**	**6.45 x 6.45 µm**

	Value	Error	F	Cx	Cy	B1	B2	K1	K2	K3	K4	P1	P2
F	**1718.02**	0.056	1.00	0.04	-0.44	-0.08	-0.06	-0.09	0.17	-0.15	0.15	0.02	-0.39
Cx	**3.88789**	0.021		1.00	0.02	-0.05	0.07	0.01	-0.00	0.00	0.00	0.93	0.03
Cy	**28.8893**	0.025			1.00	-0.02	-0.12	-0.02	-0.03	0.03	-0.05	0.02	0.93
B1	**0.78335**	0.0048				1.00	-0.01	0.00	-0.01	0.01	-0.00	-0.06	0.04
B2	**-0.0854703**	0.0048					1.00	-0.01	0.01	-0.01	0.01	-0.00	-0.12
K1	**0.158685**	0.00034						1.00	-0.97	0.93	-0.88	0.00	-0.02
K2	**0.75509**	0.0065							1.00	-0.99	0.96	-0.00	-0.02
K3	**-0.858679**	0.05								1.00	-0.99	0.00	0.02
K4	**4.14514**	0.13									1.00	-0.00	-0.03
P1	**-0.000199402**	8.5e-06										1.00	0.03
P2	**-0.00175075**	9.4e-06											1.00

Table 2. Calibration coefficients and correlation matrix.

Camera Locations

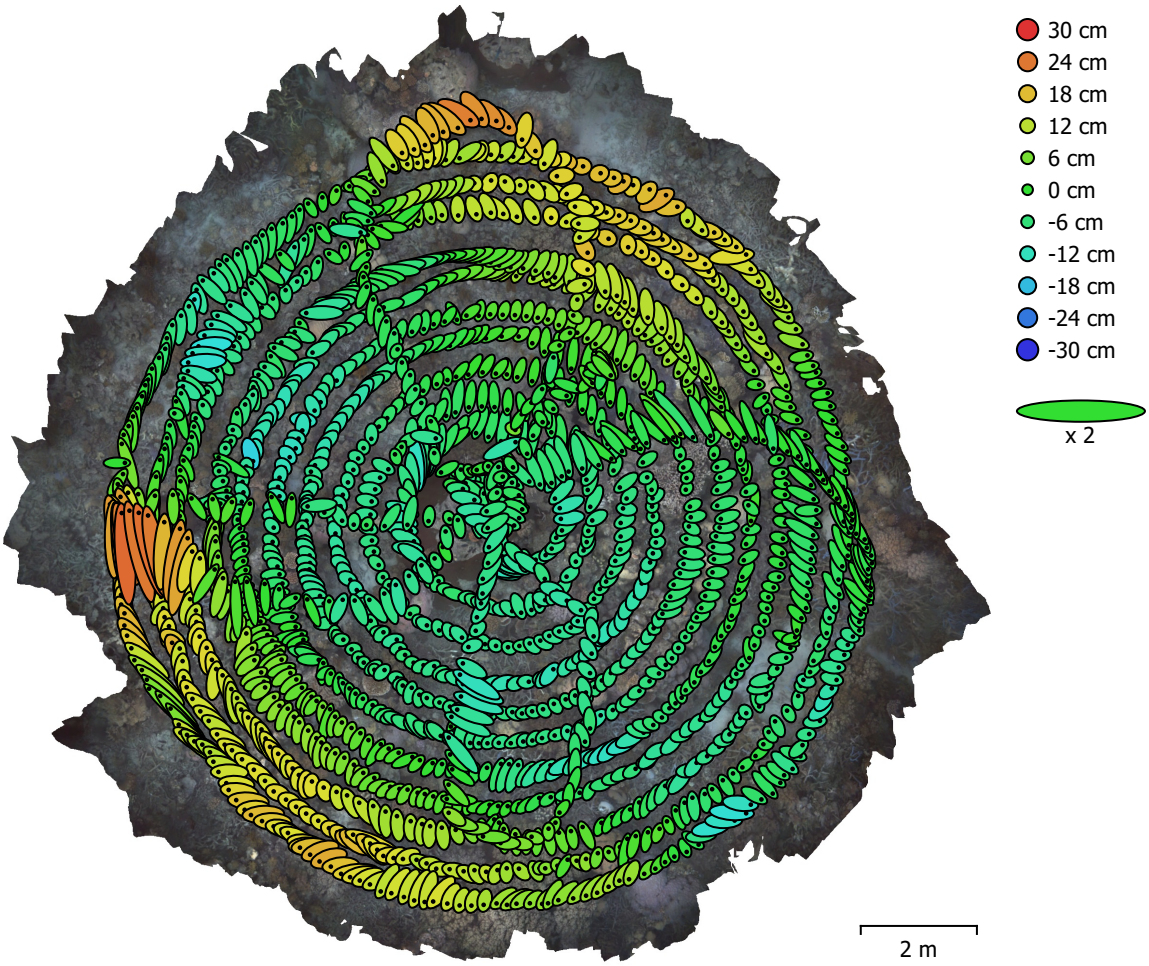


Fig. 3. Camera locations and error estimates.
Z error is represented by ellipse color. X,Y errors are represented by ellipse shape.
Estimated camera locations are marked with a black dot.

X error (cm)	Y error (cm)	Z error (cm)	XY error (cm)	Total error (cm)
9.30188	12.5802	8.32054	15.6456	17.7205

Table 3. Average camera location error.
X - Longitude, Y - Latitude, Z - Altitude.

Camera Orientations

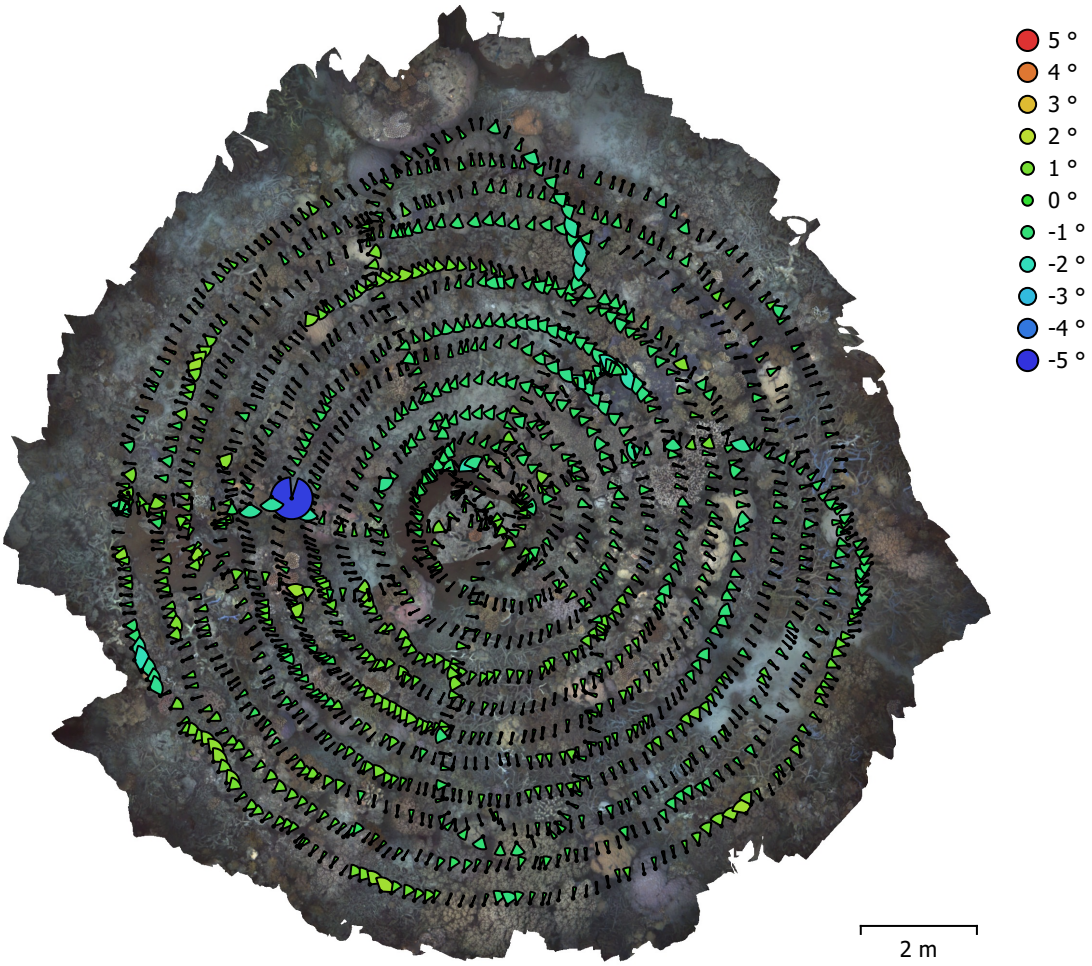


Fig. 4. Camera orientations and error estimates.
Arcs represent yaw error estimates.

Yaw error (°)	Pitch error (°)	Roll error (°)	Total error (°)
0.541962	1.88169	1.96362	2.77313

Table 4. Average camera rotation error.

Digital Elevation Model

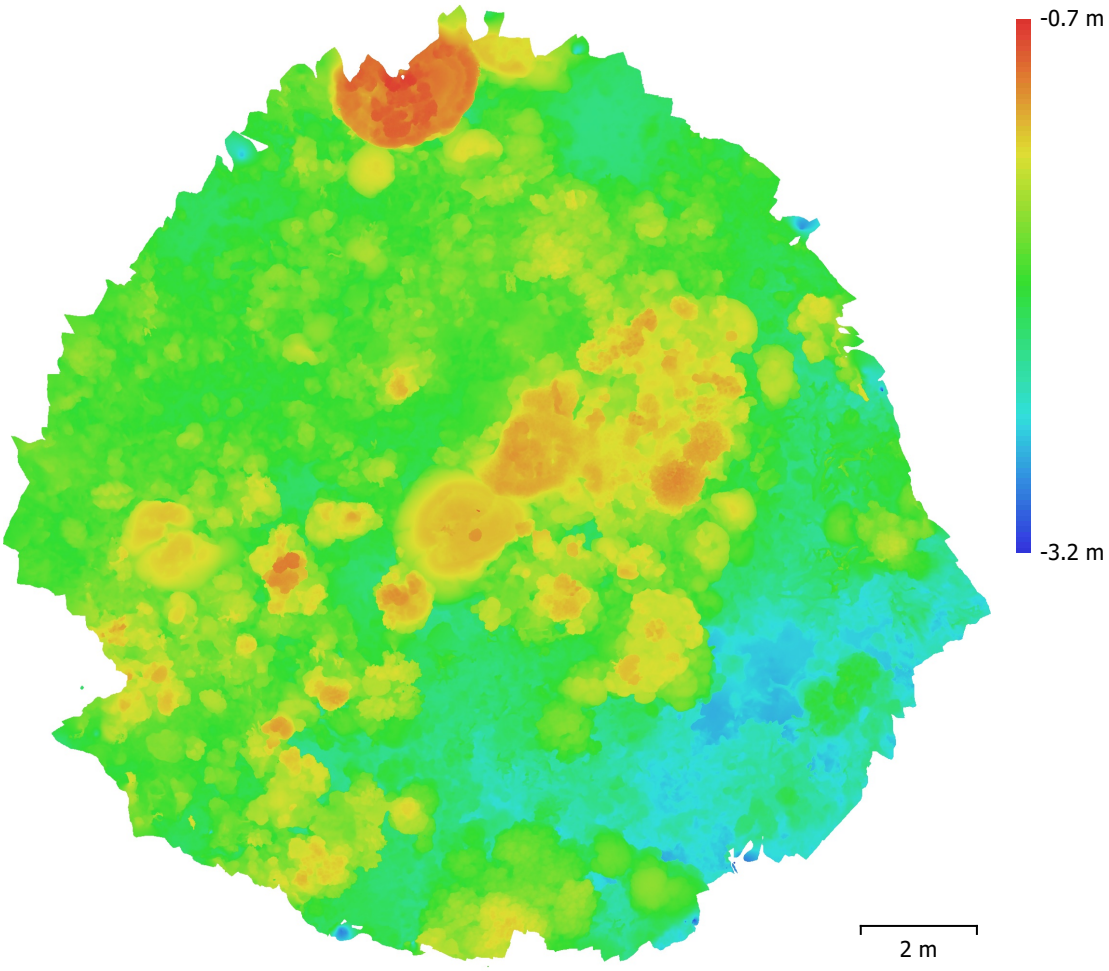


Fig. 5. Reconstructed digital elevation model.

Resolution:	3.09 mm/pix
Point density:	10.5 points/cm²

Processing Parameters

General	
Cameras	1600
Aligned cameras	1600
Coordinate system	WGS 84 (EPSG::4326)
Rotation angles	Yaw, Pitch, Roll
Tie Points	
Points	984,417 of 1,204,101
RMS reprojection error	0.143 (0.560489 pix)
Max reprojection error	0.73999 (15.4653 pix)
Mean key point size	3.6448 pix
Point colors	3 bands, uint8
Key points	No
Average tie point multiplicity	5.29197
Alignment parameters	
Accuracy	High
Generic preselection	Yes
Reference preselection	Yes
Key point limit	80,000
Tie point limit	4,000
Adaptive camera model fitting	Yes
Matching time	1 hours 15 minutes
Alignment time	1 hours 43 minutes
Optimization parameters	
Parameters	f, b1, b2, cx, cy, k1-k4, p1, p2
Optimization time	49 seconds
File size	123.42 MB
Point Cloud	
Points	39,712,554
Point attributes	
Color	3 bands, uint8
Normal	
Point classes	
Created (never classified)	39,712,554
Depth maps generation parameters	
Quality	High
Filtering mode	Mild
Processing time	4 hours 54 minutes
Point cloud generation parameters	
Processing time	1 hours 9 minutes
File size	537.00 MB
DEM	
Size	7,218 x 7,219
Coordinate system	WGS 84 (EPSG::4326)
Reconstruction parameters	
Source data	Point cloud
Interpolation	Enabled
Processing time	1 minutes 54 seconds
File size	95.68 MB
Orthomosaic	
Size	8,497 x 8,276

Coordinate system	WGS 84 (EPSG::4326)
Colors	3 bands, uint8
Reconstruction parameters	
Blending mode	Mosaic
Surface	DEM
Enable color correction	No
Enable hole filling	Yes
Processing time	6 minutes 23 seconds
File size	885.28 MB
System	
Software name	Agisoft Metashape Professional
Software version	2.0.2 build 16334
OS	Windows 64 bit
RAM	15.31 GB
CPU	AMD Ryzen 7 5800U with Radeon Graphics
GPU(s)	None

Agisoft Metashape

Processing Report

21 June 2023

Survey Data

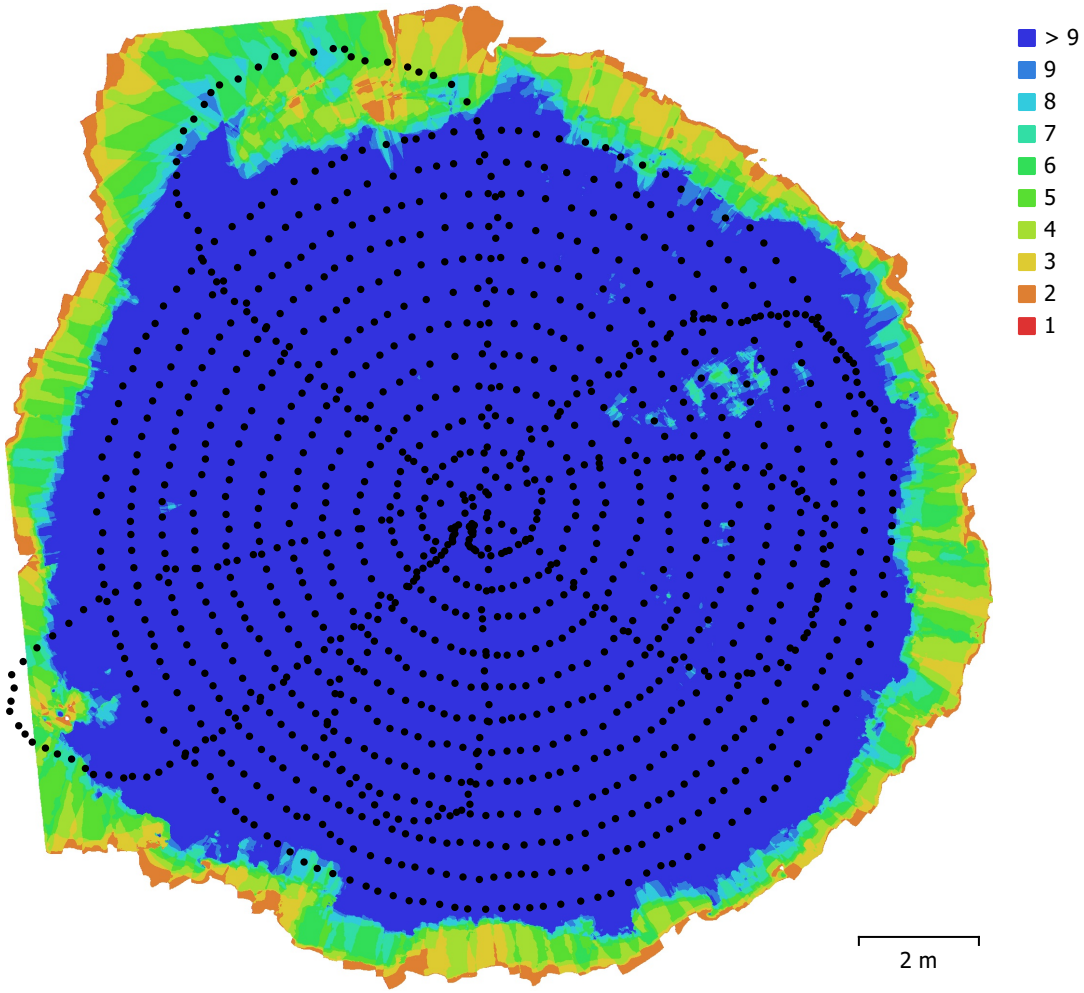


Fig. 1. Camera locations and image overlap.

Number of images:	1,234	Camera stations:	1,233
Flying altitude:	2.41 m	Tie points:	1,075,143
Ground resolution:	1.49 mm/pix	Projections:	3,887,739
Coverage area:	206 m²	Reprojection error:	0.501 pix

Camera Model	Resolution	Focal Length	Pixel Size	Precalibrated
unknown	1360 x 1024	10.67 mm	6.45 x 6.45 µm	Yes

Table 1. Cameras.

Camera Calibration

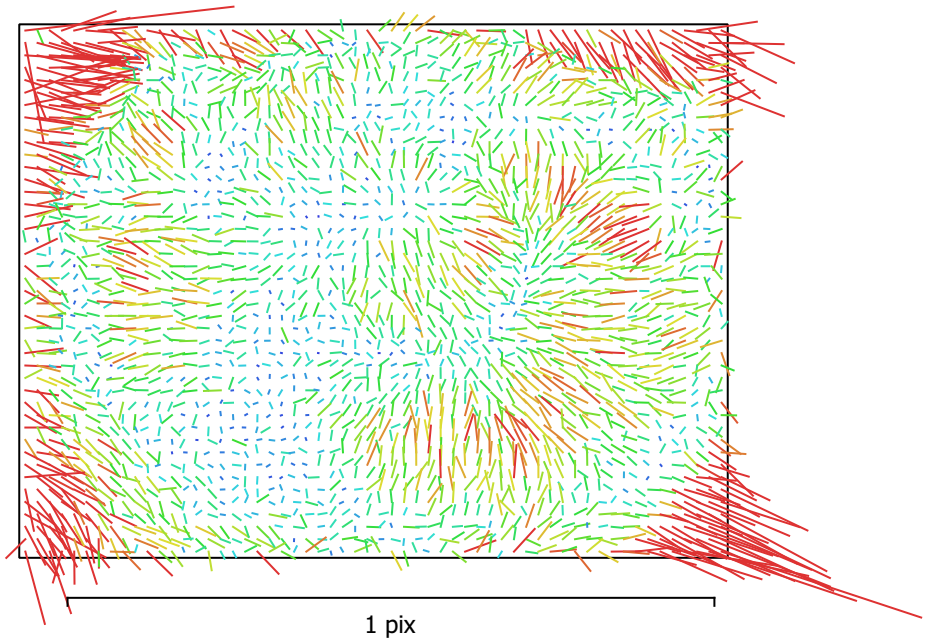


Fig. 2. Image residuals for unknown.

unknown

1234 images, precalibrated

Type	Resolution	Focal Length	Pixel Size
Frame	**1360 x 1024**	**10.67 mm**	**6.45 x 6.45 µm**

	Value	Error	F	Cx	Cy	B1	B2	K1	K2	K3	K4	P1	P2
F	**1713.34**	0.079	1.00	-0.06	-0.20	-0.03	0.03	0.01	0.13	-0.10	0.10	-0.08	-0.18
Cx	**22.6097**	0.021		1.00	0.02	-0.04	0.03	-0.00	-0.02	0.03	-0.04	0.93	0.01
Cy	**25.0146**	0.022			1.00	-0.00	-0.10	-0.01	-0.03	0.03	-0.03	0.01	0.93
B1	**0.796136**	0.0033				1.00	0.01	0.01	-0.02	0.01	-0.00	-0.07	0.03
B2	**0.00747811**	0.0033					1.00	0.00	0.00	-0.00	0.00	-0.01	-0.13
K1	**0.159102**	0.00032						1.00	-0.96	0.92	-0.87	-0.00	-0.01
K2	**0.666182**	0.0059							1.00	-0.99	0.96	-0.02	-0.02
K3	**-0.340271**	0.044								1.00	-0.99	0.03	0.02
K4	**3.07592**	0.11									1.00	-0.04	-0.03
P1	**-0.0019739**	8.4e-06										1.00	0.01
P2	**-0.00124616**	8.5e-06											1.00

Table 2. Calibration coefficients and correlation matrix.

Camera Locations

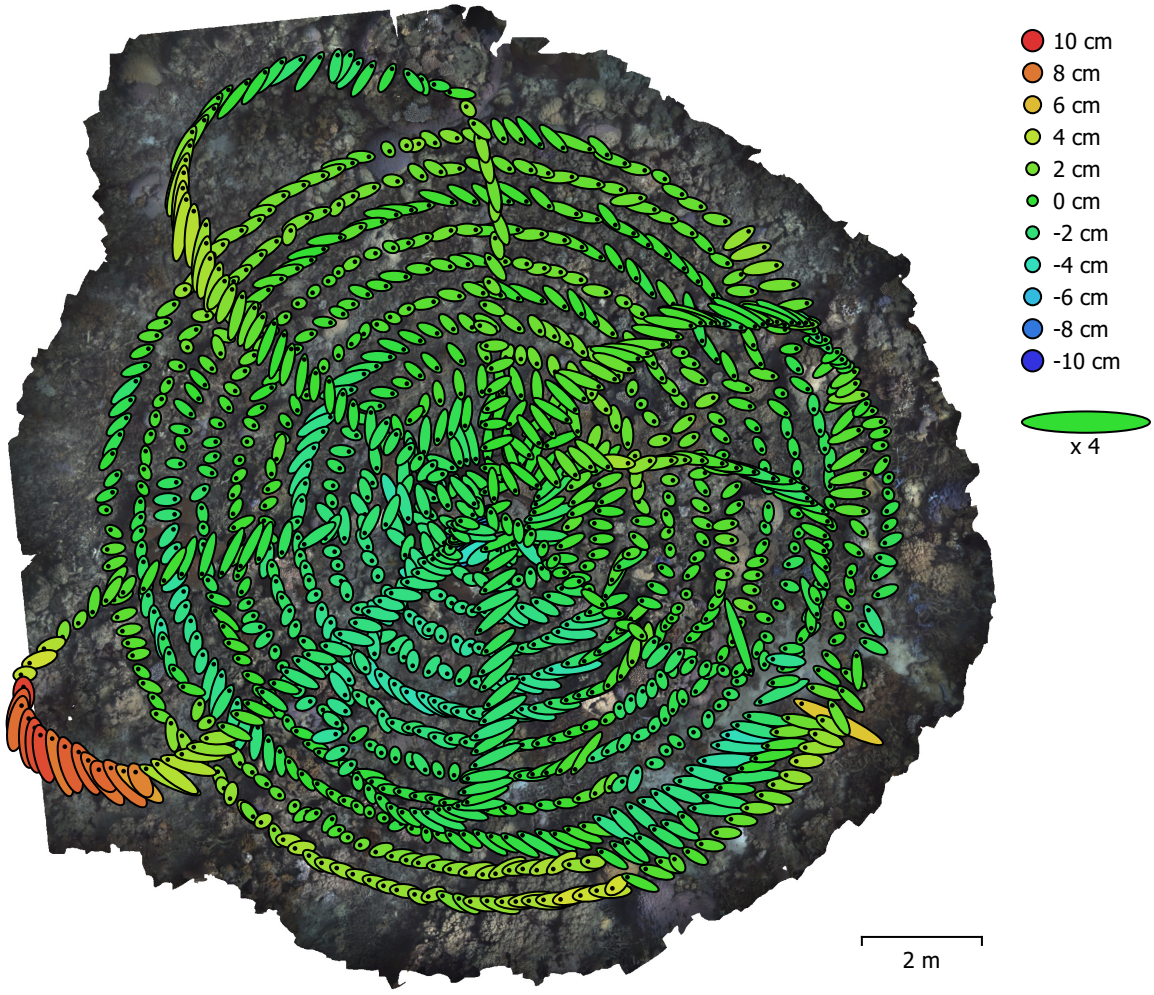


Fig. 3. Camera locations and error estimates.
Z error is represented by ellipse color. X,Y errors are represented by ellipse shape.
Estimated camera locations are marked with a black dot.

X error (cm)	Y error (cm)	Z error (cm)	XY error (cm)	Total error (cm)
7.34502	5.81798	1.95879	9.37007	9.57262

Table 3. Average camera location error.
X - Longitude, Y - Latitude, Z - Altitude.

Camera Orientations

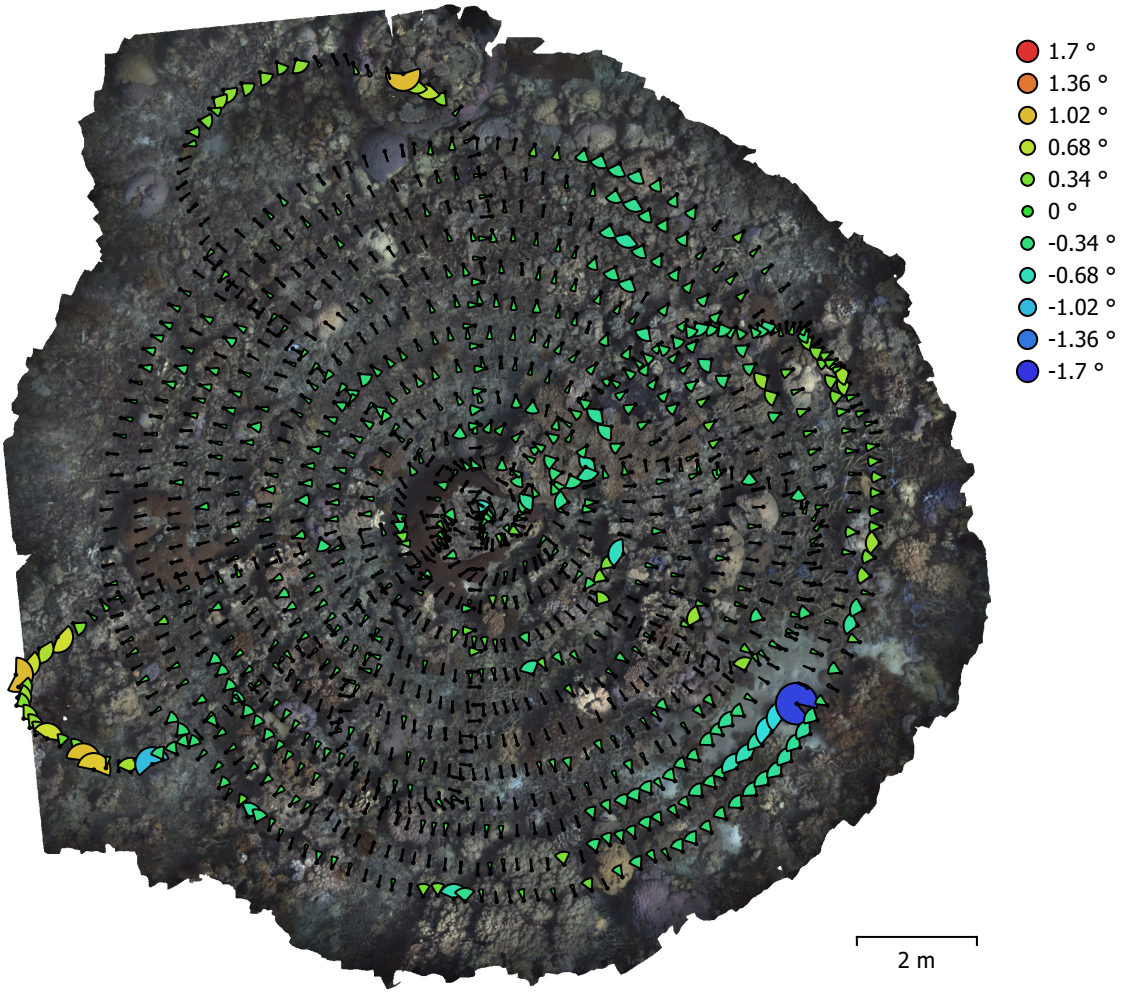


Fig. 4. Camera orientations and error estimates.
Arcs represent yaw error estimates.

Yaw error (°)	Pitch error (°)	Roll error (°)	Total error (°)
0.204265	0.644811	0.460692	0.818378

Table 4. Average camera rotation error.

Digital Elevation Model

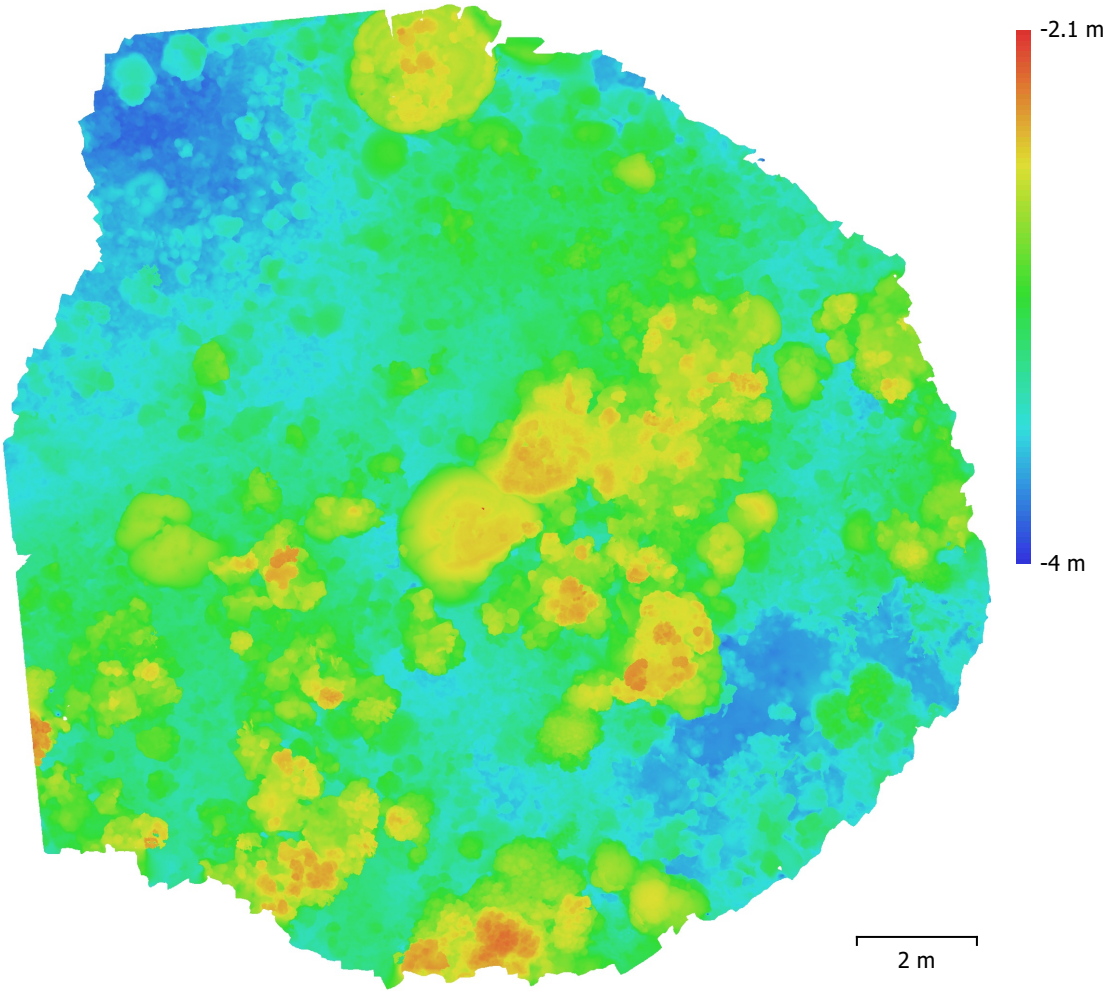


Fig. 5. Reconstructed digital elevation model.

Resolution:	2.98 mm/pix
Point density:	11.3 points/cm²

Processing Parameters

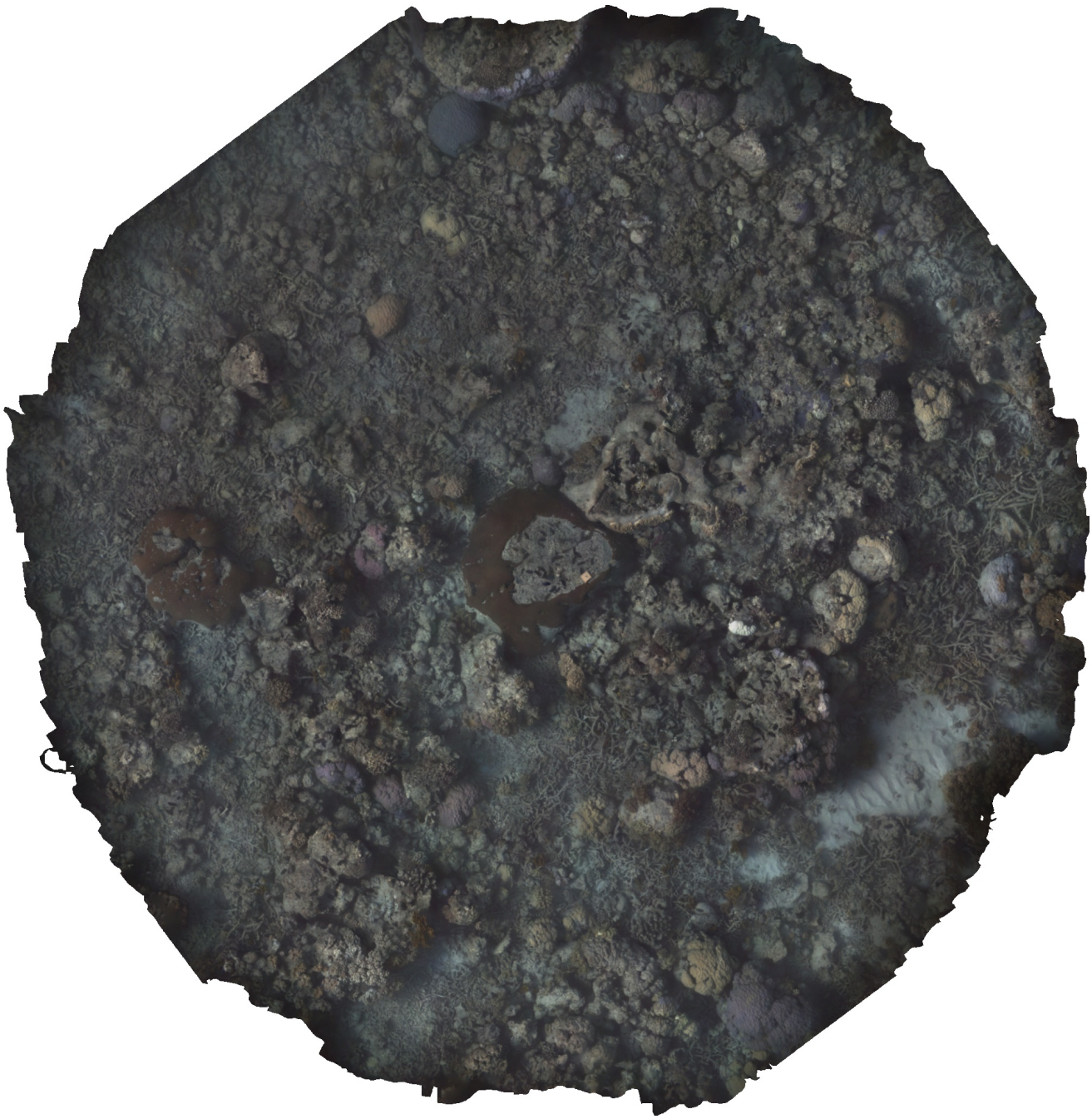
General	
Cameras	1234
Aligned cameras	1233
Coordinate system	WGS 84 (EPSG::4326)
Rotation angles	Yaw, Pitch, Roll
Tie Points	
Points	1,075,143 of 1,262,637
RMS reprojection error	0.133673 (0.500962 pix)
Max reprojection error	0.491792 (12.1894 pix)
Mean key point size	3.43269 pix
Point colors	3 bands, uint8
Key points	No
Average tie point multiplicity	4.30879
Alignment parameters	
Accuracy	High
Generic preselection	Yes
Reference preselection	Yes
Key point limit	80,000
Tie point limit	40,000
Adaptive camera model fitting	Yes
Matching time	47 minutes 48 seconds
Alignment time	1 hours 48 minutes
Optimization parameters	
Parameters	f, b1, b2, cx, cy, k1-k4, p1, p2
Optimization time	1 minutes 23 seconds
File size	111.40 MB
Point Cloud	
Points	42,312,793
Point attributes	
Color	3 bands, uint8
Normal	
Point classes	
Created (never classified)	42,312,793
Depth maps generation parameters	
Quality	High
Filtering mode	Mild
Processing time	6 hours 40 minutes
Point cloud generation parameters	
Processing time	31 minutes 46 seconds
File size	573.98 MB
DEM	
Size	5,944 x 6,151
Coordinate system	WGS 84 (EPSG::4326)
Reconstruction parameters	
Source data	Point cloud
Interpolation	Enabled
Processing time	1 minutes 32 seconds
File size	108.00 MB
Orthomosaic	
Size	8,221 x 8,195

Coordinate system	WGS 84 (EPSG::4326)
Colors	3 bands, uint8
Reconstruction parameters	
Blending mode	Mosaic
Surface	DEM
Enable color correction	No
Enable hole filling	Yes
Processing time	3 minutes 42 seconds
File size	792.27 MB
System	
Software name	Agisoft Metashape Professional
Software version	2.0.2 build 16334
OS	Windows 64 bit
RAM	15.31 GB
CPU	AMD Ryzen 7 5800U with Radeon Graphics
GPU(s)	None

Agisoft Metashape

Processing Report

21 June 2023



Survey Data

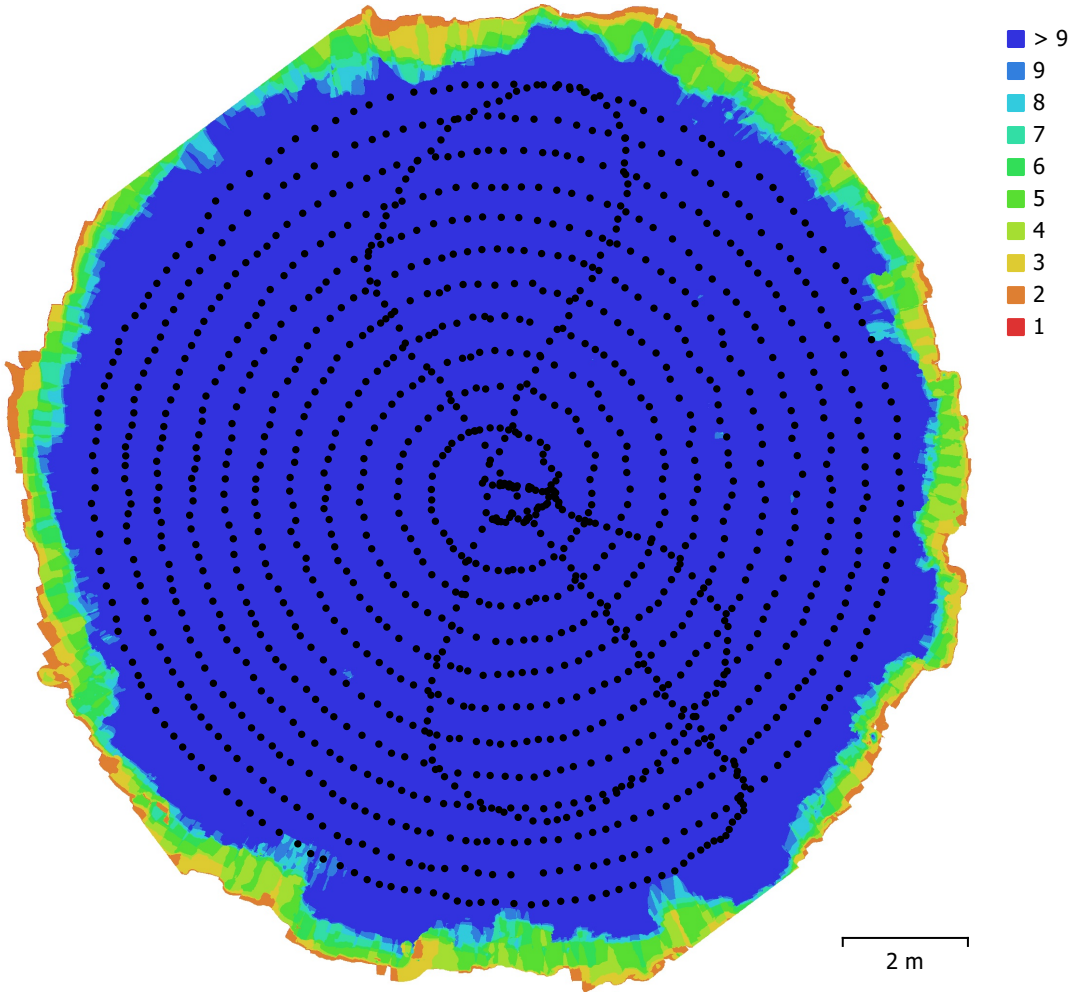


Fig. 1. Camera locations and image overlap.

Number of images:	1,348	Camera stations:	1,348
Flying altitude:	2.51 m	Tie points:	730,467
Ground resolution:	1.54 mm/pix	Projections:	3,078,298
Coverage area:	185 m²	Reprojection error:	0.57 pix

Camera Model	Resolution	Focal Length	Pixel Size	Precalibrated
unknown	1360 x 1024	10.67 mm	6.45 x 6.45 µm	Yes

Table 1. Cameras.

Camera Calibration

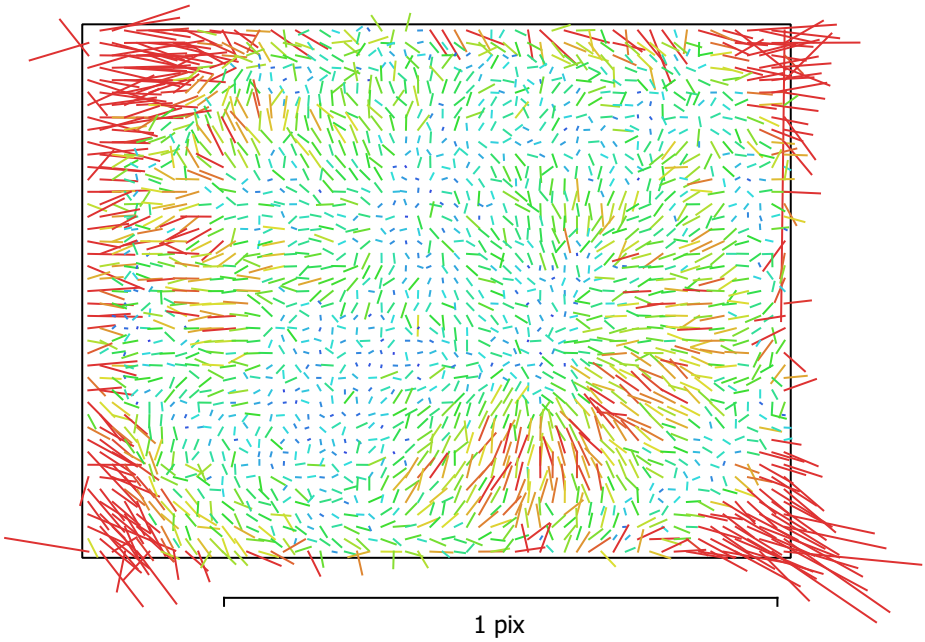


Fig. 2. Image residuals for unknown.

unknown

1348 images, precalibrated

Type	Resolution	Focal Length	Pixel Size
Frame	**1360 x 1024**	**10.67 mm**	**6.45 x 6.45 µm**

	Value	Error	F	Cx	Cy	B1	B2	K1	K2	K3	K4	P1	P2
F	**1714.41**	0.095	1.00	-0.19	-0.47	-0.03	0.04	0.00	0.13	-0.10	0.11	-0.18	-0.42
Cx	**22.6281**	0.024		1.00	0.15	0.01	0.09	-0.01	-0.02	0.02	-0.03	0.91	0.13
Cy	**25.8888**	0.028			1.00	-0.04	-0.10	-0.04	-0.03	0.02	-0.04	0.13	0.90
B1	**0.887803**	0.0053				1.00	-0.03	0.01	-0.01	0.01	-0.00	-0.04	0.02
B2	**0.035763**	0.0054					1.00	0.01	0.00	-0.00	0.00	-0.01	-0.19
K1	**0.157345**	0.00037						1.00	-0.96	0.92	-0.87	-0.02	-0.05
K2	**0.734164**	0.0071							1.00	-0.99	0.96	-0.02	-0.02
K3	**-0.788889**	0.054								1.00	-0.99	0.02	0.02
K4	**4.13651**	0.14									1.00	-0.03	-0.03
P1	**-0.00224485**	9.3e-06										1.00	0.15
P2	**-0.00130631**	1e-05											1.00

Table 2. Calibration coefficients and correlation matrix.

Camera Locations

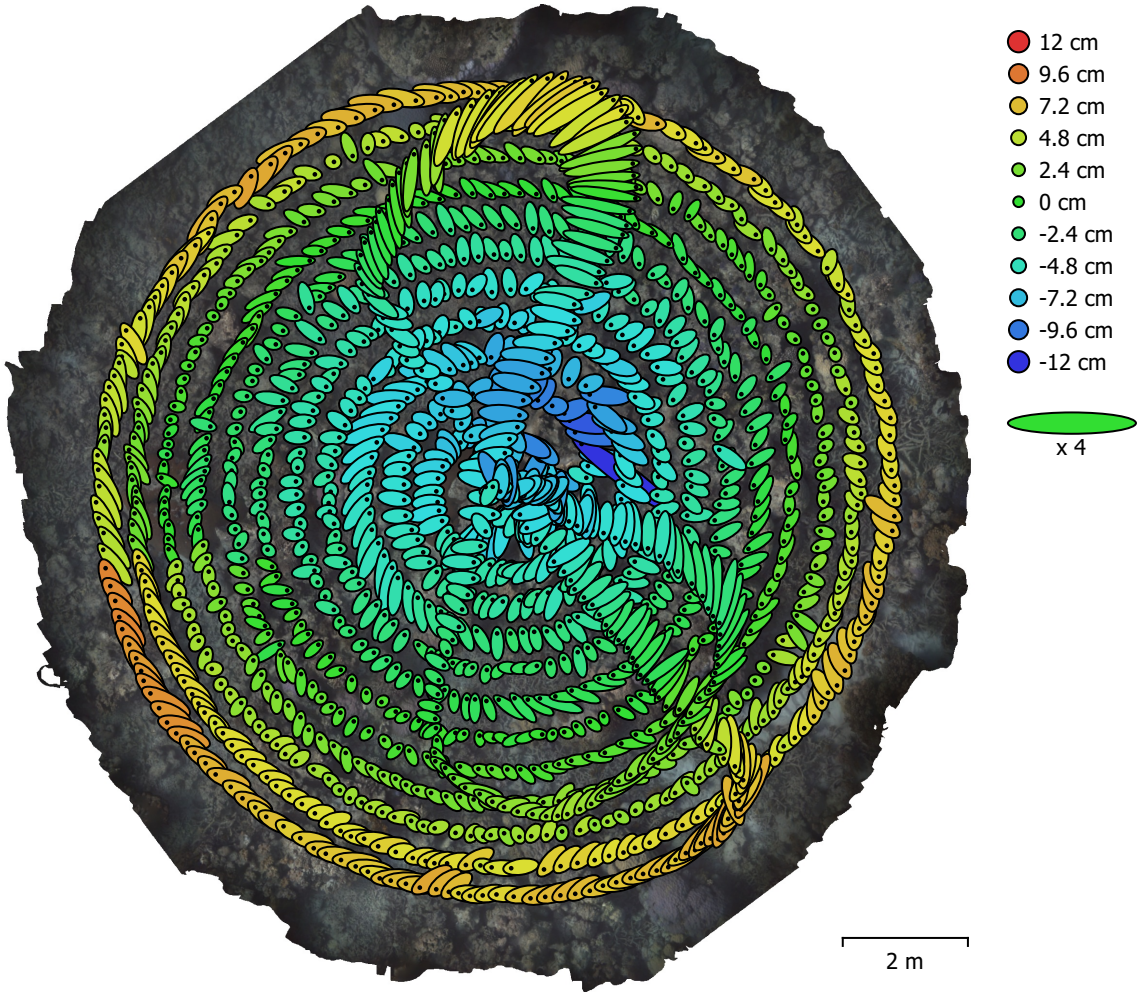


Fig. 3. Camera locations and error estimates.
Z error is represented by ellipse color. X,Y errors are represented by ellipse shape.
Estimated camera locations are marked with a black dot.

X error (cm)	Y error (cm)	Z error (cm)	XY error (cm)	Total error (cm)
6.37829	6.48373	4.50712	9.09513	10.1506

Table 3. Average camera location error.
X - Longitude, Y - Latitude, Z - Altitude.

Camera Orientations

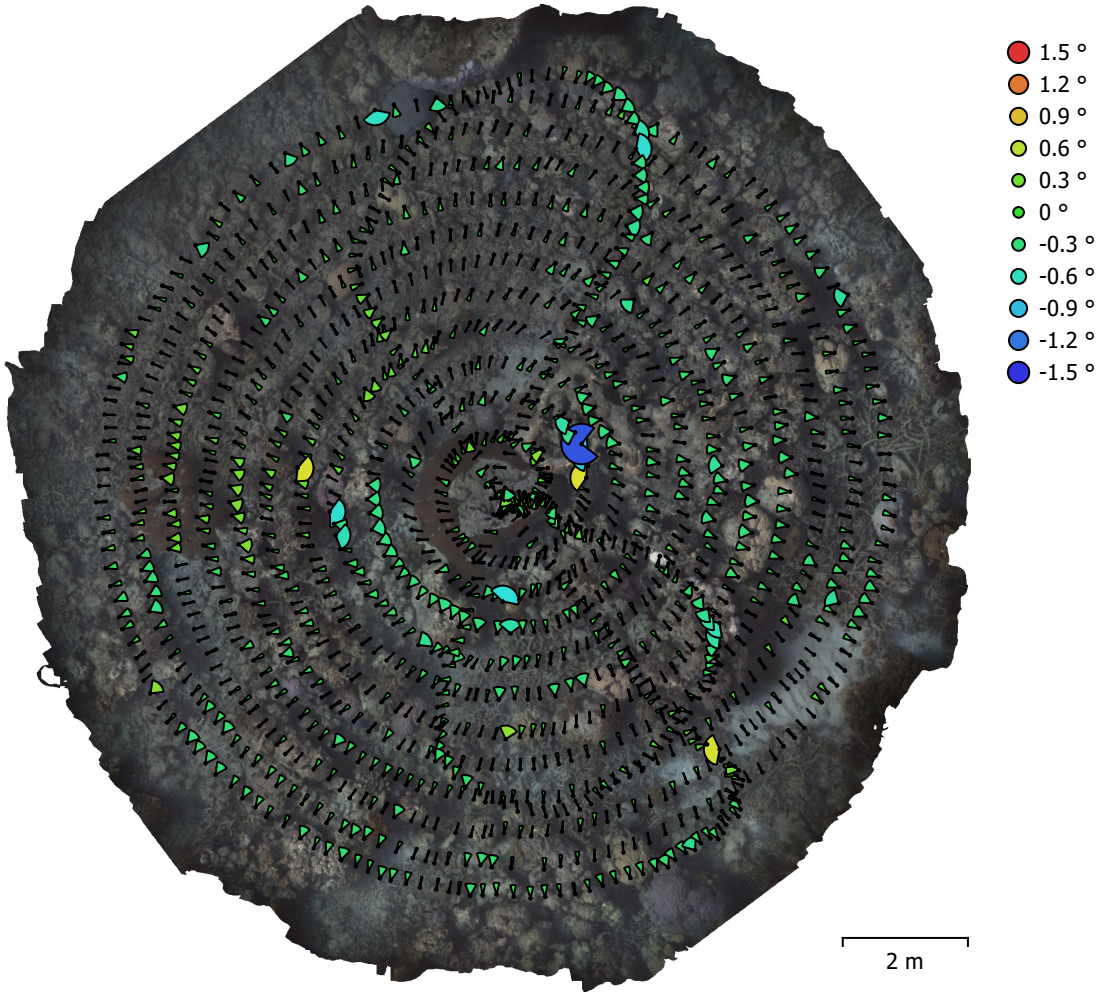


Fig. 4. Camera orientations and error estimates.
Arcs represent yaw error estimates.

Yaw error (°)	Pitch error (°)	Roll error (°)	Total error (°)
0.160236	0.966203	0.630548	1.16482

Table 4. Average camera rotation error.

Digital Elevation Model

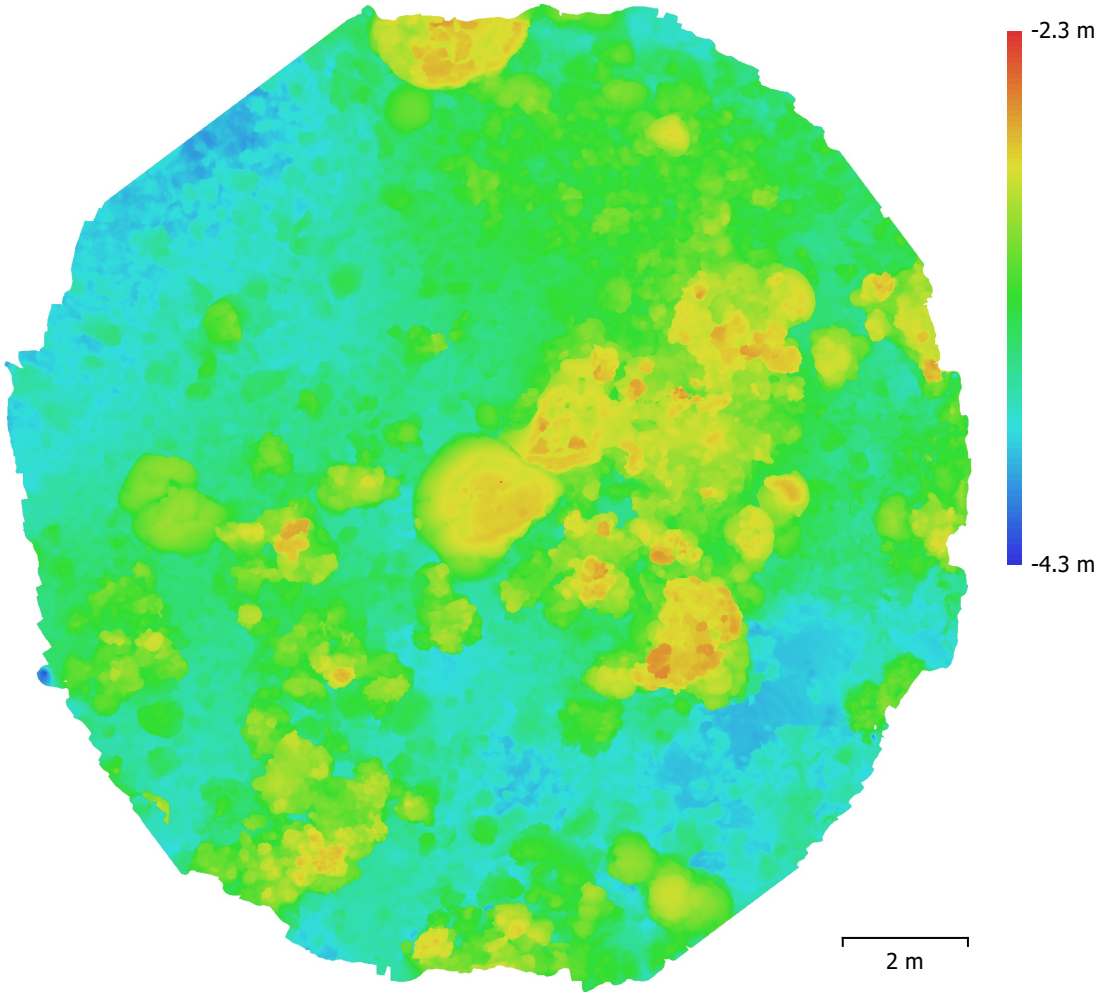


Fig. 5. Reconstructed digital elevation model.

Resolution:	3.07 mm/pix
Point density:	10.6 points/cm²

Processing Parameters

General	
Cameras	1348
Aligned cameras	1348
Coordinate system	WGS 84 (EPSG::4326)
Rotation angles	Yaw, Pitch, Roll
Tie Points	
Points	730,467 of 917,351
RMS reprojection error	0.143266 (0.569521 pix)
Max reprojection error	0.88457 (11.8382 pix)
Mean key point size	3.74281 pix
Point colors	3 bands, uint8
Key points	No
Average tie point multiplicity	5.60407
Alignment parameters	
Accuracy	High
Generic preselection	Yes
Reference preselection	Yes
Key point limit	100,000
Tie point limit	4,000
Adaptive camera model fitting	Yes
Matching time	52 minutes 32 seconds
Alignment time	2 hours 46 minutes
Optimization parameters	
Parameters	f, b1, b2, cx, cy, k1-k4, p1, p2
Optimization time	41 seconds
File size	97.96 MB
Point Cloud	
Points	36,352,159
Point attributes	
Color	3 bands, uint8
Normal	
Point classes	
Created (never classified)	36,352,159
Depth maps generation parameters	
Quality	High
Filtering mode	Mild
Processing time	6 hours 5 minutes
Point cloud generation parameters	
Processing time	46 minutes 3 seconds
File size	489.92 MB
DEM	
Size	6,954 x 6,938
Coordinate system	WGS 84 (EPSG::4326)
Reconstruction parameters	
Source data	Point cloud
Interpolation	Enabled
Processing time	1 minutes 37 seconds
File size	91.86 MB
Orthomosaic	
Size	7,715 x 7,885

Coordinate system	WGS 84 (EPSG::4326)
Colors	3 bands, uint8
Reconstruction parameters	
Blending mode	Mosaic
Surface	DEM
Enable color correction	No
Enable hole filling	Yes
Processing time	4 minutes 3 seconds
File size	760.82 MB
System	
Software name	Agisoft Metashape Professional
Software version	2.0.2 build 16334
OS	Windows 64 bit
RAM	15.31 GB
CPU	AMD Ryzen 7 5800U with Radeon Graphics
GPU(s)	None