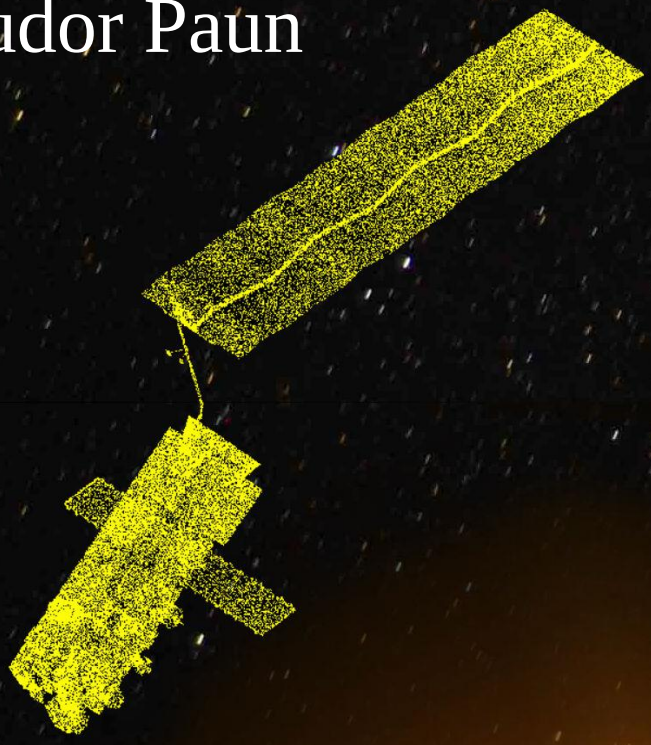
MULTI-VIEW ICP FOR 3D-RECONSTRUCTION OF UNKNOWN SPACE DEBRIS

Master of Science Thesis

Tudor Paun



22 September 2014

TU Delft

MULTI-VIEW ICP FOR 3D-RECONSTRUCTION OF UNKNOWN SPACE DEBRIS

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

AEROSPACE ENGINEERING

BY

TUDOR PAUN

ON

22 SEPT 2014

Graduation committee:
Prof. Dr. Eberhart Gill
Dr. Ir. Hans Kuiper
Dr. Ingo Ahrns
Dr. Ernst Schrama

Page intentionally left blank.

CONTENTS

CONTENTS III

ABSTRACT VII

ACKNOWLEDGEMENTS IX

NOMENCLATURE XI

1 INTRODUCTION 1

2 MISSION BACKGROUND 3

2. 1. HISTORY OF SPACE DEBRIS 3
2. 2. THE EADS APPROACH 4
2. 3. VISION SYSTEM 5
2. 4. TRAJECTORY 7

3 LIDAR SCANNER 9

3. 1. DESCRIPTION 9
3. 2. FUNCTION/WORKING PRINCIPLE 10
3. 3. DATA 12

4 SIMULATOR 17

4. 1. INTRODUCTION 17
4. 1. 1. GENERAL 17
4. 1. 2. OPTIONS 17
4. 2. SIMULATED DATA 18
4. 2. 1. REALISTICALLY AVAILABLE DATA 18
4. 2. 2. SIMULATION DATA 21
4. 3. CORRELATIONS BETWEEN THE SIMULATOR AND REALITY 22
4. 3. 1. SETUP 23
4. 3. 2. ERROR BUDGET 24

5 ITERATIVE CLOSESR POINT 25

5. 1. DESCRIPTION 25
5. 2. POINT CLOUD 28

5. 3. FINDING CORRESPONDENCES 29
5. 4. REGISTRATION 30
5. 5. ICP VARIATIONS 37
5. 5. 1. CORRESPONDENCE SEARCHES 37
5. 5. 2. NUMBER OF POINTS 37
5. 5. 3. WEIGHING 37
5. 5. 4. POINT CLOUD FLEXIBILITY 37

6 STATE OF THE ART 39

6. 1. DESCRIPTION 39
6. 2. SLAM 39
6. 3. ORTHOGONAL PROCRUSTES 41
6. 3. 1. CORRESPONDENCES 42
6. 3. 2. REGISTRATION 44
6. 4. RESULTS 45

7 GLOBAL ITERATIVE CLOSEST POINT 49

7. 1. DESCRIPTION 49
7. 2. OVERVIEW 51
7. 2. 1. LOOPS 52
7. 2. 2. PERFORMANCE 52
7. 2. 3. NAÏVE METHOD 53
7. 2. 4. CORRESPONDENCE MATRIX 53
7. 3. CORRESPONDENCE FINDING 55
7. 4. IMAGINARY POINT CLOUD 60
7. 5. REGISTRATION 64
7. 6. MODULES 66
7. 6. 1. POINT DELETION 66
7. 6. 2. KD TREE GENERATION 68
7. 6. 3. POINT REDUCTION 69
7. 6. 4. UPDATING VALUES 70

8 RECONSTRUCTION 73

8. 1. DIAGRAM 73
8. 2. MODEL RECONSTRUCTION 75
8. 2. 1. REFERENCE FRAMES 75
8. 2. 2. APPROACH 77
8. 3. TRAJECTORY RECONSTRUCTION 83

9 EVALUATION 85

9. 1. EVALUATION PROCEDURE 85

9. 2. ERROR COMPUTATION 86
9. 2. 1. PARAMETER SELECTION 86
9. 2. 2. READING THE PLOTS 87
9. 3. ICP 88
9. 3. 1. PARAMETER TESTING 88
9. 3. 2. ERROR ACCUMULATION 92
9. 4. GICP 94
9. 4. 1. PARAMETER TESTING 95
9. 5. MODEL RESULTS 102
9. 5. 1. DEOS 102
9. 5. 2. ENVISAT 103
9. 5. 3. TERRASAR 104
9. 5. 4. ASTEROID 105
9. 6. ORBIT RECONSTRUCTION 105
9. 7. DATA PROCESSING AND TRANSFER 107
9. 7. 1. DATA GENERATION 107
9. 7. 2. DOWNLINK RATE 107

10 CONCLUSION 109

11 RECOMMENDATIONS 111

12 BIBLIOGRAPHY 115

A. APPENDIX A: TRANSFORMATION MATRICES 119

B. APPENDIX B: SVD 121

C. APPENDIX C: DELAUNAY TRIANGULATION 123

Page intentionally left blank.

ABSTRACT

With a growing threat from space debris, it has become more important for space agencies to have means of reducing the probability of it escalating. For this purpose, servicer satellites need to be built. These satellites need to be able to interfere with “dead” space junk and deorbit defunct satellites. In order to be able to navigate around “dead” debris, it is necessary for the servicer satellite to be able to determine its position around the debris and the relative motion between the debris and the servicer.

This thesis describes an algorithm that determines the trajectory of the servicer satellite based on a 3D reconstruction of a model of the target. It can take 3D point clouds from a sensor and transform these point clouds into a 3D model that can be used by the servicer to navigate and determine its position around the target object. The approach uses a simulation of a LIDAR instrument to obtain the data which is used as input for the algorithm to reconstruct the path of the servicer satellite.

The resulting algorithm can perform a 3D reconstruction of a target object in real-time if appropriate parameters are being used and it can also determine the position of the servicer around this model. Moreover, it can also reconstruct the trajectory for the observer around its target.

Page intentionally left blank.

ACKNOWLEDGEMENTS

First I would like to thank my supervisors Hans Kuiper and Ingo Ahrns who have supported me throughout this thesis and for their technical advice. In the same category, I would like to thank Christoph Haskamp.

I am also thankful for professors EberHard Gill and Ernst Schrama for participating in my graduation committee.

Lastly, I would like to thank my fellow space engineering students, my friends, my family and everybody else who has supported me morally and showed interest in my work for this thesis.

Page intentionally left blank.

NOMENCLATURE

LEO	Low Earth Orbit
GEO	Geostationary Orbit
NASA	National Aeronautics and Space Agency
ENVISAT	ENVironmental Satellite
ESA	European Space Agency
EADS	European Aeronautic Defence and Space Comapny
DLR	Deutsch Zentrum fur Luft- und Raumfahrt
DEOS	Deutsche Orbital Servicing mission
LIDAR	Light Detection and Ranging
UAV	Unmanned Aerial Vehicle
GPS	Global Positioning System
INS	Inertial Navigation System
FOR	Frame Of Reference
ICP	Iterative Closest Point
SVD	Singular Value Decomposition
ASTOS	AeroSpace Trajectory Optimization Softwares
COTS	Commercial off the shelf
GICP	Generalized Iterative Closest Point
CM	Correspondence Matrix
CI	Cloud Index
GT	Ground Truth
SLAM	Simultaneous Localization And Mapping
NA	Not Available

Page intentionally left blank.

1 INTRODUCTION

From the beginning of the human race, humans have been known to want to expand and bring improvement. In the Stone Age new tools have been invented and used for the first time. New hunting and crafting techniques have been invented. Later, in the Bronze Age, more sophisticated tools have been invented to fulfill the humans' need for development and expansion. The industrial age has brought with it many machines and mechanisms which can be used to produce different products at much higher speeds at only a fraction of the initial price. And in modern times, intelligent machines are designed which use reason and logic to make decisions and assist their human creators in extremely diverse tasks. However, through all this evolution, the human society has influenced the environment around it in negative ways. From deforestation to toxic waste, the environment has suffered greatly from the useless materials that are thrown all over. In some cases, such as the Maya, these wastes and influences on the environment have caused an entire civilization to be destroyed. As the Maya were expanding, they cut more and more trees from the forests until there were insufficient trees to suck all the rain water before entering the Mayan cities. This led to flooding of the most important cities and made them uninhabitable which led to the decline of the Mayan empire.

Similarly to the Mayan catastrophe, modern humans face a new threat, one which was unimaginable at the time of the Maya. In recent history, humans have reached space. And as it has happened many times in history, humans have started to leave waste in space. And because there are millions of objects orbiting the Earth, some of the objects collide with each other forming even more, smaller objects. One famous case was in 2009 when two satellites crashed and pulverizing each other. The satellites are an Iridium 33 and a Kosmos-2251 which crashed over Siberia with a relative speed of 42,120 km/h (≈11.5 km/s) (Marks, 2009) This crash has created hundreds of thousands of new particles which can in turn crash into other satellites and cause a chain reaction. This phenomenon is called the Kessler syndrome and it is widely feared that if humans do not do anything to prevent more collisions, there will be a time in the near future when there will be so many objects in space that it will be impossible to leave the planet surface.

The Kessler syndrome was first proposed by NASA scientist Donald Kessler in 1978. He proposed a scenario in which, as stated above, the density of objects in LEO is so high that a chain reaction of collisions could take place, each collision generating more space debris which increases the likelihood of further collisions. This will create a "cage" of debris around Earth and make it impossible to for humans to use space or several decades. (Kessler & Cour-Palais, 1978)

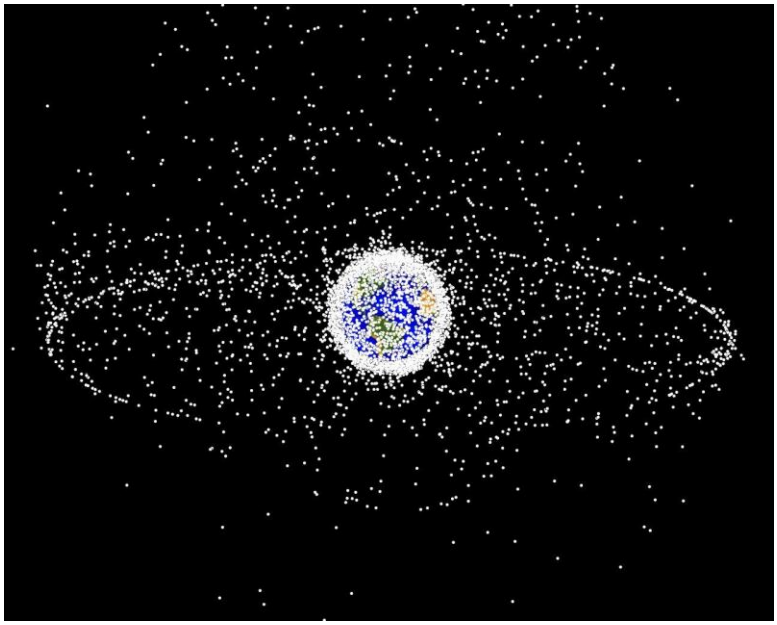


Figure 1.1 A picture showing the distribution of space debris around Earth. There are two main debris areas, one being the GEO ring of debris and the LEO cloud of debris. (CNRS, 2013)

Figure 1.1 shows the debris could around Earth looks like now. It is important to note the high density of objects orbiting very close to the earth. They make the planet look as if it has a fuzzy white coat in the image. These objects orbit in the LEO where most human satellites also exist and therefore are a threat to many of the ongoing missions.

It is greatly feared that the now defunct ENVISAT could be the trigger which would make the Kessler syndrome irreversible. ENVISAT is a large telecommunications satellite which was launched by ESA in 2002. (ESA, 2014) It weighs approximately 8200kg and orbits at a height of 750 km above the Earth's surface. (Astronautica, 2014)

In order to cope with the threat of the Kessler syndrome, many space agencies and companies around the world have begun developing means to deorbit defunct satellites in a safe manner. One of these companies is Airbus Defence and Space formerly known as EADS Astrium.

2 MISSION BACKGROUND

This chapter describes the mission at hand. It describes the reason for this master thesis and how it will be used in the scientific community. It describes the scientific usefulness of the research paper and the background work that has been done prior to the beginning of the thesis.

2. 1. HISTORY OF SPACE DEBRIS

Space Debris is a collection of no longer functioning objects which are in orbit around Earth. The debris includes everything starting from satellites to parts of satellites which are still in space. Even the rockets which carry satellites and are still in space are considered space debris. Since humans have been launching objects into space for more than 50 years, the amount of debris has grown greatly over the years.

Debris can be as small as flecks of paint or tiny bits of metal which come off spacecraft. These hardly weigh anything and are usually not a threat. Other objects are larger than 1 cm and the most common source of such objects is the explosion of larger objects in space. These larger objects are often spent rocket upper stages which still contain highly pressurized fuel on board. As the tanks can no longer hold the pressure after some time, the rocket explodes creating thousands of tiny pieces of debris.

The artificial objects in orbit around Earth are categorized into five main groups.

- Operative satellites: Satellites which are still in operation and which are orbiting Earth. Although they still function, they are still classified as space debris. This is because they are a threat to other satellites as the orbits intersect.
- Non-operative satellites: Non operational satellites are satellites which do not function any more but that remain in orbit around Earth. Some of these satellites are in high LEO, GEO or other orbits which causes them to stay in space for a very long period of time. For satellites at an altitude of 700km, the deorbit time can be as high as a century.
- Upper rocket stages: Used upper rocket stages are the top parts of rockets which brought satellites to their corresponding orbits. These objects are stranded in space as they are usually not programmed to return to Earth to be disposed of in the atmosphere. They are therefore also possible targets for debris.
- Mission related debris: Mission related debris is any kind of debris which is left over by a mission. Examples could be a lost glove or hammer that the astronauts are using while in space repairing a satellite.
- Fragments: are the rest of the objects that float in space. These are leftovers of explosions or crashes which took place in the past and have not entered the atmosphere. This category of space debris is the most numerous of them all, having thousands of particles which orbit the Earth.

Space debris is altitude dependant. The fragments of an explosion or collision tend to stay around the same altitude at which they were produced. This is of course unless the orbit is very low which would cause them to reenter the Earth’s atmosphere and burn up. Moreover, it is very clear that the orbits of the satellites which have exploded have a significantly larger amount of space debris. This can be seen in diagrams which show the number of debris pieces with respect to altitude.

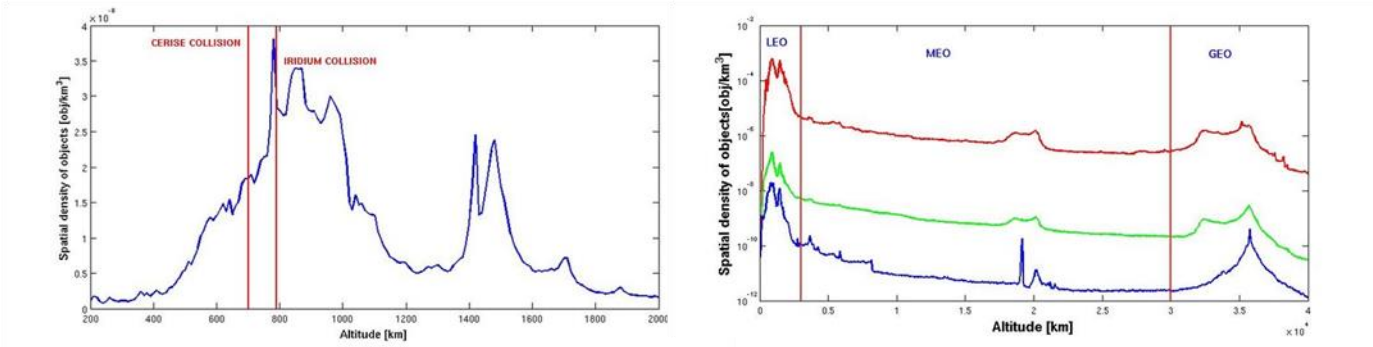


Figure 2.1 Two graphs showing the density of space debris (left) and the density of space debris taking into account the size of the debris (right) where: objects with diameter larger than 1 mm (red line), 1 cm (green line) and 10 cm (blue line). It is easy to see that the orbits where explosions occurred have a significantly larger amount of debris. (Rossi, 2011)

2. 2. THE EADS APPROACH

In 2013, the DLR has announced a project which aims to prepare a new technology mission for the maintenance and disposal of satellites. The contract for this mission has been chosen to be EADS Astrium due to its expertise in the domain.

It has been stated that the project (DEOS, Figure 2.2) will for the first time demonstrate technologies for the control and disposal of defunct satellites and rocket stages in addition to maintenance tasks such as refueling. DEOS consists of two satellites known as a "servicer" and a "client" where the servicer is satellite that carries out the necessary work and the client is the satellite which requires maintenance or disposal. These two satellites will be launched together to a height of 550 km above the surface of the Earth to complete the mission.

The difference of this mission from any other rendezvous missions is that the servicer needs to perform a large number of individual tasks which before were performed by both satellites. The servicer has to approach the client without a tracking signal or any help from the client for example. It also needs to remain at a distance of around one meter from the client for a prolonged period of time while it adjusts its position to avoid collision and keep up with the client.

Using sensors and an arm, the servicer will have to be able to grip the target satellite to stop it from moving with respect to the servicer. For this to be done, an accurate geometry of the client satellite should be obtained so that the best gripping position can be computed. The robotic arm of the servicer will have a total of seven degrees of freedom.

Because of its novelty, the DEOS project is mainly reliant on technologies that have never been tested in space operations before. In the first phase of the project prototype technologies will be developed. This will facilitate the realization of the project at a later time when it will be studied more in-depth.

Two locations of Astrium will be working together for the realization of this project: the team from Germany, Friedrischaffen and the team from Germany, Bremen. Aiding them are a group of subcontractors which are hired by EADS Astrium to complete specific and specialized tasks which they are best at.

DEOS is carried out mainly by DLR with funding from the German Federal Ministry of Economics and Technology. This is because sustainable space operations is an objective for the current German government. Furthermore, this topic of satellite disposal is a topic of great interest at the National Conference on Space Robotics which was held in Berlin in the year 2012. (Airbus-Defence & Space, 2012)

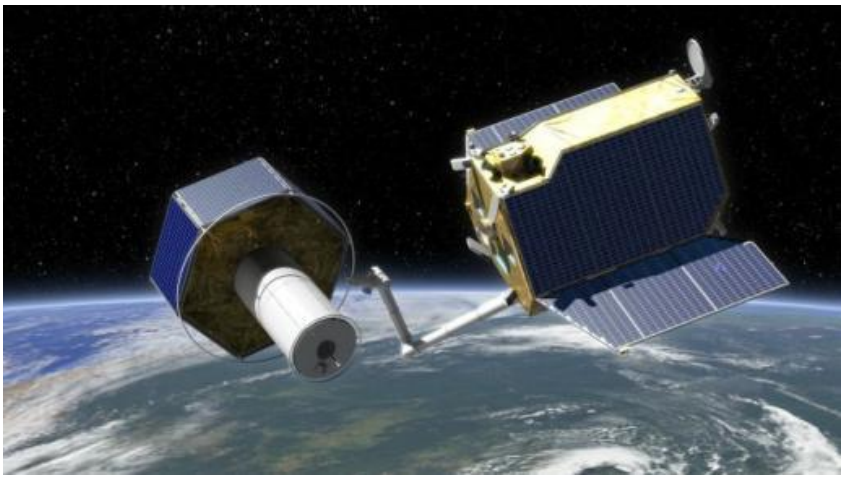


Figure 2.2 This image shows a concept of project DEOS with the servicer on the right and the client being attended on the left side of the image. (Airbus-Defence & Space, 2012)

2. 3. VISION SYSTEM

In order for the servicer to be able to grip the client, it will first need to be able to determine the location on the client where the arm can be attached. For this purpose, the servicer will need a means to "look" at the client and to determine its exact geometry. This is a crucial step as it will determine how the two satellites interact with each other.

Space has the advantage that it is completely transparent as there are virtually no particles to obscure vision. This means that given a sensor, it can detect objects as far away as a million kilometers. (Laser light is not dissipating through space and can be used to measure any distance) However, given the great distance, the resolution of the target decreases greatly which could render the data useless for any processing.

One disadvantage in space is that it is very dark. If the satellites are in eclipse there will be no light to see the target. This means that the servicer must be able to provide its own light for detecting the client even in complete darkness.

Scanning of the target satellite will continue until a full model can be reconstructed inside the memory of the servicer. This is why the servicer needs to chase the client for an extended period of time. During the chase, the scans will be taken and processed by the servicer determining the tumbling rate of the client as well as improving its own orbit. At the end of this phase the rest of the mission can proceed with resupplying the client and later deorbiting it.

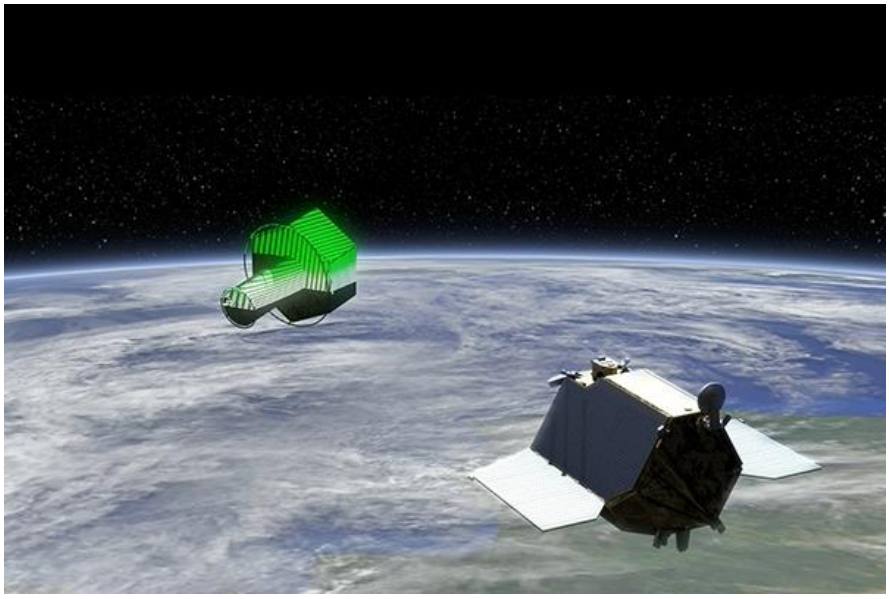
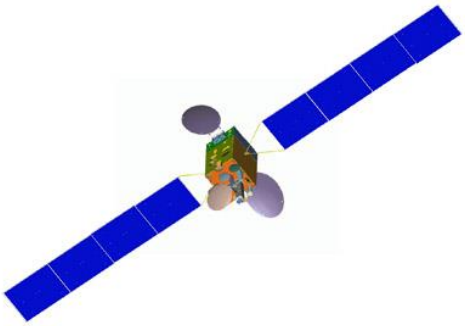


Figure 2.3 Above is an image of the servicer scanning the client. In the image the satellites are both in sunlight, however in reality this happens only half of the time. As the other half the satellites will be in shadow. (DLR, 2012)

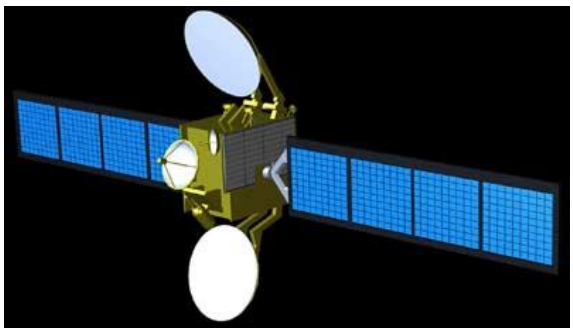
Figure 2.3 shows how the servicer is chasing the client satellite and scanning it at the same time. In the figure a laser is used to get the topography of the client satellite. Lasers are preferable because they require no outside source and because exact ranges can be calculated very precisely.

Although this mission is done with just one target satellite, in reality, the satellites that will be used are real defunct satellites. This means that the exact shape may be unknown and the tumbling rates even less so. For this reason the servicer needs to have a generalized method of geometry determination for any spacecraft it may encounter.

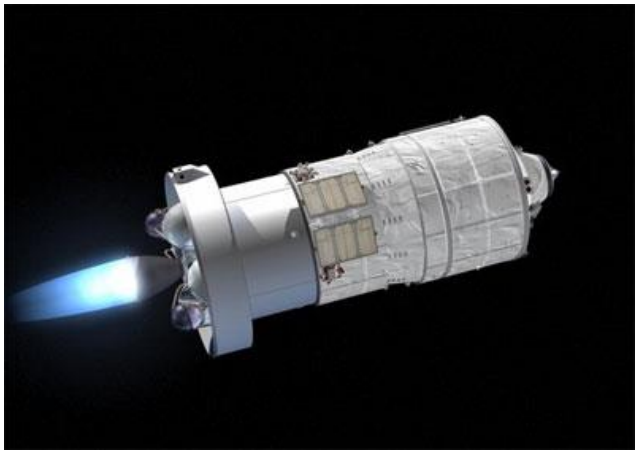
To reconstruct a 3D model of the client, the servicer must have a means of "seeing" in 3D. It needs a camera which can look at a scene and reconstruct it in three dimensions. This topic has already been discussed in the preliminary phase of the project by the project team. In the end the choice has been made to use a LIDAR sensor which can generate a point cloud of an outside scene.



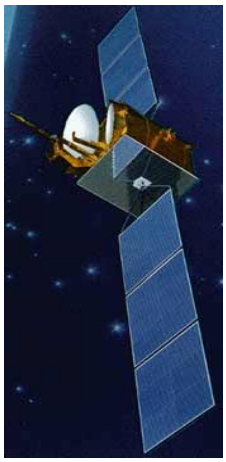
a)Thaicom 3(Astrium, 2004)



b)NigComSat-1(Page, 2010)



c) Ariane V upper stage (CLARK, 2008)



d)DFS Kopernikus 1(Page, 2011)



e)EchoStar-2(Page, 2014)

Figure 2.4 The five images show five different spacecraft which are potential targets for the DEOS project. All the satellites are currently dysfunctional but still in orbit.

2. 4. TRAJECTORY

Deriving the trajectory of the servicer around the target is the main goal of this thesis. It is also mentioned that the trajectory needs to be computed using a 3D reconstruction of the target object. Later chapters will show that the derivation of the trajectory is very simple once the 3D model of the target has been reconstructed; however, reconstructing the 3D model of the target is much more challenging.

Because of this, most of the thesis will be describing the 3D reconstruction part, and the trajectory will only be described at the end.

Page intentionally left blank

3 LIDAR SCANNER

This chapter describes the LIDAR sensor by first giving a general description of the hardware. This is followed by a description of the working principles and the technical specifications of the LIDAR. Finally the data that is obtained from the sensor is described.

3. 1. DESCRIPTION

LIDAR is a sensor which makes use of remote sensing technology that can measure the distance of a point from the sensor in the real world. It does so by illuminating a target object/region with some kind of electromagnetic radiation, and reading the reflected radiation to determine distance to that point. Nowadays, most LIDAR sensors use near infrared, ultraviolet or visible light to determine the range of the target point.(NOAA, 2013a)

The term LIDAR is an acronym of the phrase LIght Detection And Ranging, similar to the word RADAR which comes from RAdio Detection And Ranging.

LIDAR was invented in the 1960s just after the invention of the laser. At first it only allowed the users to determine the distance to a specific point. The technology has greatly evolved in the meantime and now it is possible to use it to generate large point clouds and map entire forests using just one scanner. This large amount of data is then processed to generate the terrain for multiple applications. (CSIRO, 2013)

Figure 3.1 shows a point cloud taken with a LIDAR sensor over a flooded city in the United States of America during hurricane Isaac. These images have been taken using stationary LIDARS. It is important to note that for these scans, the type of LIDAR that is used cannot penetrate water. Some LIDARs can use different wavelengths which penetrate water and allow the user to see the bottom of lakes, rivers, and other water bodies.

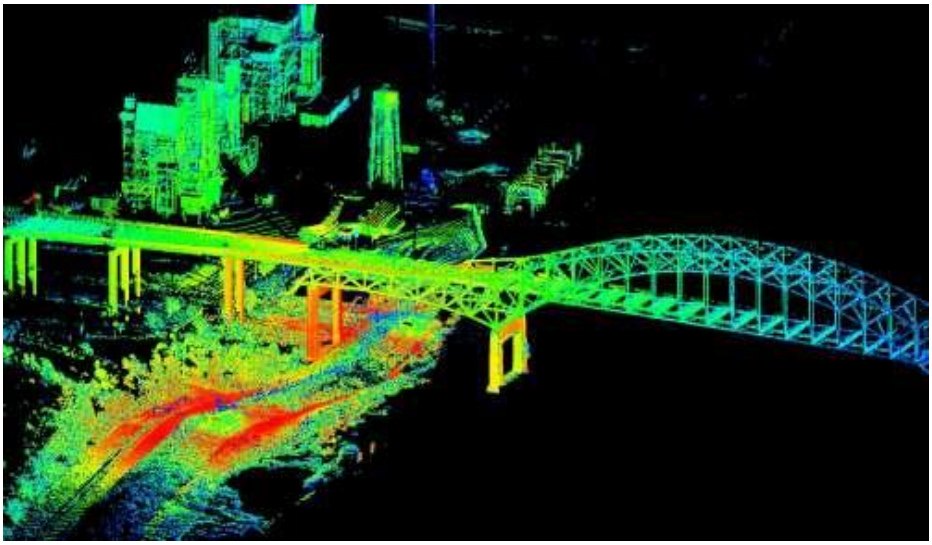


Figure 3.1 A LIDAR scan showing a flood caused by hurricane Isaac in the United States in 2012 The image is a large point cloud.(USGS, 2014)

Currently, LIDAR can be used on multiple materials to without them influencing the quality of the image. Some of the materials which can be detected include non-metallic objects, rocks and glass. Even some of

the more surprising “materials” can be detected, this includes: rain, chemical compounds, aerosols, clouds and even single molecules. (Cracknell & Hayes, 2007)

Because of its ability to detect rain and clouds, LIDAR has been used intensively in the meteorology field where it is mounted on either spacecraft or aircraft to monitor cloud movement and other weather parameters.(USGS, 2013)

Some common applications of LIDAR other than meteorology are listed below:

- Archeology: to preserve ancient artifacts in digital format.
- Geography: To map vast landscapes with accurate heights.
- Seismology: Used to study earthquakes by detecting ground movement, etc.
- Forestry: To map large forests and compute deforestation rates and reforestation rates.
- Remote Sensing: Used to map places which are far away from the sensor, used also in robotics as a remote vision sensor.
- Military: Used on UAVs to map conflict regions.(CSIRO, 2013) (NOAA, 2013b)

3. 2. FUNCTION/WORKING PRINCIPLE

As discussed in the previous section, the LIDAR illuminates targets using different types of radiation and then measures the backscatter to determine the range and position of specific points.

On a functional level, the LIDAR is defined as a combination of three main technologies into a system capable of acquiring data to produce accurate digital evaluation models. The three technologies are:

- GPS: used for determining the position of the LIDAR and the region it maps.
- INS: a system that uses various sensors such as acceleration sensors and gyros to measure rotations and displacements and compute them given an initial position and direction.
- Lasers: which are used to determine the distance to a certain point.

Of these three technologies, the lasers are the most accurate component. The INS and GPS all have errors that are much more considerable. The INS can deviate greatly after a long period of time and constantly needs to be corrected using GPS when possible. GPS depends on space weather and the atmosphere to give reliable readings. (CSIRO, 2013)

Since the speed of light is known, the distance can be computed by measuring the time it takes the light to get back to the LIDAR sensor. The other coordinates of the point are computed by combining the positions and attitudes from the GPS and INS. It is known that the LIDAR can emit at very high frequencies which range from a few points per second to tens of thousands of points per second. This results in large amounts of data being produced in a short amount of time. Assuming that the sensor can emit 10000 points per second, this would result in 600000 points per minute. This is a very large number of points which can be difficult to handle by ordinary computers both in terms of processing power and in terms of memory space.

It is known that some of the LIDAR sensor can be used to obtain multiple points of data from just one beam. This can happen if the beam hits leaves and a portion of the light passes through to hit the ground. The sensor then records multiple points, some from the interactions with leaves and one from the ground. This technique is used to measure the height of trees in forestry applications.

A diagram showing a LIDAR can be seen in Figure 3.2. This diagram shows the working principle of a LIDAR scanner. The laser is usually generated by a crystal which is charged. This light then escapes the crystal and goes through a few mirrors which eventually lead the laser to the "Scanning Mirror" denoted by the number "2" in the figure. This mirror moves in different directions to aim the laser at different regions on the object. The light is emitted on the object from where it is scattered in all directions. The sensor then picks up the signal from the scattered light. This sensor is the "Telescope" which magnifies the incoming signal and makes it easier to read by the rest of the components. Because the target could be very far, it is almost impossible for the instrument to see the target without the aid of a telescope. The signal then travels to the photomultiplier which reads the signal coming from the telescope. The final part (5) reads transforms the signal into an electrical impulse which a computer can understand.

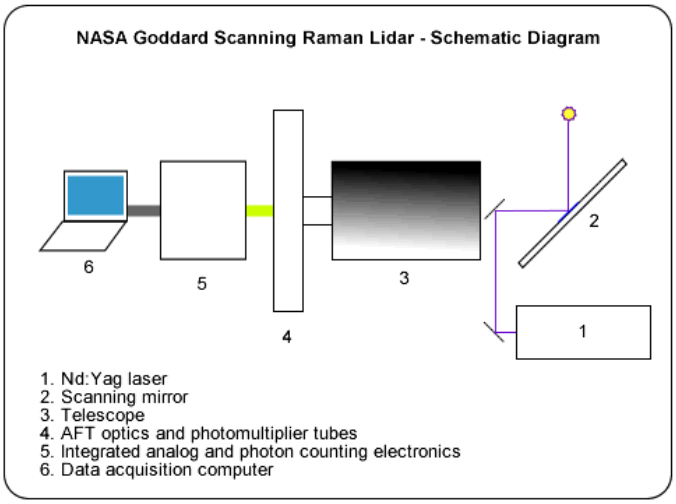


Figure 3.2 A diagram showing different parts of the LDIAR system and the order in which information passes through each of them. (Sox & Wickwar, 2012)

When the information reaches the computer it can be easily deciphered and used for processing and reconstruction of scenes.

Some of the more important components of the LIDAR system are described in more detail below to give a better description of such a system.

- Laser: The laser can range from near- infrared to ultraviolet frequencies. Different frequencies being used for different purposes. The wavelengths used range from 10nm to 1mm depending on the sensor system. One very important aspect of these lasers is eye safety. Eye safety is one of the leading design criteria while building lasers. This is because many lasers can burn the retina in a very short amount of time. The wavelength of the laser is directly responsible for the resolution of the instrument. Usually, wavelengths which are safe for the eyes result in lower resolutions. This also depends on the distance of the target from the LIDAR. If a target is far away, a high powered laser needs to be used, which results in a dangerous laser beam. In turn, to maximize safety, the wavelength is changed so that it does not affect the eyes, however this reduces the accuracy. (Cracknell & Hayes, 2007)
- Telescope: Is an optical instrument which is used to magnify the scene in front of the LIDAR. Assuming that the target is further than 500 m away, and the laser spot is 50mm^2 in surface area, the tiny spot is very hard to detect at such a large distance. In order to be able to detect it, the

telescope is used to zoom into the direction in which the laser is pointed. This allows the device to see the laser light even at a very large distance.

- Photo-detector: There are two different types of photo-detectors, photomultipliers and the solid state photo-detectors. Both of these can affect the quality of the final scan, and each of them is used in certain situations depending on the task of the LIDAR.

Figure 3.3 shows the entire LIDAR system as it scans a field. The figure shows two different LIDAR installations. One being stationary, and the other being airborne. Both images survey the same area however, the processes are quite different.

In the case of the stationary LIDAR, the position of the scanner is constant therefore it is very easy to overlay different scans on top of the other, and the resolutions achieved can be quite high. Also, the time to obtain the scan is indefinite since the stationary scanner can scan for as long as the users want. An airborne scanner by contrast has a much shorter time to scan the environment. This means that the resolution of the scan will be much lower than the stationary LIDAR. However, the airborne LIDAR can cover a much larger area than the stationary sensor, being only limited by the range of the plane and the width of the beam.

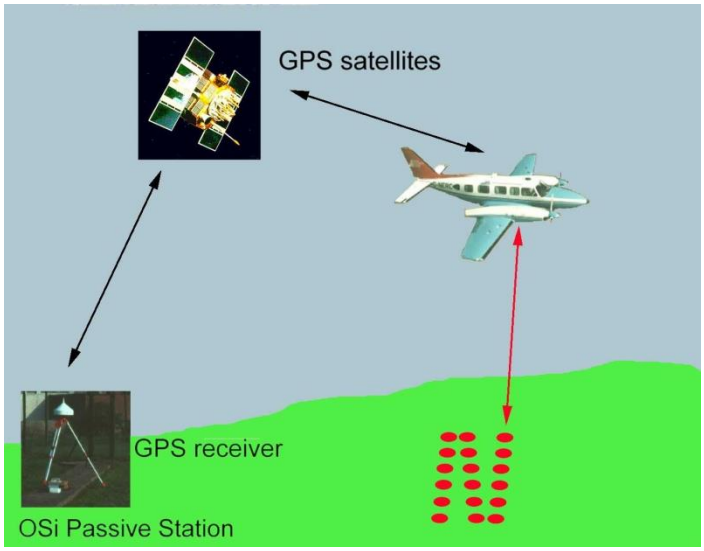


Figure 3.3 Two different LIDAR measuring methods. It is important to note that the GPS is required every time LIDAR is used to map a scenery. (NASA GSFC, 2006)

3. 3. DATA

Different LIDAR devices produce different LIDAR data types. This is mainly because of the different hardware and software that different manufacturers use, but also because of the different applications of LIDAR systems. In most cases, data obtained by LIDAR has four values, three of them giving position coordinates in one of several reference frames, and the last one giving an intensity value. The first three values are self explanatory and can easily be understood, but the fourth value, “intensity” will need further clarification.

The intensity value comes from the amount of energy with which the reflected light hits the sensor. It can also be seen as the amount of light that comes back to the sensor. This value can give properties of the

surface that has been hit. For instance, a low value can indicate a different material/color which absorbs the specific wavelength of the LIDAR.

Unlike the intensity value, the position values are easier to interpret. However, the reference frame which is being used to display the data is very important. For instance, some of the reference frames being used are described below:

- LIDAR FOR: sensors could represent the data in the format of a spherical coordinate system (azimuth, elevation and range). Where azimuth and elevations are the angles of the Scanning Mirror, and the range is the distance to the point.
- Earth FOR: by taking into account the position of the sensor on Earth, it is possible to represent the coordinates of points by using Longitude, Latitude and Altitude. These positions are physical positions on Earth which can be used to generate 3D maps of regions. The geographical position is computed by taking into account a GPS signal which tells the sensor where it is located. This technique can also be used on aircraft and spacecraft. The advantage of using this notation is that the values of the data will always represent a physical position on Earth.
- Cartesian FOR: The points can be converted to a Cartesian space simply by transforming the spherical coordinates obtained above. The Cartesian space can be relative to the LIDAR, or any other point as long as the transformation is known.

The size of a LIDAR file has no theoretical limit. As more scans are being done, more points are obtained and the clouds gain more quality. With this, the file will also take up much more memory. Depending on the applications, different files use different numbers of points. Different industries require different file sizes as they use different kinds of data.

Forestry applications, do not need the terrain to be too detailed, but they do require large areas to be scanned which results in a very large number of points. Architecture applications, by contrast, usually do not require a large area, but they do require an extremely good quality. For the conservation of a statue for example, the surface area is very small, but the amount of detail required is very high if the user wants to be able to see the marks left behind by the sculptor while he was crafting his masterpiece.

The following images show examples of point clouds used in different industries.

Figure 3.4 shows the statue of an angel after it has been scanned using LIDAR. This is an example where LIDAR is being used in archeology to preserve the statue. This method of scanning historic objects requires LIDAR of very high accuracy and very small errors. Otherwise the wrong information would be obtained.

The next image, Figure 3.5 depicts a Chinese Shi which is also preserved in a virtual format after being scanned using LIDAR. The statue has been scanned by the company PropShop for use in 3D applications and conservation.



Figure 3.4 A A splat-rendered view of a high-resolution 3D laser scan of a statue ("Lucy"). From the exact same viewing position as in Figure 3, the model-space scaled splats still fuse to create a solid surface. Because the zoom factor is beyond the level supported by the scan's density, individual splats can be discerned. (Doc-Ok, 2013)



Figure 3.5 A LIDAR scan of an Imperial Guardian Lion from China (Propshop Ltd., 2012)

Figure 3.6 shows a nature scene which was scanned for forestry reasons. The image shows trees as well as the shape of the terrain. Using algorithms, this image can help scientists determine the heights of trees and the number of trees. This allows different scientists to monitor the environment and the effect on forests and ecosystems. Also the scans are sometimes used to generate 3D maps of entire regions, which are used for different purposes by scientists.

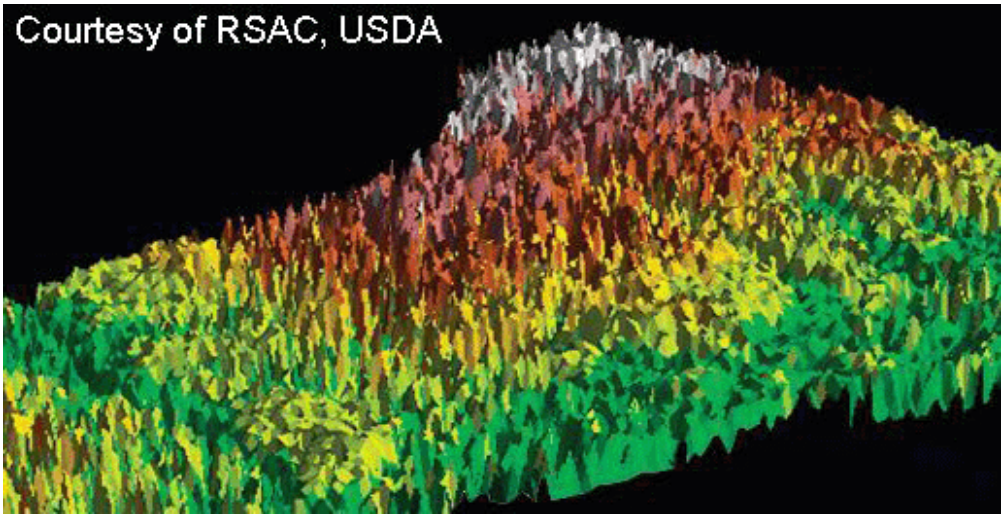


Figure 3.6 Lidar technologies provide information on the comparative height of forest canopy elements. (Jeffrey S. Evans & Hudak, 2012)

Figure 3.7 shows a LIDAR scan of Manhattan in New York. This image shows how LIDAR can be used in urban areas to map cities and show buildings.

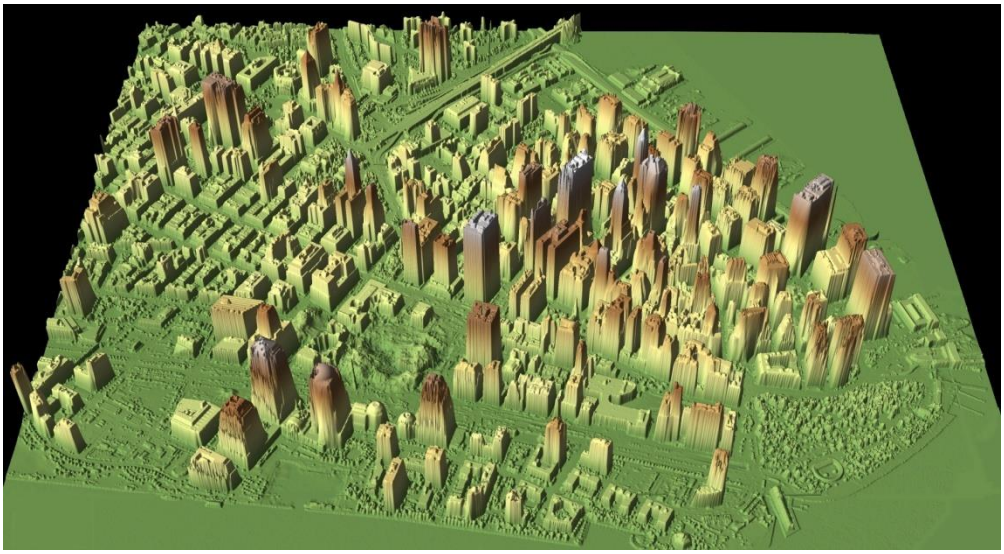


Figure 3.7 A LIDAR image of Manhattan (Dawson & Hu, 2013)

Figure 3.8 shows the rainforest in Honduras. This image, although it shows a forest landscape has been used to find ancient cities which are too hard to find on foot. Once the cities have been found using LIDAR, humans can go there on foot and thus save a very lengthy and expensive search for ancient cities which would normally be done on foot.



Figure 3.8 View of Honduras rainforest. Laser mapping scientists flew over a remote part of the forest and discovered what appear to be ruins. (Diep, 2012)

4 SIMULATOR

As is the case with any expensive project, where new technologies are used, the technologies need to be tested before they are put into practice. This is to ensure that they perform as expected when they need to complete the mission that they were designed for. Since it is much harder to test space equipment by sending them to space, these instruments are usually tested on Earth using improvised scenarios which simulate space and the conditions in space. However, in some cases this is not possible due to the scale of the objects. In this case, simulators are used to computationally recreate events in space.

4. 1. INTRODUCTION

This section will give a brief introduction of the ASTOS simulator and the ASTOS company. It is an important part of this thesis because this simulator has been used to validate the algorithm which has been designed.(Astos, 2014)

4. 1. 1. GENERAL

ASTOS Solutions is a company which deals with the fields of Space, Aeronautics, Automotive, Robotics and Biomedical. It is a software oriented company which offers COTS (Commercial Off The Shelf) software to different clients. The company also offers support for the software which they design. Support of the software is mainly in the field of Orbit Optimization and different simulation services. In some cases it also offers individual solutions for special problems which are not covered by the standard COTS software.

In Airbus, the simulator is currently used for the DEOS mission to simulate various aspects of the mission. (SpaceTech, 2014) One of the tests being the LIDAR sensor which generates 3D point clouds of the target object for the ICP algorithms to be tested. The current simulations use an ENVISAT model which is being scanned by a virtual LIDAR scanner. This is done to collect data for the thesis work.

However, any 3D model can be put into the simulator for scanning. The requirement being that the model needs to be stored in the ".obj" format. In the end, multiple models will be used to validate the algorithm.

The software outputs 3D "frames" which store point clouds that have been scanned. The point clouds are loaded into C++ using algorithms developed at Airbus and other algorithms which have been developed specifically for this thesis.

4. 1. 2. OPTIONS

The advantage of using a simulator to obtain the data is that every parameter of the simulation can be controlled. This includes any kind of error can be simulated with varying degrees of influence to test even the worst case scenarios.

It is also possible to omit errors; this causes the cloud model to be a perfect match to the actual satellite. The flexibility of the simulator is exactly what makes it so much more attractive to use than other means of obtaining the data. It would be possible for instance to have a large facility with an actual LIDAR device and a setup with a decoy satellite which can be observed in different positions. This setup however, will be much more expensive and it would not allow the flexibility which the LIDAR has. (NOAA, 2014) Furthermore, the simulator can be used very quickly to simulate a 3 hour test in a several

minutes depending on the quality required. This makes the simulator much more desirable in such a situation.

This simulator has been used to obtain many types of data with varying degrees of errors. Some data sets have no errors, while others have selective errors, or realistic errors. The models with selective errors are data sets in which errors in specific directions have been computed. This allows the user to mold the data for the required task at hand.

If for example, only a certain aspect of the algorithm requires testing, then it would be much more convenient to simulate a specific error in order to prevent other errors from influencing the result. This can happen in the case of local minima. If the algorithm has a tendency to run into local minima, it is important to generate multiple models with different errors in order to see which errors influence this aspect of the problem and to be able to find a direct solution to this problem.

Other parameters which have been changed are related to the orbit of the satellites and the relative motion between the two objects. The size of the target model has also been modified in some tests in order to test the algorithm against an incomplete model of the target satellite. This is a situation which could happen in real life and is something that should be investigated.

4. 2. SIMULATED DATA

Simulated data is used to test the algorithm, but before more can be said on the testing, it is important to know more about the data format. Knowing the type of data that goes into the algorithm can allow a much better understanding of how it uses the data to obtain results.

The simulator takes as input a model file and parameters for the simulation. It then gives the output depending on the specified requirements.

The simulator produces two types of data which will be used by the algorithm. First, there is the realistically available data which is data that a real LIDAR would give during a mission. This data contains noise and errors which need to be taken into account by the algorithm. The second type of data is data which is only made available because of the simulator. This data represents the exact error magnitudes of the realistic data. It also contains the correct transformations which can be used to obtain the 3D model back of the satellite. If this data is used to reconstruct the model, the error would be exactly zero. However, this data can only be used for validation as it is not available in a real mission.

The following two sections will elaborate on these two data types and explain its format.

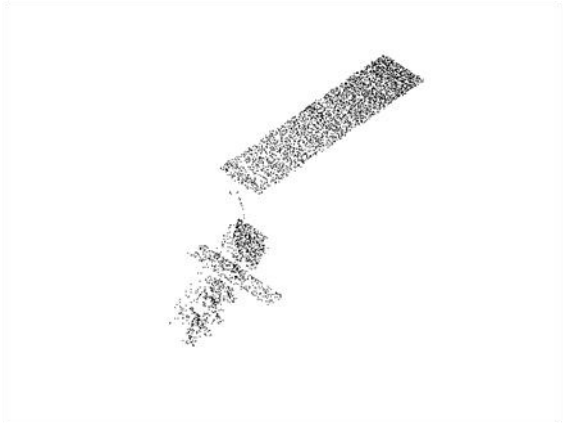
4. 2. 1. REALISTICALLY AVAILABLE DATA

The main goal of the simulator is to provide realistic LIDAR data to the user. This data is the same type of data which would be used in an actual mission where the servicer satellite only has a LIDAR instrument to obtain information about the surrounding environment.

Simulated data is stored in a 3D frame which can be used by the algorithm. A frame represents one set of measurements taken in one instant in time. The frame described above is similar to a frame in a movie except in this case the frame is not a picture, but a 3D point cloud.(PCL, 2014) A complete scan of the entire trajectory includes multiple consecutive 3D frames which show the target satellite from different angles as the two bodies move with respect to each other.

The simulator is set to take frames at a rate of approximately 3 Hz. Although this rate is much higher than what is needed for the ICP algorithms, it offers a large set of images which can be used by the algorithm. The actual algorithm will use only a small fraction of the point clouds. However, the rest of the generated point clouds are still useful as they can be used later to validate the model which is generated by the algorithm.

Figure 4.1 shows several frames which have been obtained from the simulator. Some of the properties are given below. These images should give a first impression of the type of data that is being inserted in the algorithm.



a) Frame 1: Image taken from bottom



b)Frame 15: Image taken from left



c) Frame 30: Image taken from bottom right

Figure 4.1 Three different point clouds. Each image shows a frame taken from a different position around the target spacecraft

The properties of the scans shown in Figure 4.1 can be seen in Table 1. These properties are general parameters of all the scans taken from the model, and of the simulation. They help the reader get a better understanding of the simulation environment.

Table 1 Properties of the figure above

Property	Value
Number of points	2000 – 5000

Size of model	18x10x10 [m]
Errors used in the model:	±5[cm]
Type of orbit of the model:	circular at 700km altitude
Scan frequency:	3[Hz]
Number of images in one full revolution	40
Target	ENVISAT

The format of the output is crucial to understanding the algorithm and how it works. For the purpose of writing this thesis, the point clouds have been saved in text files. Text files have been used because they are easy to read and to check for errors. Each file contains all the points from one point cloud, which means that the number of files is equal to the number of point clouds or frames of the model.

Each line of the file stores information for one 3D point. The information stored represents the X, Y and Z coordinates of the specific point. This results in a large file with three main columns and thousands of lines. For simplicity, there are no headers or any other type of information available in the files. The number of the point cloud is saved in the name of the file.

An example of this can be seen in below in Code 1 where five 3D points are being saved. The values represent coordinates in 3D space for some of the points from the first point cloud of ENVISAT which is also shown above in Figure 4.1 a).

```
-3.1342278532209775e+000 2.4474273525816592e-001 -4.7343018160635788e+000
-2.8157409123990007e+000 4.4854097027906370e-001 -4.8851315999194762e+000
-2.8706315150083341e+000 3.6484201535785132e-001 -4.8917686979504493e+000
-2.7781424941554604e+000 5.6614362742357827e-001 -4.8063681135799170e+000
.
.
.
 1.6962289471954829e+001 2.4769684042292202e+000  1.1473999241991971e+001
```

Code 1 Snippet from the first point cloud taken of the ENVISAT model

It should be noted that the text file format is used for this thesis for clarity, but any other format that stores 3D points can be used with the algorithm.

Another important aspect is that the points from Code 1 are all denoted in Cartesian coordinates. However, a LIDAR does not detect the object in Cartesian coordinates. The LIDAR takes measurements in spherical coordinates. This means that each point's position is given by two angles (azimuth and elevation) and a range measurement.

The two angles are obtained by measuring the angles of the scanning mirror(see Figure 3.2) present in the LIDAR instrument, while the range is obtained by computing the time it takes the laser beam to hit the target and bounce back. However, the algorithm only accepts point clouds in Cartesian coordinates, which means that the points represented in spherical coordinates need to be transformed to Cartesian coordinates. In order to transform spherical coordinates to Cartesian ones, three formulas can be used. These three formulas are given below in Equation (1).

$$\begin{cases} x = r\sin\theta\cos\varphi \\ y = r\sin\theta\sin\varphi \\ z = r\cos\theta \end{cases} \quad (1)$$

Where:

- x, y and z are the Cartesian coordinates of the point
- r is the measured range to the point in spherical coordinates
- θ and φ are the elevation and azimuth respectively measured in spherical coordinates

If the range is equal to 0, it can be seen that all the values for x, y and z are also zero. The range is only zero when the light beam does not travel any distance from the LIDAR. In other words, the origin of the coordinate system is always the position of the LIDAR and the servicer spacecraft. This information is very important for rebuilding the trajectory of the servicer around the target satellite.

Figure 4.2 shows a part of the raw scans of the ENVISAT satellite. The shape that is visible in the figure appears because of the rotation of the servicer around the target satellite. When placed in the same reference frame, the clouds appear to be rotating around a central point.

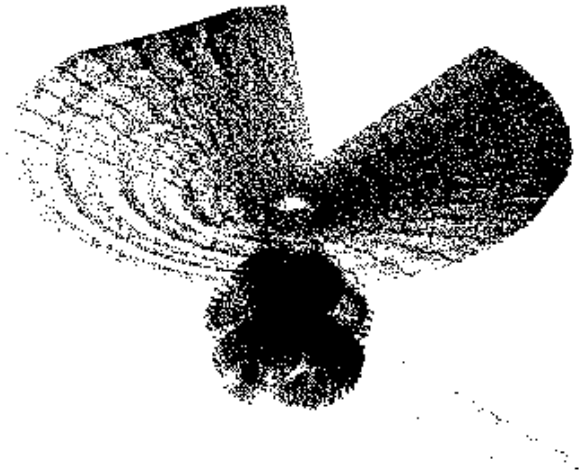


Figure 4.2 Raw scans 1 to 30 of the ENVISAT

The clouds in Figure 4.2 represent the ENVISAT while the servicer is rotating around it. Only a part of the clouds are being shown for a clearer impression of what the input data of the algorithm looks like. Note the large errors present in the bottom right corner of the image.

4. 2. 2. SIMULATION DATA

In addition to point clouds, the simulator also outputs other values which a real LIDAR sensor would not be able to give. The most important output which can only be obtained from the simulator is the Ground Truth (GT). The Ground Truth represents a series of rotations of the servicer around the target. These rotations capture the complete relative motion of the two satellites.

If these transformations are used to transform the point clouds shown above in Figure 4.2, a perfect reconstruction of the target satellite is obtained. The perfect reconstruction after applying the Ground Truth transformations can be seen in Figure 4.3.

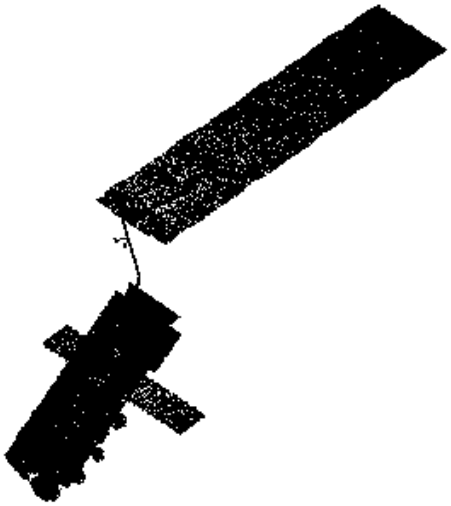


Figure 4.3 Full reconstruction of ENVISAT using the Ground Truth

These rotations are what the algorithm needs to compute. They are the rotations which, if applied to the raw scans, will result in a perfectly reconstructed model of the target satellite as mentioned earlier. In the end, the rotations computed by the algorithm are compared to the rotations of the Ground Truth in order to determine the errors of the algorithm. In the ideal case, there is no difference between the transformations computed by the algorithm and the ones taken from the simulation directly.

Moreover, because this information cannot be obtained in a real life scenario, the information is only used in this phase to verify the errors of the model. This is another advantage of using a simulator over obtaining the data experimentally.

Similarly to the point clouds, the Ground Truth is also saved in a text file. However, the format of the Ground Truth is different from that of the point clouds. This file stores one transformation for each point cloud stored on different lines in the file. This means that each line corresponds to one transformation. The transformation has a rotation and a translation component. Each of these components has three parameters.

For the translation, the parameters represent distances in the x, y and z directions, while for the rotations, the parameters represent values for roll, pitch and yaw.

This means that the file that stores the Ground Truth contains six columns where the first three contain the values for translation and the last three columns contain the values for rotation.

4. 3. CORRELATIONS BETWEEN THE SIMULATOR AND REALITY

A simulator is used to obtain data for the algorithm because setting up a full scale test is much too expensive and hard to accomplish. Simulators offer a fast and easy way to obtain data. They also offer many options which the user can select for different scenarios and cases.

If a full scale test were to be attempted to obtain the data, it would mean that the orbiter would need to be 50m from the target satellite and that it should rotate around the target. This would also involve having a full scale model of the target object. This would be very expensive to obtain and place considering the

size of the ENVISAT. Another problem with a full scale test would be the atmosphere. In space there are almost no atoms to interact with the LIDAR instrument. However, if the test were to be attempted on the ground, the atmosphere would be a factor that could induce errors which would otherwise not have appeared in space.

For all these reasons, the research team has decided to use a simulator to obtain the data for the algorithm. As mentioned earlier, a simulator has the added benefit of allowing the user to configure many different parameters such as accuracy, error, distances, rotations, point densities, etc.

Using a scale model and a LIDAR sensor to generate the data may have been discussed in an earlier stage of this project, but the decision to use a simulator has already been made before the start of this thesis and this cannot be changed.

The next two sections will discuss the setup of the actual and the virtual experiments and the error budget and how this translates from the algorithm to reality.

4. 3. 1. SETUP

As mentioned above, the simulator can be used to generate data under many different circumstances. The simulator simulates a space environment with a target and a servicer which interact with each other.

However, as with any kind of simulator, there will always be differences between the simulated data and real life data. Fortunately, for this thesis, the data generated is not very complicated and the differences between the simulated data and actual data can be discussed and examined to see just how influential this aspect is.

For this thesis the relevant data consists of points in 3D space, and are collectively called a point cloud. A point cloud is a collection of points each of which has three values, one in each dimension. If there are differences between the simulated point cloud and the actual point cloud, they would occur either in the position of points or in the number of points.

For example, for the first kind of error, the simulator could be giving resulting positions of points which are different from the positions that would result from a real case. The errors could have higher standard deviations or means that they normally would in reality. While for the second kind of error, a given configuration in the simulator would give a different number of points than the same “configuration” in reality. Given these two types of possible errors, the algorithm needs to be flexible enough to account for differences in data between real and simulated sources.

Because of the simplicity of the data it is very easy to distinguish correct data from wrong data even by simple visual inspection. This is because in reality, the resulting point cloud would resemble the target object very closely. Similarly, if the simulated point cloud resembles the target object it also means that the simulator is successful. Because of the nature of the algorithm, the only requirement is that the resulting point clouds appears visually similar to the object and does not require a precise reconstruction of the target. This can be seen in Chapter 9. 4. 1 where errors (noise) five times larger than expected are tested and the algorithm still runs correctly with minimal increases in the total error.

This means that because of the nature of the data, it the errors which are caused by the fact that the data is generated artificially and not realistically can be neglected and are smaller than the errors which have been tested. Therefore, the fact that data is only simulated will not negatively affect the conclusions and realistic data would only increase the complexity and cost of the project while not offering any benefit.

4. 3. 2. ERROR BUDGET

Before doing any serious calculations, it is good practice to examine the error budget of the setup. This will provide a better understanding of the errors which can appear and where they come from. Having a better understanding of the errors and their origins is essential for writing an application which needs to take these into account.

The 3D coordinates of each point are computed based on three variables. All the measurement errors emerge from inaccurate readings of these values. The variables are:

- Laser Range sensor
- Azimuth angle of mirror
- Elevation angle of mirror

It is important to note that the LIDAR not natively compute positions in the Cartesian space. Instead, when measurements are taken, they are in spherical coordinates with the mirror located at the center of the coordinate system.

The first kind of error that can appear is an error in the distance of the point to the origin. Of the three variables described above, this variable will give the lowest error. This is because measuring ranges with laser light is a very accurate process. The larger errors are caused by the aiming of the mirror as can be seen below.

Both azimuth and elevation angles are used to aim the mirror in the direction of the target. However, each of the angles corresponds to a different axis. Although the exact specifications of the LIDAR have not been provided due to sensitivity of the information, it is known that for a distance of 50 m, the errors are approximately 4 cm. This translates to an angle measurement accuracy of approximately 0.05^{o}, which means that the angle of the mirror can be measured to an accuracy of 0.05 deg. The angular error is expected to grow linearly with distance.

In order to ensure that the measurement errors which result from the LDIAR do not affect the final trajectory reconstruction, tests will be made with errors which are much larger than the realistic errors. If the algorithm will be able to deal with errors several times larger than the realistic errors, then it is sure to work even for realistic (smaller) errors.

5 ITERATIVE CLOSESR POINT

This chapter describes the Iterative Closest Point (ICP) algorithm. First it describes what an ICP algorithm is, and why it is so important for this project. Second it describes the working principle of the ICP and also gives formulas which the reader can use to better understand the logic behind the algorithm. Lastly, the different variations of ICP are explained and the advantages and disadvantages of each of them are described.

5. 1. DESCRIPTION

The ICP algorithm is currently being widely used in many applications around the world. The ICP is employed to minimize the distance between two point clouds which have overlapping regions. It does this by finding the parts which overlap and joins them together with a minimum error to construct one continuous point cloud from the two initial point clouds. Some of the uses are listed below:

- 3D Reconstruction: one of the most widely used task for ICP is to reconstruct 3D surfaces based on multiple 3D scans. It takes two scans at a time and combines them with minimum errors to obtain a single continuous scan.
- Robotics: because more and more robots use LIDAR sensors for vision, the information gathered from these sensors can be used to determine the position of the robot and to compute the optimal path to a certain point. Usually robots compute their position based on internal sensors which tell them how far they traveled. However, since this is not always possible, the ICP method can be used on LIDAR scans to determine the distance traveled by the robot.

The algorithm works by aligning two properties of the clouds. First it aligns the rotation to ensure that the objects have the same orientation. Afterwards it aligns the translation to ensure that the clouds overlap well with each other. There are some cases in which the algorithm will be unable to align the clouds properly and this will be explained below. (Young-Hoo Kwon, 1998)

For a few years, the Iterative Closest Point algorithm has been the primary method for aligning 3D models based purely on the geometry and in some cases, color of the point clouds. The ICP starts off with two meshes, which can be seen as point clouds generated by two different scans of one object. (Pulli, 1998) The algorithm also assumes an initial transformation between the two bodies. As it runs, the Iterative Closest Point algorithm tries to refine the initial guess of the relative positions of the point clouds. This is done by minimizing an error function which is defined by the user.

The algorithm works best when there are many features on the object which it can use for alignment. If for example there are two flat plains, the algorithm will perform quite badly because it will be unable to align the translation. This will result in clouds which have the correct orientation but which ate slightly shifted with respect with each other. In order to avoid this, the clouds need to have features which are structures on the plane. These “structures” will help the algorithm align the model correctly. The more unique features there are, the better it will work. A point cloud with good features is a point cloud which looks uneven and has no symmetry. This makes it possible to have only one good alignment and the algorithm will have one optimal solution. It reduces number the local minima of the optimization function and makes the global minimum stronger and more likely.

The initial alignment guess can be done in a variety of ways such as taking the positions of the scanner for both scans, by computing the principal axes of the scans or even randomly. The larger the initial error however, the longer it will take for the algorithm to converge. (Young-Hoo Kwon, 1998)

Figure 5.1 shows two point clouds which are about to be aligned using ICP. It is very clear that the two clouds have a similar shape and therefore have similar features which can be aligned by the algorithm. In this case, the face has parts of both sides of the face in its point cloud. If the two models had completely different sides, then this alignment would have been impossible. However, since the nose, lips, chin and forehead have so many parts which exist in both point clouds, the alignment has been done with a good convergence. (Rusinkiewicz & Levoy, 2001)

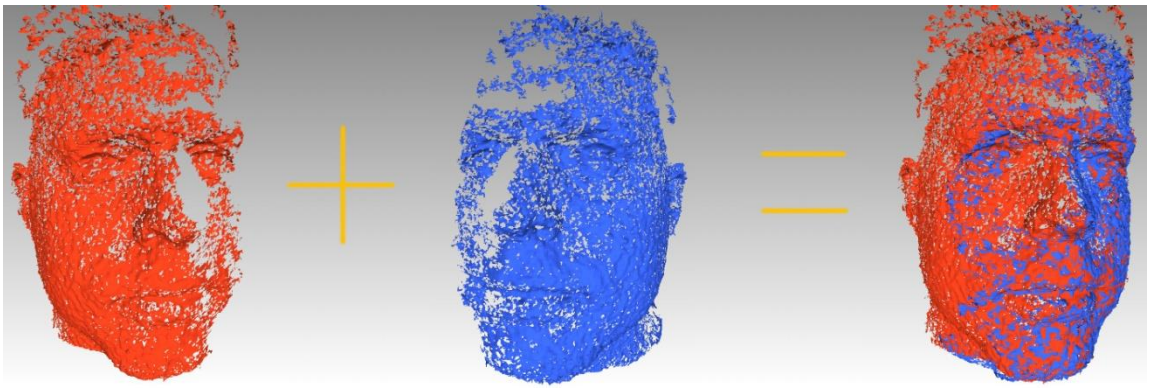


Figure 5.1 Two sides of a face being brought together using an ICP algorithm (ISR., 2012)

Currently, the ICP algorithms can only take two clouds because of the exponential complexity of using more than two clouds. From these two clouds, one of them is taken as stationary (the reference point cloud), and the second one is movable with respect to the reference. The algorithm then starts to iteratively move the second point cloud on top of the reference cloud. Usually the algorithm converges within 15 steps as it is quire efficient at managing the data.

Figure 5.2 shows two point clouds which need to be aligned using ICP. The point clouds depict the same wall of a room. If the image were to be correct, the two walls would be overlapping. However, it can be easily seen that this is not the case. Not only do the walls have a translation of 2 m between them, but they are also not oriented in the correct direction. (Amberg & Vetter, 2003)

What the Iterative Closest Point algorithm is expected to do is to align the clouds perfectly one on top of the other. Depending on which cloud is chosen to be the reference, the other will be the one that moves.

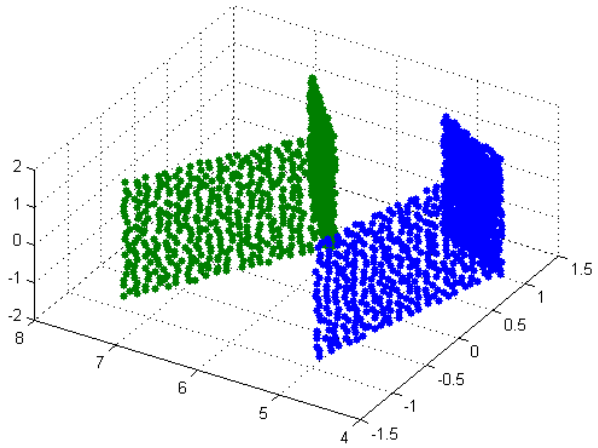


Figure 5.2 Two point clouds of similar shapes are shown in different colors

The way in which the ICP functions is to first find the closes point to each point in the reference cloud. This is best seen in Figure 5.3 where there are two point clouds being shown. The red point cloud is looking for correspondences in the blue point cloud. The lines connecting the two point clouds represent connection lines between each point. It is clearly seen that some of the points in the blue point cloud have no correspondence in the red point cloud. This is because they are too far away from any other point in the red cloud.

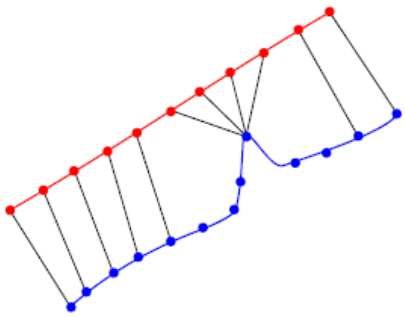


Figure 5.3 Two point clouds with the corresponding points from the red point cloud to the points in the blue point cloud. (Kjer & Wilm, 2010)

After finding the closest points, it ignores points which have no connection to other points. This ensures that the two clouds are only made up of points which have a connection to the other cloud. (Brown, 2004) After the extra points are deleted, the remaining points are put into a matrix. Each cloud is used to generate one matrix which has the column number equal to the number of dimensions which in this case is three, and the number of lines equal to the number of points in the matrix. After the two matrices are generated, one for each point cloud, they are multiplied to obtain a 3x3 matrix. The resulting matrix is used to obtain the rotation and translation of one point cloud to the second. (Kjer & Wilm, 2010)

To obtain the rotation a (Singular Value Decomposition) SVD function is used. The results of the SVD are three matrices which used to obtain the rotation and translation of one of the point clouds with respect to the second one. After obtaining the transformation matrix, the wall which needs to be moved is multiplied by the matrix in order to update its position and then the whole process is iterated.

Figure 5.4 shows the connections between some of the points from the two point clouds in Figure 5.2. The reason for not showing all connections is because there would be too many lines in the image and it will become impossible to understand. (Agarwal & Arge, 2006)

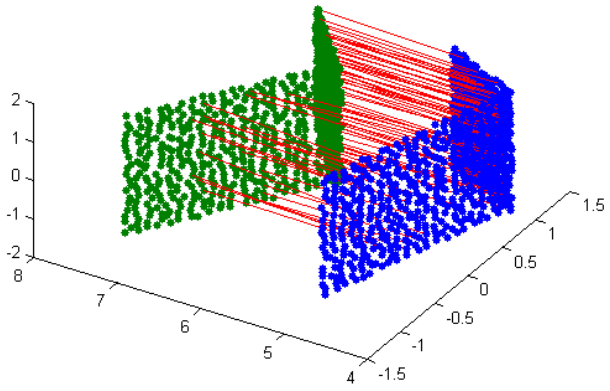


Figure 5.4 A small part of the connections between the two point clouds is shown

On the second iteration, all the points of the clouds are used again. The connections are reformed in each step of the algorithm. This is because as the objects move relative to each other points in the reference point cloud will have different closest neighbors in the second cloud. (Sorkine, 2009)

In the end, the two point clouds will converge and the image will look as it does in Figure 5.5.

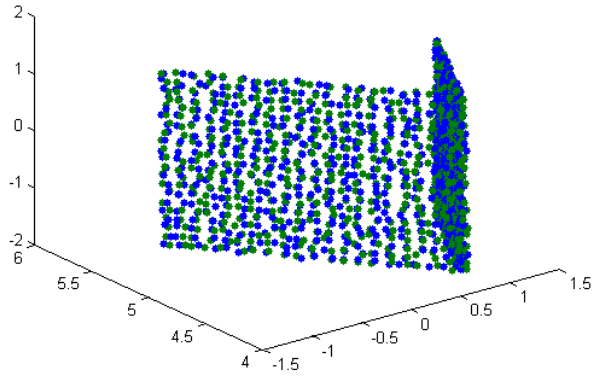


Figure 5.5 The two point clouds are shown overlapping, after alignment

5. 2. POINT CLOUD

A point cloud is a collection of points in a coordinate system. The set of points can be obtained through any means either generated artificially or measured by an instrument. Such data structures are used in this algorithm to generate a reconstructed model and trajectory of space debris. (Sorkine, 2009)

A point cloud is defined in mathematical notation in Equation (2):

$$S = \{p_i \in \mathbb{R}^3 | i = 1..n\} \quad (2)$$

Where:

- p is a point in a 3-dimensional Cartesian coordinate system and the three dimensions are all real numbers.
- n is an integer which represents the index of the point
- S is a set of points p

Depending on the task at hand, it may be more convenient to represent point clouds as matrices to simplify multiplications and other operations. (Sapiro, Hall, E, & Mémoli, 2004)

5. 3. FINDING CORRESPONDENCES

The first step in the ICP algorithm is to find correspondences between the two point clouds. This can be done using a wide variety of methods. The most common means to find correspondences is by using a kd tree to generate a 3D search space which can then be used to quickly find the closest points to any given point in the second point cloud. (Princeton University, 2005)

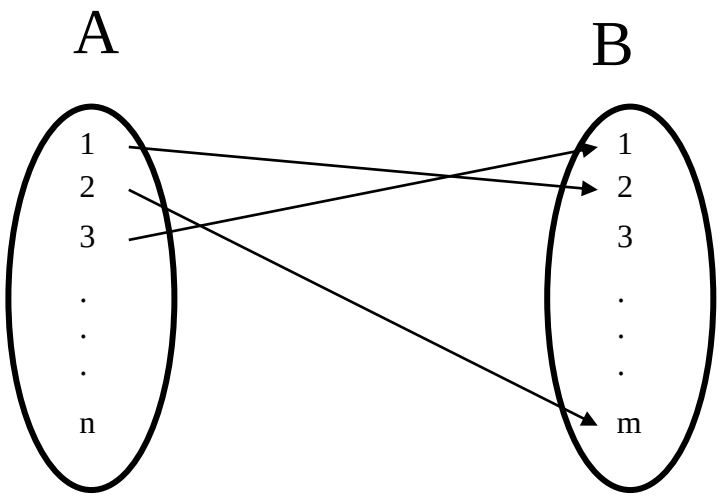


Figure 5.6 Two sets of points that represent point clouds A and B, only the indices of the points are represented in the image.

Figure 5.6 shows two point clouds which need to be aligned. The point cloud on the left is called point cloud A and it contains n points. The point cloud on the right is point cloud B and it contains m points. The two point clouds are used in the following example for the algorithm to find correspondences.

This part functions by finding the closest points from the second point cloud to every other point in the first point cloud. In Figure 5.6 the correspondence of point point 1 from point cloud A is point 2 from point cloud B. (Kjer & Wilm, 2010)

In terms of a mathematical notation, point cloud A can be seen as the domain, while point cloud B can be seen as the codomain. Using this notation, there exists a function $f: A \longmapsto B$ for which every element in A has one connection to another element in B. For a clearer understanding, A and B are described below in equations (3) and (4). Function f (Equation (5))thus finds the element in B which is closest to the element in A. (Princeton University, 2005)

Given:

$$A = \{a_1 \quad a_2 \quad a_3 \quad \dots \quad a_n\} \tag{3}$$
$$B = \{b_1 \quad b_2 \quad b_3 \quad \dots \quad b_m\} \tag{4}$$
$$f(a_i) = bj; \left|a_i - b_j\right| = \min_k\{a_i - b_k | k = 1 \dots m\} \tag{5}$$

Where an arbitrary point a_i is defined below in equation (6).

$$a_i = (a_{x_i} \quad a_{y_i} \quad a_{z_i}) \tag{6}$$

The formulas above shows that for every point in the domain, the closest point from the codomain is found by using the minimum function. After computing the correspondences they are used in the next part of code to move the point clouds on top of each other.

5. 4. REGISTRATION

Registration is the part of the algorithm in which the correspondence information is being used to compute the transformations between the point clouds. The Registration code block is composed of several modules each with its own function. The functions of these modules will be discussed below for a clearer idea of how the registration code block functions. (Breitenmoser & Tishchenko, 2010)

The ICP uses several formulas to obtain the results which it gives as output. However, before diving into the formulas, it is important to know how the ICP functions. The segment below is a pseudo-code of the ICP algorithm. This pseudo code will give a brief introduction to the actual coding of an Iterative Closest Point algorithm. (Agarwal & Arge, 2006)

Pseudo Code of ICP:

```
1. Initialize guess transformation
2. Transform cloud 2 by guessed transformation
3. Find correspondences
4. Eliminate points with no correspondences
5. Move point clouds to the origin
6. SVD
7. Update Cloud 2
8. Iterate from step 3
```

Code 5.1 Pseudocode of an ICP algorithm (Nelson, 2011)

To have a better understanding of this code block, a flow chart will be given which shows which modules are involved and the order of these modules.

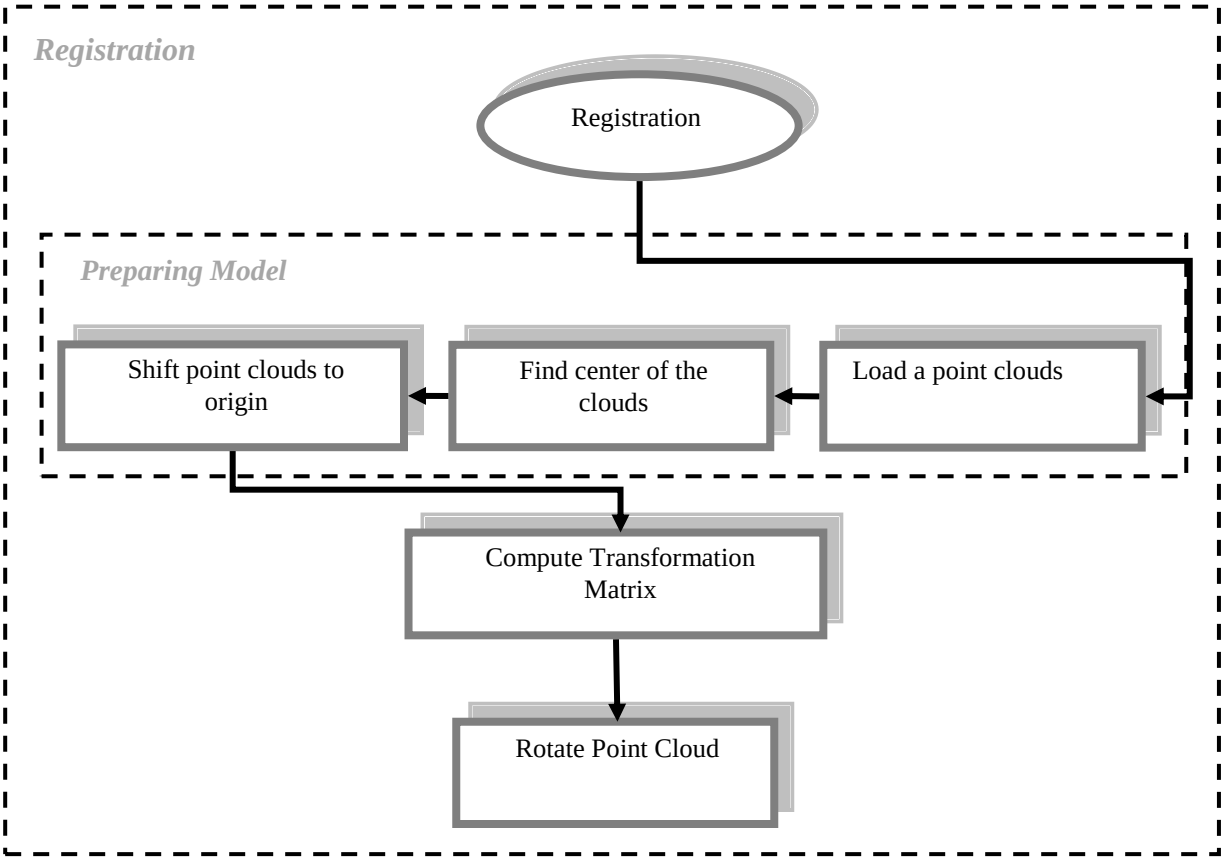


Figure 5.7 Block diagram of the Registration procedure

Figure 5.7 is a flowchart of the entire "Registration" section of the algorithm. This section of code is composed of five different steps. Each of the steps has its own specific task which helps the ICP converge.(Kabsch, 1976)

The way in which this section works is first by loading the two point clouds. Then the center of the point clouds is computed by averaging the dimensions of all the points in the point clouds. And by moving the point clouds to the origin of the coordinate system to allow the algorithm to begin transforming one of the clouds. After obtaining the transformation matrix, the point cloud is transformed by multiplying point cloud 2 by the matrix. The number of times that this iteration is done depends on how long it takes the algorithm to converge, but as mentioned previously, 15 iterations of this process is enough to achieve an almost exact alignment of the point clouds.

Following the steps given in Figure 5.7, each of the steps will be described in more detail.

It should be noted that for making this demonstration simpler to understand, 2D-point clouds will be used. However, in reality the point clouds are three dimensional. This can be easily imagined by replacing all the z-axis values with 0 and only visualizing the x and y axes.

The demonstration starts off by giving an example of input data. For now, the input data consists of two point clouds and the connections between these two point clouds. The connections between the point clouds are obtained using the method described in section 5. 3 Finding Correspondences.

First, the point clouds and their respective correspondences are being loaded. However, due to the fact that not all the points may have connections to other points, not the entire point clouds are taken. Instead, only the relevant points are being used. (Eaton, 1983)

This process ensures that only the points which have valuable information are used for the model reconstruction. Points which have no useful information are hence ignored until they have connections. This can happen in later iterations of the algorithm.

In this example, the red point cloud represents point cloud A and the blue dots represent point cloud B. This means that the blue point cloud needs to be moved over the red point cloud. Figure 5.8 shows these inputs in an arbitrary reference frame. It should be noted that some of the points have no connections as mentioned in the paragraph above.

Note: figures used are not to scale. They are purely used to graphically explain the algorithm.

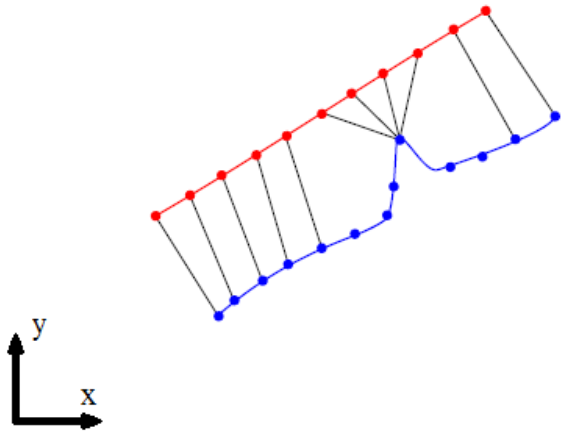


Figure 5.8 Two point clouds with the connections shown

Next, the inputs are sent to the first step which loads the point cloud and the imaginary point cloud. The results of this process are displayed in Figure 5.9. It is immediately visible that the points which have no correspondences are ignored.

In reality it is possible to have an unequal numbers of points in both of the point clouds. However, for the algorithm to be successful, the numbers of points in the two point clouds need to be the same. In order to fix this, the point clouds are “recreated” based on the connections between the point clouds.

Since each connection has two ends, (one in point cloud A and one in point cloud B) it is possible to use the points from the connections to regenerate the point clouds. In the example given above it can be seen that four of the red points connect to the same blue point. This means that the red point cloud theoretically has three extra points compared to the blue point cloud.

To avoid this problem, the blue point which has four connections is “copied” three more times. To visualize this, it can be said that there are in fact four blue points in the exact same position in 3D space.

This representation allows the algorithm to function with point clouds of any size. The step done here is crucial for the rest of the algorithm, as the numbers of points needs to be consistent in order to have a good solution.

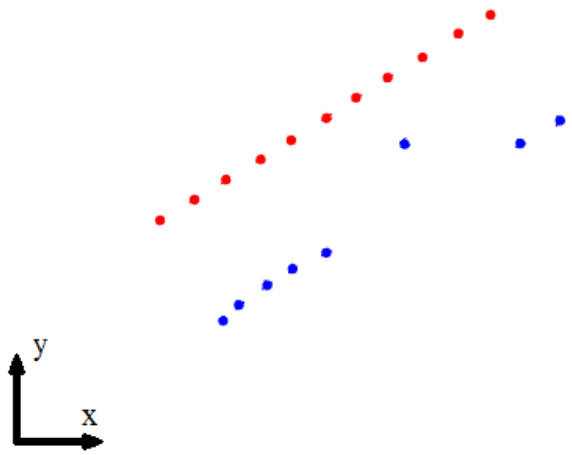


Figure 5.9 points with no connections have been eliminated

It is very important to notice that the point clouds are not in the origin of the coordinate system. This will be fixed in the following steps which have the exact aim of bringing the clouds in the center in order to compute the transformations.

In the next step, the incomplete point clouds selected by the previous step are used to compute the centroids of each of the point clouds. The reason for finding the centroid is so that the points can later be brought to the origin of the coordinate system. (Sapiro et al., 2004)

The position of this point is computed by averaging the positions of all the points in the point clouds in all three dimensions.

Figure 5.10 shows the centroids of both point clouds. The centroids are represented using crosses. The blue cross represents the centroid of the blue point cloud while the red cross represents the centroid of the red point cloud. (Princeton University, 2005)

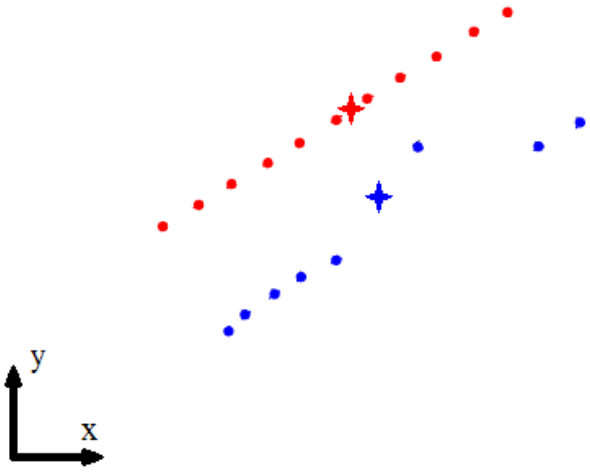


Figure 5.10 the centroids are computed and shown for each point cloud using crosses

Equation (7) represents a point in 3D space, while the centroid is computed with Equation (8).

$$c = (c_x \quad c_y \quad c_z) \tag{7}$$

$$c_x^k = \frac{1}{n^k} \sum_{i=1}^{n^k} a_{x_i}^k \tag{8}$$

In the above formulas, the variable k represents the point cloud number. It becomes clear that the centroid is computed for all three dimensions, even though only one is shown above. The centroid is used in the next part of the algorithm to move the point clouds.

After finding the centroids of the clouds, they are subtracted from all the points in each of the point clouds. The results of this can be seen in Figure 5.11, where the origin of the coordinate system is in middle of the point clouds. This makes it possible for the next step to compute the rotations of the point clouds. (Breitenmoser & Tishchenko, 2010)

The point clouds have moved close to each other as well. This is because not both of the centers of mass coincide with the origin of the reference frame.

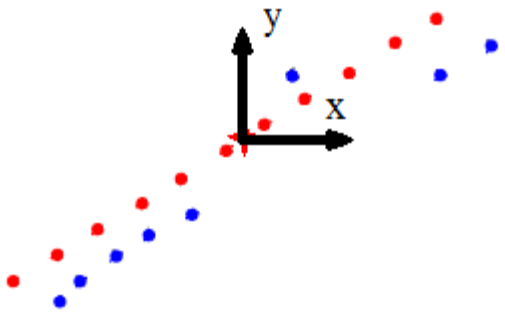


Figure 5.11 The point clouds are brought to the origin, so that the centroids overlap

In order to move the point clouds to the center of the figure, the centroids have to be subtracted from all the points in the point cloud. This is done using the equation (10), while equation (9) is a definition.

$$A_X = \forall a_{x_i} \tag{9}$$

$$A' = \begin{cases} A'_x = A_x - c_x^A \\ A'_y = A_y - c_y^A \\ A'_z = A_z - c_z^A \end{cases} \tag{10}$$

Applying this method to all the point clouds will bring them in the center of the coordinate system.

Now that the point clouds are both in the center of the reference frame, the rotational matrix can be computed. The rotational matrix is computed by applying the singular value decomposition function to the point clouds and using the results of the singular value decomposition to obtain the transformation matrix. (Breitenmoser & Tishchenko, 2010)

However, the Singular Value Decomposition (SVD) uses as input a 3x3 matrix which is not available at this point in the algorithm. Therefore, in order to obtain the required matrix, this step starts by multiplying the point clouds with each other to obtain a 3x3 matrix which is used when computing the SVD function.

After the results from the SVD function are obtained, they are used to compute the rotation matrix in the same way as it would be computed for the simple ICP algorithm.

Each point cloud has several points and each point has three dimensions. If the point clouds are treated as matrices where the number of rows represent the points and columns represent dimensions as given in Equation (11):

$$A = \begin{bmatrix} A_{x_1} & A_{y_1} & A_{z_1} \\ A_{x_2} & A_{y_2} & A_{z_2} \\ A_{x_3} & A_{y_3} & A_{z_3} \\ \vdots & \vdots & \vdots \\ A_{x_n} & A_{y_n} & A_{z_n} \end{bmatrix} \quad (11)$$

In order to generate the input for the SVD, the covariance matrix of the two point clouds needs to be computed using Equation (12) given blow:

$$D_{ij} = \sum_{k=1}^{n} A_{ki} B_{kj} \quad (12)$$

Where:

- D is the covariance matrix
- A and B are the matrices representing the point clouds which need to be aligned.
- i and j represent indices of the columns of the two point cloud matrices
- n is the number of points in the point clouds

Note: n is the number of points in both point clouds. The reason for which the point clouds have the same number of points is described in this section, above.

Matrix D is the input to the SVD matrix. For a detailed description of the SVD matrix Appendix B: SVD can be consulted.

The result of the SVD function is three matrices. Two of the matrices are used to obtain the required transformation matrix which rotates the second point cloud over the first point cloud. By definition, the result of the SVD has the form shown in Equation (13). (Jia, 2013)

$$M = U \sum V^T \quad (13)$$

Where:

- M is the 3x3 input matrix of the SVD function.
- U is a 3x3 real or complex unitary matrix.
- $\sum$ is a 3x3 nonnegative diagonal matrix with real numbers.
- V^T is the transpose of V and a 3x3 matrix unitary matrix. (Sorkine, 2009)

Assuming that the rotation were correct, the matrix M would be the identity matrix because the point clouds would be perfectly aligned. Furthermore, the matrix Σ is replaced by the rotation matrix R. This gives Equation (14):

$$I = V^T R U \quad (14)$$

To obtain the rotation matrix, Equation (15) needs to be used. (Sorkine, 2009)

$$R = V U^T \quad (15)$$

However, because of the nature of the process described, the optimal rotational matrix is obtained. This optimal matrix however, could contain reflections in addition to rotations. To determine if there is a rotation, Equation (16) is used:

$$\begin{cases} \det(VU^T) = -1 & contains\ reflections \\ \det(VU^T) = 1 & no\ reflections \end{cases} \quad (16)$$

Therefore, to make up for rotations as well as reflections, the formula given below (in Equation (17))is used which reflects the rotation matrix in case $\det(VU^T) = -1$ and leaves it the same otherwise.

$$R = V \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & det(VU^T) \end{pmatrix} U^T \quad (17)$$

After computing the rotation of the point clouds, the translation needs to be computed. The translation is computed by applying the rotation to the correct point cloud in Figure 5.10 and then subtracting the values of the two centers of mass. This is therefore the correct translation of the entire point cloud. Together, the translation and rotation, computed make up the complete transformation matrix of the point cloud.

The translation is computed using Equation (18):

$$t = c^A - Rc^B \quad (18)$$

Where:

- A and B represent the ponint clouds which are in the algorithm
- c represents the centroid of a certain point cloud
- R is the rotation matrix computed above.

The results of applying the rotation to Figure 5.11 can be seen in Figure 5.12 where the point clouds have been aligned using the transformation matrix.

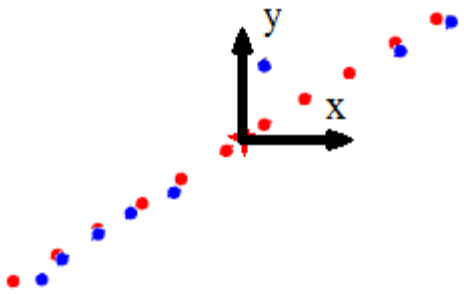


Figure 5.12 The rotations are applied to the point clouds and not they overlap.

5. 5. ICP VARIATIONS

There are many variations of the ICP algorithm. Since the normal ICP performed quite slowly on large point clouds, many variations have been made to the algorithm to make it run faster and more accurately. Some of the approaches will be described in this section.(Rusinkiewicz & Levoy, 2001)

5. 5. 1. CORRESPONDENCE SEARCHES

The bottleneck in the ICP algorithm is the correspondence searching algorithm. Traditionally, this part of the ICP finds the nearest neighbor by computing the distance between two points. This process can be very time-consuming for the computer because applying Pythagoras theorem to two point clouds of 1000 points each would result in one million applications of the theorem. For this purpose other methods have been searched. (Rusinkiewicz & Levoy, 2001)

The kd-Tree search is so far the fastest approach. A kd Tree is a k dimensional tree where the distance to a point can be found much faster than using Pythagoras theorem. In this case, the search space has three dimensions, so the result would be a 3d Tree search. Generating the kd Tree is the longest process, but it is still much faster than the initial method. (Kjer & Wilm, 2010)

5. 5. 2. NUMBER OF POINTS

Another method frequently used to make the algorithm run faster is to use a fraction of the points when searching for correspondences. This means that fewer computations are being made and also helps the algorithm avoid a local minima. A local minima can easily be reached by using all points, however if random points are being used, and each iteration different points will be used, the chances of reaching a local minima are much lower and the chance of reaching a global minima are therefore much larger than initially. (Pulli, 1998)

5. 5. 3. WEIGHING

A very frequently used method is to weigh the points. Depending on the scanner, some of the points which are obtained could have large errors in their values. If these points are weighed as much as any other point, then the algorithm will assume that there are no errors and settle in a local minimum.(Kjer & Wilm, 2010)

By weighing points based on different functions, it is possible to give some points more importance than other points. It is also possible to exclude some of the points from the computation completely. This approach gives the algorithm an “intelligent” edge as the algorithm will select which points it will use and which points it can trust as having a “correct” value. (Rusinkiewicz & Levoy, 2001)

5. 5. 4. POINT CLOUD FLEXIBILITY

While some algorithms take the point clouds as being rigid, and only change the alignment, others consider the point clouds to be flexible and able to bend.(Pulli, 1998)

The most common type of ICP algorithm will use rigid clouds and shift those clouds. However, if the scanned object is moving, there could be alignment problems. This is because scans are not taken instantly, and while the scanner is scanning one part of the object, the object rotates or moves. This causes the second part of the scan to appear in another place. And hence the scan will be distorted. In order to

compensate for this, some ICP algorithms make use of non-rigid point clouds. This means that even if two point clouds are not perfect matches, they can be warped by the algorithm so that they will fit perfectly. (Amberg & Vetter, 2003)(Brown, 2004)

6 STATE OF THE ART

This chapter will discuss progress that has already been done in this field by other people around the world. It will describe the algorithms which have already been made and what they can achieve. Moreover, some important properties of the algorithms discovered in literature will be discussed. These properties will be present in the algorithm developed for this thesis as well, however, they could be obtained in different ways than described in the literature.

6. 1. DESCRIPTION

Many algorithms and types of model reconstruction techniques have been examined for this thesis in order to have a better understanding this branch of technology. Of all the literature studied for this thesis, two algorithms stand out as exceptionally good at what they are meant to do.

One of the algorithms is Simultaneous Localization And Mapping (SLAM) based on path optimization. This method uses the path and the distances between certain features to optimize the trajectory of the sensor around the target object. By optimizing the path, a good reconstruction of the target object is achieved by the algorithm.

SLAM algorithms are a popular choice for robotics enthusiasts as they offer a fast and efficient approach to determine the position of a robot in 2D space. This is because a room can be seen as simply a 2D floor plan on which the robot can navigate.

The second algorithm is an Iterative Closest Point (ICP) based approach which aligns multiple point clouds to each other based on certain rules which connect the points of all the point clouds.

Both of these methods will be described in more detail. This will allow the reader to better understand how the methods work and why they are not sufficient to solve the problem at hand.

This should serve as an introduction to the workings of such algorithms and the basic principles that go behind them.

6. 2. SLAM

The general problem that needs to be solved using this algorithm is to rebuild a map of the surrounding environment by using range scanners which only see parts of the world.

The scanners used are called horizontal range scanners. These types of scanners sense the environment and give back a series of range measurements and the direction from which these measurements come from. Such scanners have been known to be used for robot positioning if the surrounding environment is known. Since one scan gives only a small portion of the world, multiple scans can be used to reconstruct the model of the surrounding world.

Scanners can be either sonar or laser scanners which use different means of obtaining distances in the real world.

The essential issue of this problem is aligning the scans to obtain one consistent model of the environment, however, in reality, there are many odometry errors which accumulate after each successful

scan. These errors cause the model to diverge unless it is processed. The processing has the goal of reducing the errors caused by odometry measurements by the robot's sensors.

The most common approach used to rebuild the world model is to incrementally add new data of the world while integrating the data to already existing data. After a new set of data is obtained, it is aligned to the previous set of data to form a global model which is expanded with each new piece of information.

The most often used integrators are either a Kalman filter or an averaging method. The main problem is ,as mentioned before, that after several scans have been added, the model becomes inconsistent and it diverges. Moreover, if the scans are combined, then the errors become permanent and therefore impossible to fix. (Lu & Milios, 1997)

To avoid all the problems described above, the approach uses all the scans independently without merging them. This means that the model can be improved infinitely as the individual scans are free to move relative to each other. To achieve this, the algorithm remembers all the positions of the robot in one global reference frame. However, the errors in positions are also remembered in the global reference frame. However, by not combining the point clouds, the previous positions can be improved using newer data points. The old positions are therefore updated to better match the global model. This allows the algorithm to take all the positions into account simultaneously and to use the connections between the positions to obtain a correct model reconstruction of the space around itself.

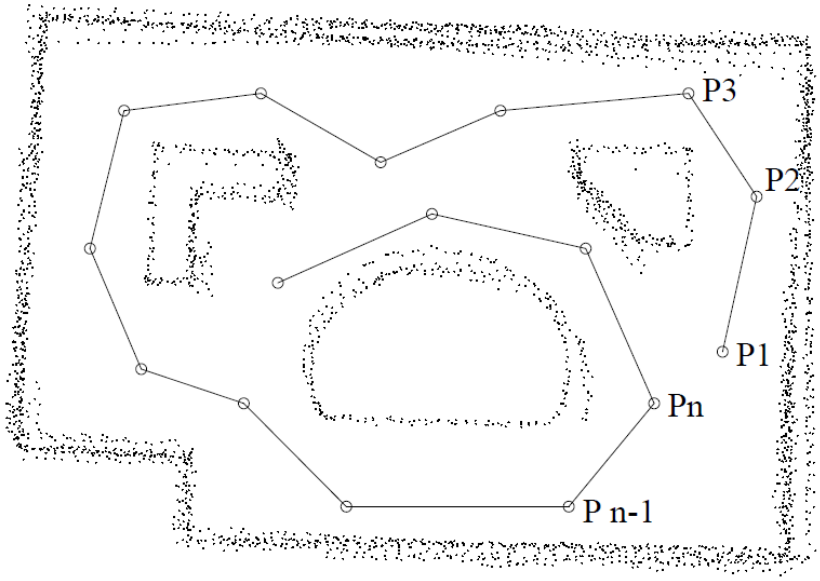


Figure 6.1 A raw model of a room obtained by the robot. The accumulation of errors is clearly visible by the end of the route.

Figure 6.1 shows the result of a robot going around a room. The scans are all plotted in a global reference system. This is the raw output of the sensor added sequentially as the robot progresses through the room. Points denoted as "P" in the figure represent positions where the robot stopped to scan the surroundings.

The reason for which the robot stops is because if the scans are made on the fly, the resulting image will be distorted due to the motion of the robot. This will therefore create shapes which cannot be aligned perfectly to any other shapes. As a result, it is necessary for the robot to stop in certain positions to measure the surrounding space.

The next step is to construct a network from the different positions. This network will connect positions which have a direct "line of sight" to other positions. This can be seen in Figure 6.2 where the dotted lines represent the connections between different positions. These connections in turn form a network of different positions and estimations of positions based on multiple scans of the surrounding environment.

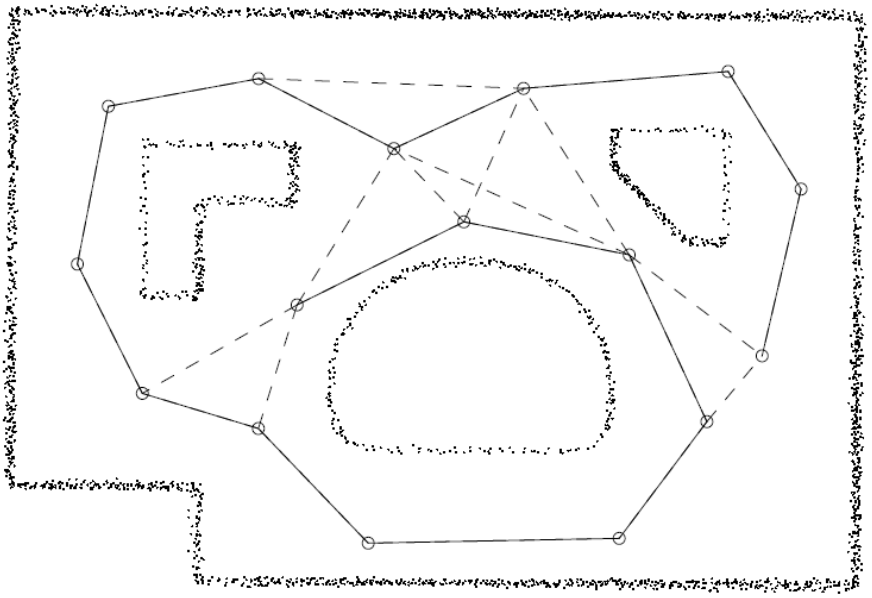


Figure 6.2 A reconstructed model of a room. The reconstruction process uses connections between positions to optimize the surrounding environment.(Lu & Milios, 1997)

Connections established earlier between the positions are used to optimize the positions. The optimization is done by taking into account the positions where the robot thinks it is for each measurement and compares that to the positions where it thinks it is in different measurements. Then it computes the distances between all these positions and updates all the estimates at once. The method gives increasing accuracy with increasing number of scans added to the system. Moreover, because the positions are updated on a global level, even the previously aligned positions are improved and not just the new ones.

As Figure 6.2 also shows the improvement in the reconstruction of the environment can be quite significant. It is clearly visible that Figure 6.2 contains a room with three features in it, and the edges of all the features and walls are clearly drawn.

The results after the reconstruction are much better than the ones shown in Figure 6.1 and this is done by the algorithm that is meant to optimize the trajectory of the object.

6. 3. ORTHOGONAL PROCRUSTES

The second approach which is considered very important for this thesis makes use of an Iterative Closest Point algorithm to align multiple point clouds. The goal of the algorithm is to extract a model from 3 – dimensional measurement points.

The algorithm described in this paper is an optimization problem in which the solution gives orthogonal transformation matrices between two matrices minimizes the least square of the residuals(Toldo,

Informatica, Beinat, Crosilla, & Cotonificio, 2007). The least square residuals formula is shown below in Equation (19).

$$E = cAR + jt^T - B \quad (19)$$

Where:

- E is the error of the problem which needs to be minimized.
- A, B are both matrices of equal size. These matrices need to be matched to eachother.
- R is the rotation matrix by which A is multiplied to obtain a result as close to B as possible
- c is a scaling factor
- j is a unit vector
- t is a k dimensional translation vector for matrix A

Unlike the previously described algorithm, this one works by using the measurements themselves to obtain a model of the world instead of using the path coordinates to do the optimization. The approach works by first finding correspondences between points from one measurement scan to every other scans. This is done for every scan in the model, which results in a huge amount of connections between he points of the model.

The connections are then used to extract features of the point clouds which are shared across multiple point clouds. The features are found based rules of connections between different points.

Finally, the connections are used to align scans from the model to each other and obtain the improved model. This is made possible using the ICP algorithm principle applied to all the scans in the model.

6. 3. 1. CORRESPONDENCES

The approach described in this paper uses a different method to obtain point connections between different point clouds. This different approach uses a Delaunay Triangulation on all the point clouds in the model.

Each point cloud is triangulated in order to make it easier to search the point clouds for points. The triangulated points are then used by the rest of the point clouds to search for the closest points. A representation of this can be seen below in Figure 6.3 where a Delaunay Triangulation is shown on a simple point cloud. A more in-depth explanation of the Delaunay Triangulation can be read in Appendix C: Delaunay Triangulation.

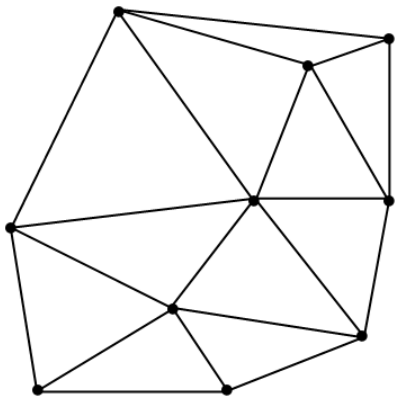


Figure 6.3 Delaunday Triangulation of a point cloud with 10 points (Delaunay, 2006)

After triangulating each point cloud, a search algorithm is used to find correspondences to other point clouds. The Delaunay triangulation helps in this respect because it creates a "map" on the point clouds which makes it easy for a search algorithm to pass from one point to another and find the closest point at a reasonable speed. (Toldo et al., 2007)

If the model has three point clouds denoted A, B and C, the correspondences to point cloud A are found by taking the triangulation of point A and the non-triangulated B and C point clouds. The non-triangulated point clouds are then used to search in the triangulated point clouds. After all the correspondences are found, the next triangulated point cloud is taken (point cloud B) and point clouds A and C are non-triangulated. The connections to point cloud B are now created. Finally the triangulation of point cloud C is taken and the connections to this point cloud are made using the non-triangulated versions of point clouds A and B. This approach can be applied to any number of point clouds as it is scalable.

Instead of using a threshold, this approach uses another approach to ensure that the connected points are not too far from each other. The approach takes into account only matches of points which are mutually nearest neighbors. This means that point $p \in A$ is mutually nearest neighbor to $q \in B$ if the closest point of q is p and the closest point of p is q. This approach offers the possibility to disregard any thresholds because when selecting the mutually nearest neighbors, which ensures that connections which are further away are automatically excluded from the computation. A graphical example of this process is given in Figure 6.4.

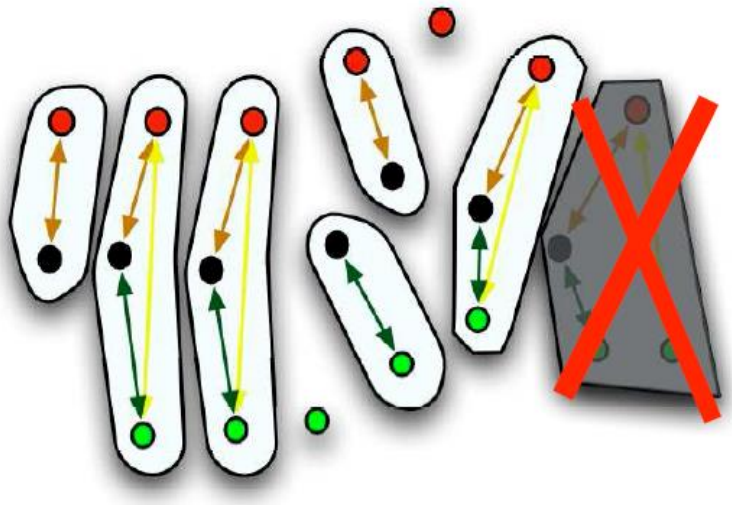


Figure 6.4 Mutually nearest neighbors selecting process(Toldo et al., 2007)

Figure 6.4 shows the connections between three point clouds. In the figure above, the different point clouds are differentiated by colors. The red dots belong to one point cloud while the black ones belong to a second point cloud and the green dots belong to a third point cloud. The arrows represent connections between the points of each point cloud. It can be seen that some points have multiple connections while other have only one connection. Points which are mutually connected are encircled with a black line, which helps the reader see groups/families of points.

Figure 6.4 also shows an example of points which, although have connections, are not inter connected amongst themselves. This example is the group to the right crossed out with a red cross. The points in that

selection are not connected amongst each other which means that a "family" of points is not formed. A family of points is formed only when every point is connected to every other point such as the families to the left which are not crossed out. It is also possible to see points which have absolutely no connections, these points are also ignored by the algorithm.

6. 3. 2. REGISTRATION

Registration is the part of the algorithm which uses the connections established earlier to move point clouds over each other until they match. Unlike the ICP method, this approach aligns all the point clouds at the same time instead of aligning them one at a time.

However, since it is easiest to align point clouds using ICP algorithms, this approach also uses an ICP to align the point clouds. However, the way in which the ICP is used differs greatly than the traditional way of using the ICP algorithm.

Registration takes place by selecting point clouds one by one and aligning them to the rest of the point clouds. The alignment is done by combining all the point groups and forming a point cloud which represents the entire model. Combination of the groups is done by computing the centroids of the points in each group. The centroids of the groups are computed using Equation (20):

$$c_i = \frac{1}{n_i} \sum_{j=1}^{n_i} P_j^i \qquad (20)$$

Where:

- c_i is the centroid of group i
- n_i is the number of points in group i
- P_j^i is the j[th] point from group i

This process can be easily seen in Figure 6.5 where each group of points from Figure 6.4 contains an extra grey point at the center. This extra point is the centroid of each group of points and it is computed using Equation (20)

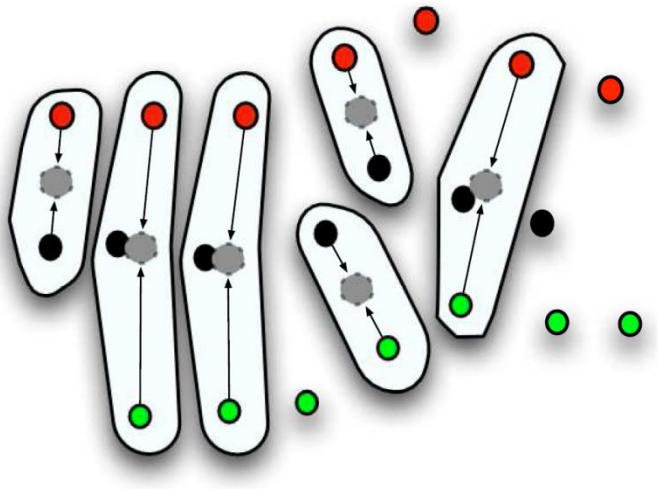


Figure 6.5 Centroids of various groups of points(Toldo et al., 2007)

After all the centroids from all the groups have been computed, it is possible to align the point clouds using the ICP algorithm.

Alignment is done by taking one point cloud and the centroids of all the groups. The centroids of all the groups are treated as a single point cloud for this part of the algorithm. The alignment is thus done by a simple ICP algorithm applied to all the point clouds separately.

For example, assume there are three point clouds, A, B and C, which have a centroid point cloud called C'. The registration will first align point cloud A to C', then point cloud B to C' and finally C to C'.

In the next iteration, the C' pointcloud is regenerated and the rest of the point clouds are realigned to the centoids.

6. 4. RESULTS

As mentioned in previous chapters, this master thesis has several goals which need to be addressed. In order to achieve the goals of this thesis, a new method for computing the position of the orbiter relative to the target satellite needs to be written.

The solution for the master thesis will be in some aspects similar to the two methods given above. While in other aspects it will not resemble the two methods at all. A comparison of the two methods described in the literature can be seen in Table 2.

Table 2 Comprison of the two main literature methods

SLAM	**Orthogonal Procustes**
Can add new pointcloudus while the algorithm runs	Algorithm must start with all point clouds
Optimizes the path	Optimizes the alignment
Uses all points	Uses a part of the points
Fast enough to be real time	Too slow to be real time
The existence of a surrounding environment is necessary	The existence of a surrounding environment is not necessary
Model does not have to be prealigned	Model needs to be prealigned

From experiments done with the algorithms above, it turns out that the algorithms also have many cases in which they do not work as intended. This results in a divergence of the results and a bad alignment/resulting model.

A simple example of this appeared when testing the Orthogonal Procustes algorithm on a model. After the alignment, it turned out that one of the point clouds was very poorly aligned to the rest of the model. Even though there was an initial prealignment of the model. In the end, the algorithm was not good enough to correctly align the clouds. This is one of the main problems which needed to be fixed when designing the new algorithm. The implementation must be more robust than the one described above.

The incorrect alignment which was discussed in the previous paragraph is shown below in Figure 6.6. In Figure 6.6 each color represents a different point cloud.

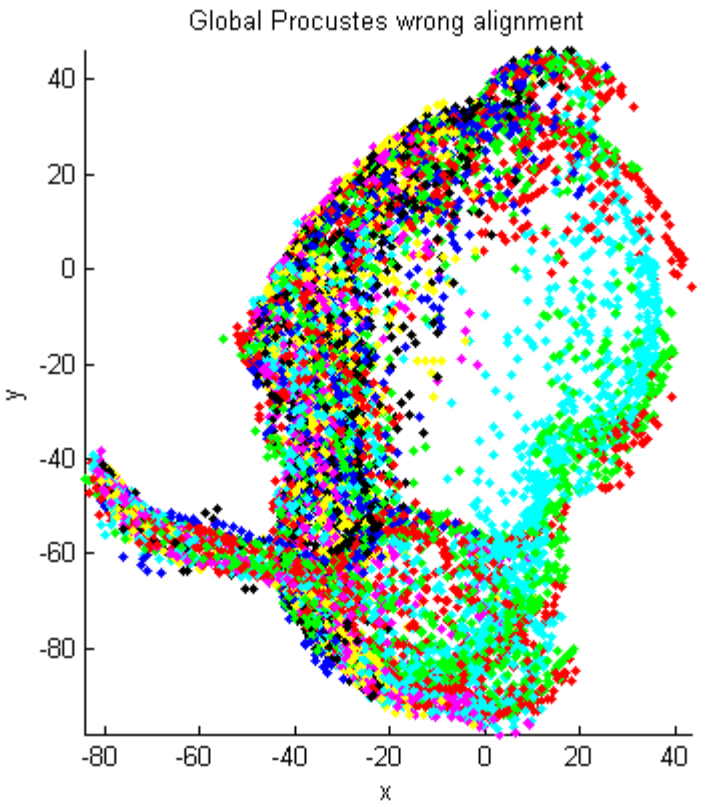


Figure 6.6 Top view of a model that was not correctly aligned

After seeing the image above, it becomes clear that some of the point clouds are incorrectly aligned even after several iterations of the algorithm. The errors in the algorithm can be seen at the top right and bottom right of the model where the edges are discontinuous. This is caused by forces between the points of the model which are pulling the point clouds towards the center of the image. This error is caused by incorrectly selecting the features of the model. It should be noted that increasing the number of iterations does not improve the result of the final model.

The goal of this thesis is to create an algorithm which can:

- Add new point clouds while the algorithm runs
- Optimizes the alignment of point clouds
- Uses either all points or a part of the points
- Be fast enough to operate in real-time
- Reconstruct a model without the existence of a surrounding environment
- Reconstruct a model that is not prealigned
- Can use points in a smart way in order to reduce alignment errors
- Flexible with respect to point clouds and the number of points which go in the model
- Flexible with respect to alignment parameters (parameters which can be set to different values in order to allow the algorithm align different types of models.)

These features which are listed above require the algorithm to be completely different from the algorithms described above, and it must combine the best aspects of the two algorithms while improving on the general reconstruction aspect.

Page intentionally left blank

7 GLOBAL ITERATIVE CLOSEST POINT

In this chapter, the main algorithm developed during the duration of the thesis is discussed. As the main part of this thesis, this chapter will also go into the most amount of detail. It begins by describing the algorithm from a very high level perspective, and as the chapter progresses, increasingly more low level information will be given. Low level information in this case is the specifics on how it functions and the mathematical theory which goes behind the workings of the algorithm. Improvements made on the algorithm are discussed near the end of this chapter, as well as the reasons for the improvements.

7. 1. DESCRIPTION

As described in previous chapters, the ICP algorithm is an algorithm which takes two point clouds and tries to make them overlap as well as possible. There are errors however, and these errors accumulate with the addition of new point clouds. If there are many point clouds to be added, then the error could be so large that it would be very impractical to use such an algorithm to reconstruct a model. Therefore, a more general algorithm is needed which takes all the clouds into consideration at the same time. Such an algorithm is called a GICP or a Generalized Iterative Closest Point algorithm.

The GICP developed in this master thesis is a real time algorithm which means that it can work while new point clouds are added to the algorithm. For example: if the user starts with two point clouds and then has a third point cloud that they want to add to the algorithm, they will not need to stop the algorithm, instead, the algorithm will include the new point cloud directly into the computations. Moreover, unlike the ICP it will take all the point clouds into considerations during computation and not just the last two (as is the case for ICP).

One important aspect of the algorithm is that it is built in a semi-modular fashion. This is because it is a very complex algorithm to understand and maintain and by making it more modular means that it is more clearly structured and allows the user to have a simpler understanding of the code. The modularity of the algorithm also makes it easier to maintain. A modular algorithm is a type of software design technique which separates the code of a program into independent modules. Each module is therefore specialized to solve one specific aspect of the problem. This makes debugging much easier as the modules can be isolated and bugs can be found much faster. In a non-modular computer code, including many techniques into the software will increase its complexity exponentially due to the difficulty of the programmer to understand and debug the code. However, modularity means that many different techniques can be included to work together without increasing the complexity of the program. As they are easy to interpret once they have been implemented and can be easily taken out in order to simplify the code if the user desires.

Figure 7.1 below shows a diagram of the entire program. This diagram clearly shows the workings of the program as well as the core parts and the additional modules. Later in the chapter these additional modules will be further elaborated. A selection of modules which can be implemented will be given and the selected modules will be described further and a motivation for their selection and implementation will also be given.

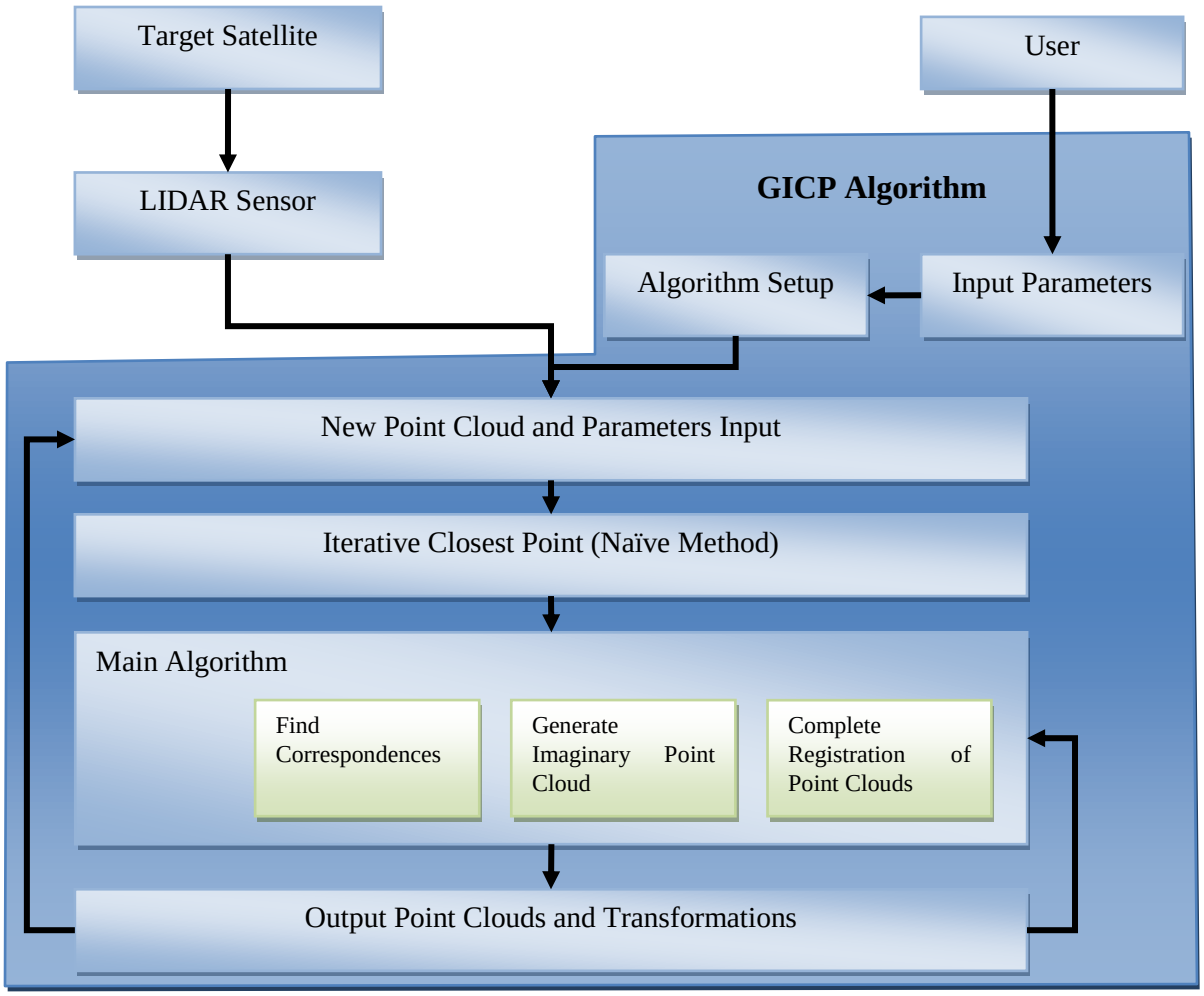


Figure 7.1 A block diagram of the algorithm and the environment in which it functions.

Figure 7.1 describes the whole GICP algorithm and the factors which influence its performance and functions.

Everything starts from the target satellite which the algorithm is supposed to examine. Information from the target satellite goes to the LIDAR through the form of laser light. This information is then processed by the LIDAR scanner and useful information for the algorithm is prepared.

At the same time, a user needs to input some parameters for the algorithm. These parameters will help the algorithm better align the target point clouds. If these parameters are not correct, the algorithm could diverge and produce a very bad result which will be impossible to use for reconstruction of the orbit of the satellite.

Next, information from both these sources (LIDAR and user) is inserted in the algorithm. The information from the user is first used to set some initialization parameters for the algorithm. These parameters define the mode in which the algorithm functions. Since the algorithm is very flexible, there are many ways in which to set up the algorithm. These properties are discussed further in Chapter 9.

Having set up the algorithm, information from the LIDAR can start to be processed. Because the algorithm runs in real time, information from the LIDAR is continuously inserted in the algorithm, and the algorithm processes only the information which is available. As discussed in Chapter 4, the information that comes from the LIDAR is stored as point clouds. Each time a new point cloud is added, no more. However, it is possible that there is no new information added. This can happen if the algorithm finishes processing before a new point cloud is available from the LIDAR. In this case, the algorithm can either continue processing the information that it already has, or it can stop and wait for more information to be added.

After a new point cloud is added to the algorithm, some of the parameters set up by the user are used to perform operations on the point cloud in order to store it in a format that is easier for the algorithm to understand.

In the next step, the Naïve Method is used to align the last point cloud to the one before it. This is a crucial step because most of the alignment is done in here. The rest of the code is to further align and to ensure that errors do not stack up.

After applying the Naïve Method, all of the point clouds are transferred to the next phase of the algorithm. The next phase is the most computationally expensive one and this is where all the point clouds are aligned at the same time instead of aligning just two of the point clouds. This phase starts off by finding correspondences between all the point clouds. The number of correspondences grows exponentially with the number of point clouds (see Chapter 7. 2. 2) which results in a huge number of computations as the algorithm progresses. There are many smaller modules which are also activated in this phase of the algorithm. These modules have the sole purpose of simplifying the information and reducing the number of computations that the algorithm makes. Sometimes accuracy is traded off in favor of computational time, depending on the geometry of the target.

In the next step, an "Imaginary Point Cloud" is generated. This Imaginary Point Cloud is used as an assisting object for aligning all the point clouds. More information on why this point cloud is needed and how it functions is available in Chapter 7. 4.

Finally, the point clouds are all aligned to each other at the same time. This step of the algorithm is very similar to the registration function described in Chapter 6. 3. 2.

After having aligned all the point clouds, the information is rearranged and stored in a format which is easier to use once a new point cloud is added. Also in this phase, the algorithm ensures that the reference frames are stationary and that it always knows its position relative to the target. This is a very important aspect to take into account in order to accurately reconstruct the trajectory of the servicer with respect to the target satellite.

7. 2. OVERVIEW

An overview of the algorithm will be given in this section. It will discuss the basic working principle and the mentality that goes behind the workings of the Global Iterative Closest Point Algorithm.

It first should be noted that the algorithm only works with a minimum of two point clouds. This is because with one point cloud there is nothing to align. However, once there are two or more, the algorithm can begin functioning.

7. 2. 1. LOOPS

As can be seen in Figure 7.1 there are two main loops in the algorithm. Each loop has a specific task which will be briefly discussed in this section.

The first loop which starts at the Outputs and goes back to the Input of a new Point Cloud is called the "Major Loop". This loop has the function of adding new point clouds to the algorithm. Every time it reaches the top (Input phase), one point cloud from the LIDAR scanner is taken and incorporated into the algorithm. It is possible to take either the latest point cloud from the LIDAR, or to take a point cloud which was saved previously. All the point clouds which are added to the algorithm will remain in the algorithm until the algorithm stops working. This ensures that no information that has been added is lost in the process of aligning new point clouds to the rest of the model.

The second important loop starts at the Outputs again and ends in the main algorithm just after the Naïve Method. This loop has the task of aligning the newly added point cloud to the rest of the model. It is known that the algorithm does not align a point cloud in just one iteration of the algorithm, and that instead, it needs several consecutive iterations to correctly align the new point cloud to the rest of the model. This loop is different from the Major Loop because before a new point cloud is added, the one which was added previously needs to be sufficiently well aligned otherwise the reconstructed 3D model will diverge.

All the parameters which control these two loops can be set by the user before the algorithm begins functioning. This allows the user to have a direct link to this aspect of the algorithm. It is very important for the values which control these loops to be set correctly otherwise the algorithm might not converge. However, the values can be picked such that the algorithm computes more than it is needed and the required time is too long. A balance needs to be found between choosing the speed of computation and the degree of convergence of the 3D model.

7. 2. 2. PERFORMANCE

As new point clouds are added to the algorithm, there are not point clouds which are being taken out of the algorithm. This means that the amount of information that the computer manages increases constantly. Moreover, since each point cloud forms connections to every other point cloud, the complexity of the problem grows exponentially. This can be easily understood with a simple example.

In the first loop, there are two point clouds in the algorithm. This results in two inter-cloud connections. One going from point cloud one to point cloud two, and a second connection from point cloud two to point cloud one. This means that the algorithm needs to compute only two connections. Now if another point cloud is added to the algorithm, there are a total of three point clouds in the Global Iterative Closest Point. And since each point cloud connects to every other point cloud (except itself) there are a total of six connections.

A formula that gives the number of connections can be seen below in Equation (21):

$$C = n^2 - n \qquad (21)$$

Where:

- C represents the total number of connections between the point clouds
- n represents the number of point clouds in the algorithm

It can now very easily be deduced that if the number of point clouds in the algorithm is equal to 50, there are a total of 2450 connections which need to be considered by the algorithm. The number of point clouds that go into the algorithm.

7. 2. 3. NAÏVE METHOD

The Naïve Method is a simple application of the ICP algorithm over two point clouds. Because the global optimization algorithm (GICP) deals with many point clouds at the same time, convergence is much slower. Therefore, the 3D model needs to be prealigned by some other algorithm before the Global Iterative Closest Point can improve the model.

The prealignment happens by using an Iterative Closest Point(ICP) algorithm to align the last two point clouds which have been added. If point cloud n has been added in this iteration, then the algorithm aligns point clouds n and $n-1$. This process aligns the new point cloud to the rest of the model and allows the global optimization to have a good starting point for further alignment.

This process of aligning point clouds in pairs can be applied to the entire model, however, without applying the Global Iterative Closest Point, there will be an accumulation of errors which will result in huge errors by the end of the algorithm. The errors can get so large, that it will become impossible to close the gap between the first point cloud and the last point cloud. This approach is called a "Naïve Method". A more detailed explanation of why this is not possible can be found in chapter 9. 3 ICP.

7. 2. 4. CORRESPONDENCE MATRIX

All the correspondences are stored in the `CM` variable. `CM` is a matrix which stores all the correspondences between the points in the model. The exact format of this matrix is extremely important as it allows the reader to follow the algorithm.

Note: `CM` does not need to be a matrix for the algorithm to be successful, other index storing formats can also be used. However, for the purpose of this thesis, it has been determined that the simplest way to explain the algorithm is by using a large matrix to store the indices of the inter connected points.

FORMAT

`CM` is an nxm matrix where n is the number of points in the entire 3D model (sum of all points from all point clouds), and m is the number of point clouds in the 3D model. Usually the matrix is very large as the number of points in the model can be in the order of tens of thousands and the point clouds can be in the dozens. This translates into a matrix which has tens of thousands of lines and dozens of columns.

In order to further simplify the understanding of this Correspondence Matrix, a different notation will be used. Assume that CM can be simplified to a square (mxm) matrix. In this case CM can be seen as a matrix of vectors, where each vector contains indices from one point cloud. This can be seen below in Equation (22).

$$CM = \begin{bmatrix} I_1 & I_2 & I_3 \\ I_4 & I_5 & I_6 \\ I_7 & I_8 & I_9 \end{bmatrix} \quad (22)$$

Where:

- $I_{1\ldots9}$ represent vectors of indices of points which are used for storing correspondences.
- CM is the matrix storing these indices.

In the notation given above in Equation (22), there are only three point clouds in the 3D model. And column one will always contain vectors which contain indices from point cloud one. Column two will always contain vectors that store indices of points from point cloud two, and so on. This means that each time an index from a given column is accessed, it can be directly determined which point cloud the index belongs to.

Another property of the Correspondence Matrix is that the index vectors located on the diagonal of the matrix always contain the unaltered indices of their respective point clouds. This means that I_1, I_5 and I_9 contain the indices of their respective point clouds in ascending order.

READING THE CM

As mentioned before, the purpose of this matrix is to store correspondences between all the points in the point cloud. Thus, it should be possible for a human to interpret the matrix. In its current design, it is very easy to read the Correspondence Matrix and to determine connections between points from this matrix.

In order to read correspondences from the matrix, it is much easier of the correspondences are read row by row. Therefore, the first row of the matrix in Equation (22) is taken away from the matrix. The row has the form $[I_1 \quad I_2 \quad I_3]$ where I_1 is the vector located on the diagonal of the matrix. This means that all the indices given in the other vectors represent connections from other point clouds to I_1. Vector I_2 contains the connections of point cloud two (because it is located on column two) to point cloud one, while vector I_3 contains connections from point cloud three (because it is locate on column three) to point cloud one.

Similarly, in order to read connections to point cloud two, the second row is taken. The second row has the form$[I_4 \quad I_5 \quad I_6]$ where I_5 is the vector located on the diagonal of Equation (22). This means that in order to find the correspondences of point cloud one to point cloud two, it is necessary to look on column one at vector I_4. This vector stores indices of points which have correspondences to indices from point cloud two. In the same way, it is possible to find the correspondences of point cloud three to point cloud two, by simply looking at column three. Column three (I_6) stores the indices of the third point cloud which correspond to points located in the second point cloud.

Assume that there is a 3D model that contains m point clouds. Each point cloud contains n_j points where $1 \leq j \leq m$. It can also be said that the total number of points in the model (number of lines in the matrix) is equal to n. This is shown in Equation (23).

$$n = \sum_{j=1}^{m} n_j \qquad (23)$$

Now it is assumed that there is a function g which is described in Equation (24).

$$g(a_i) = bj; |a_i - b_j| = \min_k \{a_i - b_k | k = 1 \dots n_l\} \qquad (24)$$

Where:

- a_i represents the i[th] point from the current point cloud
- b_j represents the j[th] point from the second point cloud
- n_l number of points in point cloud l.

Now assume there is a function f defined in Equation (25).

$$f_A(B) = I; I = g(a_i | 1 \leq i \leq n_j) \qquad (25)$$

Where:

- A, B represent point clouds A and B respectively
- I represents a vector of indices with correspondences from point cloud A to point cloud B
- n_j the number of points in point cloud A

By substituting this equation in Equation (22), it is possible to rewrite the matrix as seen in Equation (26).

$$CM = \begin{bmatrix} I_1 & f_1(2) & f_1(3) \\ f_2(1) & I_5 & f_2(3) \\ f_3(1) & f_3(2) & I_9 \end{bmatrix} \quad (26)$$

Now it is clearly visible how the matrix can be read, and the mathematics that go behind the correspondence matrix.

Although it has been described that it is easier to view the correspondence matrix as a matrix of vectors, in reality, it is saved as a simple matrix. The vector visualization only helps humans have a better understanding of the matrix.

7. 3. CORRESPONDENCE FINDING

This subsection discusses the module which is responsible for finding correspondences. As this is the heart of any ICP/GICP algorithm, it is important to have a very good understanding of how it works and how it is used in the context of this implementation.

As discussed before, the GICP is a general version of the ICP. This means that it is capable of aligning multiple point clouds at the same time. To achieve this however, it needs to be able to find the correspondences of multiple point clouds at the same time in order to be able to analyze the alignment possibilities of each point cloud. Storing this amount of information in a logical and easy to understand way is also not an easy task as there are thousands of points in each point cloud. This translates to which have tens of thousands of connections when there are multiple point clouds. All these connections have to be saved perfectly in order to have a good result in the end. If the points are mismatched, the point clouds will be moving in the wrong directions which results in greater misalignments and thus works counterproductively.

In order to better understand part of code, a pseudo code will be used to describe how it functions and show all the processes which take place for correspondence selection. Next, the part of the algorithm dealing with correspondences will be discussed in more detail and more in depth information on how it functions is given, followed in the end by some mathematical theory and an example.

Correspondences

```
1.    Generate kd Trees for all point clouds
2.    For i = 1 to Number of Point Clouds
3.          Add All Indeces of Point Cloud i to CM
4.          For j = i+1 to Number of Point Clouds
5.                Find correspondences in kd for cloud i in j
6.                Find correspondences in kd for cloud j in i
7.                Add PointCloud(i).indexes to CM
8.                Add PointCloud(j).indexes to CM
9.          End
10.   End

CM – Correspondence Matrix
```

Code 2 The Pseudo code for of the Correspondence finding part of the algorithm. Only essential lines are being shown in this code block.

Code 2 shows the pseudo code of the correspondence finding component of the algorithm. This, as mentioned before, is one of the most important parts of the algorithm, and the way in which it functions will be described. In the first line, it can be seen that kd trees are being generated for all point clouds.

A kd Tree is a space partitioning data structure for points in a k-dimensional space. It is a means of storing points in a structure that makes it very fast to search for the closest points to any given point. This method is much faster than computing the distances between all points and then comparing the distances to find the lowest one. It functions as a simple search algorithm in multiple dimensions. The reason for using a kd Tree instead of a Delaunay Triangulation as is mentioned in Chapter 6, is because a kd Tree search is much faster and easier to implement than a Delaunay Triangulation.

Having generated a kd Tree of all the point clouds, it is now possible to go further with the algorithm and find correspondences of the points. For each point from every point cloud, a correspondence has to be discovered in every other point cloud. In the end, each point has several correspondences.

Assuming that there are three point clouds, of which the first and second have four points each, and the fifth has five points. The aim is to find correspondences between each point from each point cloud and write them in the Correspondence Matrix in such a way that is easy to understand and later to be used on in the algorithm. It should be noted that the CM is filled according to the code given in Code 2.

The following picture shows the point clouds which will be used.

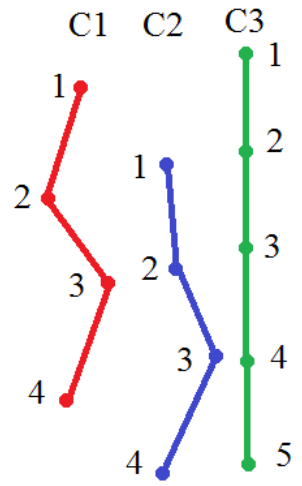


Figure 7.2 Three arbitrary point clouds

Figure 7.2 shows three point clouds which will be used for this example. The formula used to generate the CM matrix for a 3-point cloud model is given below in Equation (27).

$$CM = \begin{bmatrix} I_1 & f_1(2) & f_1(3) \\ f_2(1) & I_5 & f_2(3) \\ f_3(1) & f_3(2) & I_9 \end{bmatrix} \quad (27)$$

In Figure 7.2 there are a total of 13 points, which means that there will be 13 rows in the final CM matrix and there are only three point clouds which results in 3 columns. In order to make the CM matrix easier to understand, the elements which are part of the diagonal are written in bold.

At the beginning, the first element of the matrix (I_1) needs to be filled, this is also the first element of the diagonal and it is just a transcription of all the indices from the first point cloud in consecutive order. It can be easily seen in Figure 7.2 that the first point cloud contains four points with the indices given in Equation (28).

$$I_1 = [1 \quad 2 \quad 3 \quad 4] \quad (28)$$

These indices will be filled in first in the matrix, and because they are from I_1 and thus from the diagonal, they will be written in bald text so that it is easier to distinguish them.

Following the first element of the diagonal, which is saved in line three of Code 2, the second loop is started. In the second loop, a searching algorithm uses the kd trees generated in line one to search for the nearest neighbors of the points from other point clouds.

It should be noted that the second loop begins from point cloud i+1 which in this case is equal to 2, because i is still in the first iteration. Therefore, the following point cloud which is searched is point cloud two. The algorithm first computes the function $f_1(2)$ followed closely by function $f_2(1)$ using lines 5 and 6 from the algorithm. This gives the algorithm new indices to complete the Correspondence Matrix according to lines 7 and 8 from Code 2.

After completion of the second point cloud, the value of j (again from Code 2) is increased to 3. This results in the computation of $f_1(3)$ and $f_3(1)$ which are also saved in the CM matrix. Following this, j cannot increase anymore and the algorithm exits the second loop in line 9. The new Correspondence Matrix takes the form shown in Equation (29).

$$CM = \begin{bmatrix} \mathbf{1} & 1 & 1 \\ \mathbf{2} & 1 & 2 \\ \mathbf{3} & 2 & 3 \\ \mathbf{4} & 4 & 4 \\ 2 & () & () \\ 3 & () & () \\ 3 & () & () \\ 4 & () & () \\ 1 & () & () \\ 1 & () & () \\ 3 & () & () \\ 3 & () & () \\ 4 & () & () \end{bmatrix} \quad (29)$$

Now that the indices of some of the point clouds have been stored in the correspondence matrix, the second iteration of the larger loop can begin. In the next step, where i is equal to 2, the indices of the second point cloud are filled into I_5 in the same way as they were for I_1, and can be seen in Equation (30).

$$I_5 = [1 \quad 2 \quad 3 \quad 4] \quad (30)$$

This vector is then transferred to the correspondence matrix and written in bald to simplify the reading of the matrix. Now, the second loop is activated with $j = i + 1 = 3$. This means that the equations that need to be solved are $f_2(3)$ and $f_3(2)$. After completing these equations in lines 5 and 6, the indices are again saved in the Correspondence Matrix using lines 7 and 8 of Code 2. The results are given below in Equation (31).

$$CM = \begin{bmatrix} \mathbf{1} & 1 & 1 \\ \mathbf{2} & 1 & 2 \\ \mathbf{3} & 2 & 3 \\ \mathbf{4} & 4 & 4 \\ 2 & \mathbf{1} & 2 \\ 3 & \mathbf{2} & 3 \\ 3 & \mathbf{3} & 4 \\ 4 & \mathbf{4} & 5 \\ 1 & 1 & () \\ 1 & 1 & () \\ 3 & 2 & () \\ 3 & 3 & () \\ 4 & 4 & () \end{bmatrix} \quad (31)$$

Following this addition to the CM, the first “for” loop is activated a third and last time. Same as previous times from Equations (28) and (30), the indices of the third point cloud are updated to I_9 as shown in Equation (32). These indices are then placed in the CM matrix.

$$I_9 = [1 \quad 2 \quad 3 \quad 4 \quad 5] \quad (32)$$

This time however, the second “for” loop is not activated because $j = i + 1 = 4$ which is already larger than the number of point clouds. Therefore, in the end, the Correspondence Matrix is given in Equation (33).

$$CM = \begin{bmatrix} \mathbf{1} & 1 & 1 \\ \mathbf{2} & 1 & 2 \\ \mathbf{3} & 2 & 3 \\ \mathbf{4} & 4 & 4 \\ 2 & \mathbf{1} & 2 \\ 3 & \mathbf{2} & 3 \\ 3 & \mathbf{3} & 4 \\ 4 & \mathbf{4} & 5 \\ 1 & 1 & \mathbf{1} \\ 1 & 1 & \mathbf{2} \\ 3 & 2 & \mathbf{3} \\ 3 & 3 & \mathbf{4} \\ 4 & 4 & \mathbf{5} \end{bmatrix} \quad (33)$$

Now, the Correspondence Matrix given above has all the correspondences from the three point clouds given in Figure 7.2. It is a very large matrix and the complexity only increases with growing numbers of points and point clouds. It should be noted that this example is given solely for understanding purposes and that an actual problem would consist of tens of thousands of points and dozens of point clouds which will increase the complexity exponentially. (As it has been shown in section 7. 2. 2)

For an easier representation, the Figure 7.3shows what the connections look like for each point cloud in a visual representation of the CM matrix given above.

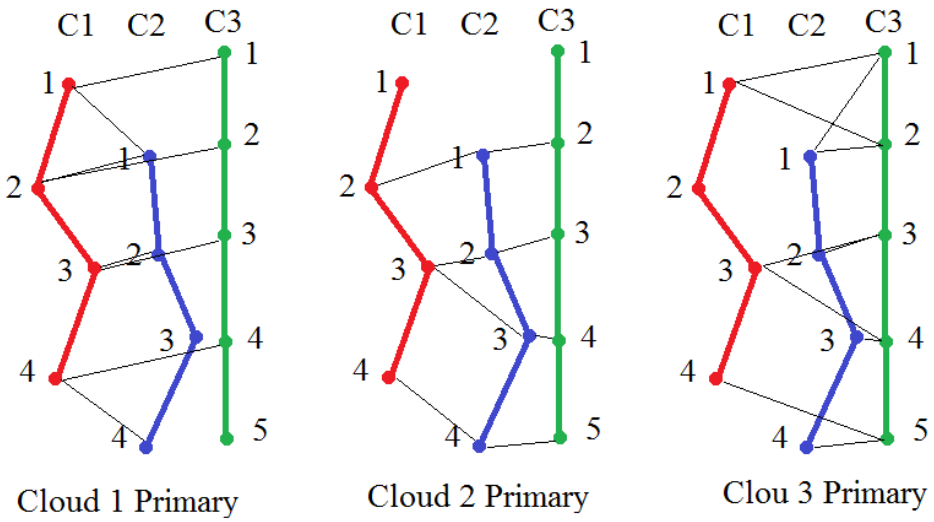


Figure 7.3 Visual representation of the Correspondence Matrix

Figure 7.3 gives a better understanding of the connections described above. At the end of this module, the output is a Correspondence Matrix which gives the indices of all the connections between the point clouds. This information is further refined and in subsequent steps of the General Iterative Closest Point algorithm.

7. 4. IMAGINARY POINT CLOUD

The “Imaginary Point Cloud” is a point cloud created with the help of the connections stored in the Correspondence Matrix. It is very important to generate this point cloud in order to have a good reconstruction of the 3D model in the end.

The “Imaginary Point Cloud” is a very large point cloud that is generated just before the registration part of the algorithm. This point cloud is a single point cloud which represents the entire 3D model of the target satellite.

Before generating the point cloud however, there are several functions applied to the Correspondence Matrix to reduce its size. These functions reduce the number of rows by deleting points with no connections and other points which are repeated.

Although in the ideal case, each point would have as many connections as there are point clouds in the model, there are functions which prevent some of the connections from being formed. These functions are placed in the Correspondence finding code and they are activated in specific cases. The exact functions will be discussed in Section 7. 6. However, because of these functions, the number of connections is slightly reduced after the Correspondence finding component of the algorithm.

After reducing the size of the Correspondence Matrix, it is possible to begin constructing the Imaginary Point Cloud.

Imaginary Point Cloud

```
1. Reduce Correspondence Matrix
2. For every point in the Correspondence Matrix
3.     Compute centroid based on connections
4.     Add point to ImagPC
5. end


ImagPC – Imaginary Point Cloud
```

Code 3 The imaginary point cloud pseudo code

Code 3 represents a simplified version for the pseudo code of the Imaginary Point Cloud generating component of the algorithm.

What this component of the algorithm does in essence, is it reduces the Correspondence matrix from an nxm matrix to an $nx2$ matrix where there are just two huge point clouds which are aligned in “patches”. All the raw point clouds are combined together to form one large point cloud which is broken down again when the model is realigned. This process will be explained below by continuing the example given in Section 7. 3.

This code starts with the Correspondence Matrix as input from the previous block of code. The Correspondence Matrix is then reduced to have fewer points in order to have a better organization of the information in the algorithm. (the reduced correspondence matrix is shown in Equation (34)) The input matrix is given below, as well as the reduced matrix right next to it. The logic behind these reductions is beyond the scope of this section and it will be explained further in Chapter 7. 6.

$$CM_{new} = simplify\left(\begin{bmatrix} \mathbf{1} & 1 & 1 \\ \mathbf{2} & 1 & 2 \\ \mathbf{3} & 2 & 3 \\ \mathbf{4} & 4 & 4 \\ 2 & \mathbf{1} & 2 \\ 3 & \mathbf{2} & 3 \\ 3 & \mathbf{3} & 4 \\ 4 & \mathbf{4} & 5 \\ 1 & 1 & \mathbf{1} \\ 1 & 1 & \mathbf{2} \\ 3 & 2 & \mathbf{3} \\ 3 & 3 & \mathbf{4} \\ 4 & 4 & \mathbf{5} \end{bmatrix}\right) = \begin{bmatrix} \mathbf{1} & 1 & 1 \\ \mathbf{3} & 2 & 3 \\ \mathbf{4} & 4 & 0 \\ 2 & \mathbf{1} & 2 \\ 3 & \mathbf{2} & 3 \\ 3 & \mathbf{3} & 4 \\ 4 & \mathbf{4} & 5 \\ 3 & 2 & \mathbf{3} \\ 3 & 3 & \mathbf{4} \\ 0 & 4 & \mathbf{5} \end{bmatrix} \quad (34)$$

The reduction process takes place in the first line of the pseudo code from Code 3, and it is represented by the "simplify" function in the above equation. After the reduction, the new Correspondence Matrix is used to rebuild the connections seen in figure Figure 7.3. The new connections can be seen in Figure 7.4. These new and reduced connections will be used to generate the imaginary point cloud.

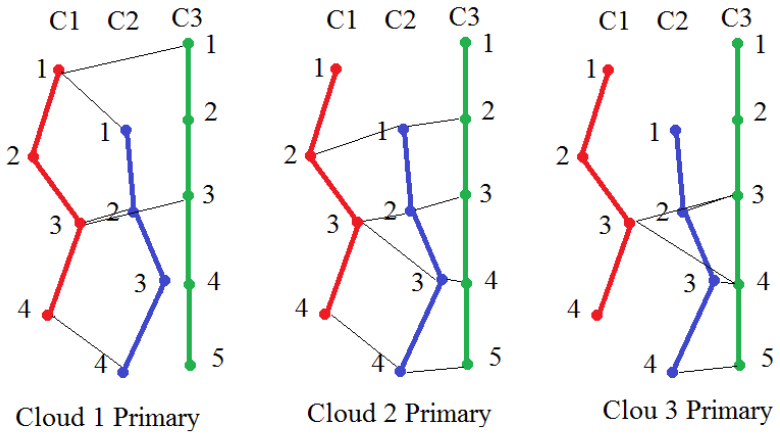


Figure 7.4 Reconstructed connections based on the new Correspondence Matrix

Having used the Correspondence Matrix to reconstruct the connections and showing them in Figure 7.4, it is possible to now possible to continue with the algorithm. Note that in the new correspondence Matrix, a value of "0" means that there is no connection. For example, line 3 of the correspondence matrix shows the values [4 4 0]. That means that point 4 of point cloud one connects to point 4 of point cloud two, and to no points in point cloud three.

The next step is to compute the centroids of all the lines from the Correspondence Matrix. This step forms the actual imaginary point cloud. The centroids are computed using Equation (35).

$$c_i = \frac{1}{n_i}\sum_{j=1}^{n_i} P_j^i \;;\; CM(i,j) \neq 0 \quad (35)$$

Where:

- i represents the index of the point in the Imaginary Point Cloud

- j represents the point cloud index
- n_i represents the number of points which are connected (number of non-zero elements in each line of the Correspondence Matrix)

The centroids which are computed based on the new Correspondence Matrix are now displayed in Figure 7.5. The centroids are represented by black dots in the roughly in their actual position. Note that the figure is not drawn to scale.

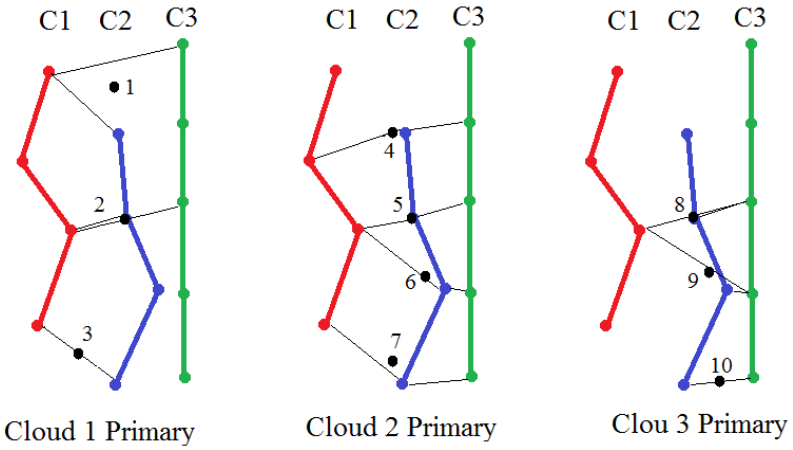


Figure 7.5 Imaginary Point Cloud generated for different point cloud combinations.

Figure 7.5 has a total of ten black points which make up the imaginary point cloud. It can be seen that the new Correspondence Matrix also contains ten lines. This is because each line corresponds to one point. The figure also has some interesting properties which should be explained before proceeding.

As can be seen in Figure 7.4 some of the points are connected in every one of the three cases. For example, the following combination of points appear with every point cloud as the primary point cloud: [3 2 3]. This can easily be seen in the new Correspondence Matrix, where the combination appears on lines 2, 5 and 8. As a consequence of this line appearing multiple times, the centroid will also be in the same spot, which is expected since the same points are used to generate the centroid. It can also be seen that in some cases there are only two connected points such as the connection between point 4 from point cloud one and point 4 of point cloud two. In this case, the centroid lies exactly on the middle of the connecting line.

Having gone through the basics of Imaginary Point Cloud generation, it is now essential to maintain the correspondences between the point clouds and the imaginary point cloud. Before, points could form connections only to points from other point clouds. This is seen very clearly in Figure 7.3. However, now that an Imaginary Point Cloud has been generated, the old connections will be lost to form new connections.

The new connections are easily derived by using a certain representation of the matrices and vectors which store the points of all the point clouds (including the Imaginary Point Cloud).

$$CM_{ICP} = (CM_{new}, I_{ICP}) = \left(\begin{bmatrix} \mathbf{1} & 1 & 1 \\ \mathbf{3} & 2 & 3 \\ \mathbf{4} & 4 & 0 \\ 2 & \mathbf{1} & 2 \\ 3 & \mathbf{2} & 3 \\ 3 & \mathbf{3} & 4 \\ 4 & \mathbf{4} & 5 \\ 3 & 2 & \mathbf{3} \\ 3 & 3 & \mathbf{4} \\ 0 & 4 & \mathbf{5} \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \end{bmatrix} \right) = \begin{bmatrix} \mathbf{1} & 1 \\ \mathbf{3} & 2 \\ \mathbf{4} & 3 \\ \mathbf{1} & 4 \\ \mathbf{2} & 5 \\ \mathbf{3} & 6 \\ \mathbf{4} & 7 \\ \mathbf{3} & 8 \\ \mathbf{4} & 9 \\ \mathbf{5} & 10 \end{bmatrix} \quad (36)$$

It is possible to create a vector which contains all the indices of the Imaginary Point Cloud which is denoted as I_{ICP} in the above representation. Evidently the number of lines matches the number of lines in the CM_{new} because I_{ICP} is generated using CM_{new} as a starting point. Now it is important to know which points from the original point clouds are the ones being connected to which points from the Imaginary Point Cloud. The answer is that the points represented in bald letters are the ones which form the connections therefore these are the points which will be connected to the points in the Imaginary Point Cloud. The new connections are given in CM_{ICP} and are shown in Equation (36).

The connections that result from CM_{ICP} can be seen in Figure 7.6. However, since the CM_{ICP} is a matrix which contains only two columns, this means that there are only two point clouds in theory which makes it impossible for every point cloud to move relative to another point cloud. To avoid this, the algorithm remembers where one point cloud begins and where it ends. Different colors have been used in CM_{ICP} to differentiate between the point clouds. This makes it easier for the user to understand the notation being used in this algorithm.

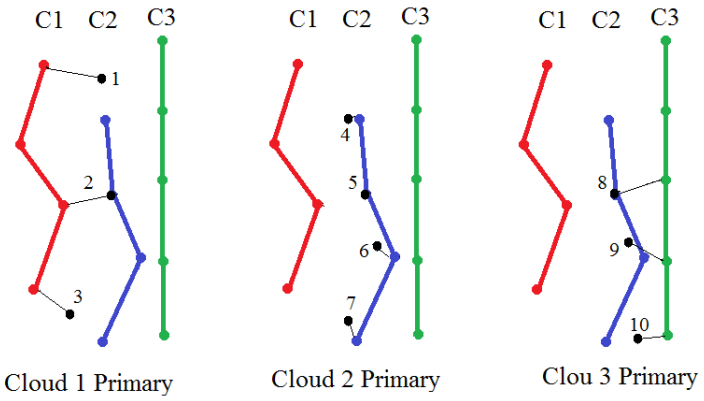


Figure 7.6 Original Point Clouds with connections to the Imaginary Point Cloud

At the end of this step, the previous connections which were available in the Correspondence Matrix have been completely remade. This was possible with the help of an Imaginary Point Cloud which is created using the centroids of the connections computed in Section 7. 3. The raw point clouds are now connected to the new Imaginary Point Cloud. It is very important to use this Imaginary Point Cloud further in the Registration part of the algorithm because this is the only point cloud which will not change its position. All the other point clouds will be moved with respect to the Imaginary Point Cloud.

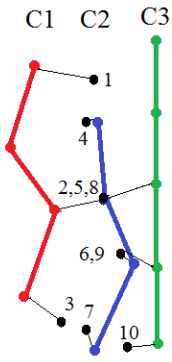


Figure 7.7 The complete Imaginary Point Cloud with all the connections present

Figure 7.7 shows the final connections in one single reference frame. It can easily be seen that points 2, 5 and 8 overlap, as well as points 6 and 9. Eliminating the points which overlap would consume extra computational resources. This is not needed to ensure the success of the algorithm, as the number of overlapping points in realistic data is insignificant when compared to the total number of points.

Note: The connections available in at the end of this step are very much dependent on the selection methods used for the "$simplify$" Equation (34).

7. 5. REGISTRATION

After obtaining the Imaginary Point Cloud, the registration component of the algorithm can be discussed. This part of the algorithm is very similar to the registration discussed in Chapter 5. However, there are some modifications which have been made in order for the registration to be applicable to the new alignment method discussed in the previous sections. These modifications will be explained in the following section.

The inputs to this phase are the raw point clouds, the Imaginary Point Cloud and the connections between the raw point clouds and the Imaginary Point Cloud. These inputs are then processed to align every point cloud and save the transformations between these point clouds.

The approach functions by taking in two point clouds at one time and trying to align the point clouds as best as possible based on the connections formed in the previous step. Of the two point clouds, one is a raw point cloud, while the second one is a part of the Imaginary Point Cloud. Note that the Imaginary Point Cloud is never used entirely as it is very large and only a small part of it is connected to any single raw point cloud.

This results in an algorithm which iterates over the raw point clouds and tries to move them as close to the imaginary point cloud as possible. However, after all the point clouds have been aligned once, the whole process will be iterated again, and after a few iterations, the number of point clouds will increase as more data is being added to the algorithm. The fact that the amount of data in the algorithm changes and that there are so many iterations results in a need for the algorithm to keep track of the positions of the point clouds with respect to all the other point clouds. If this relative cannot be kept track of, then the new point clouds could diverge instead of converge and therefore make the entire algorithm useless.

In order to account for this, the registration component of the algorithm always shifts point clouds back to a reference frame so that when a new iteration starts, the point clouds are all in the same reference frame and therefore compatible.

This entire process is shown in more detail in Figure 7.8 where the entire registration algorithm is represented in a block diagram. Note the difference between this block diagram and the one in Figure 5.7. Figure 5.7 is meant to deal with only two point clouds at one time and it does not take into account the reference frame at the end of the translation. (Of course, with just two point clouds this is not an issue) However, with a growing number of point clouds it is expected to have a more sophisticated registration.

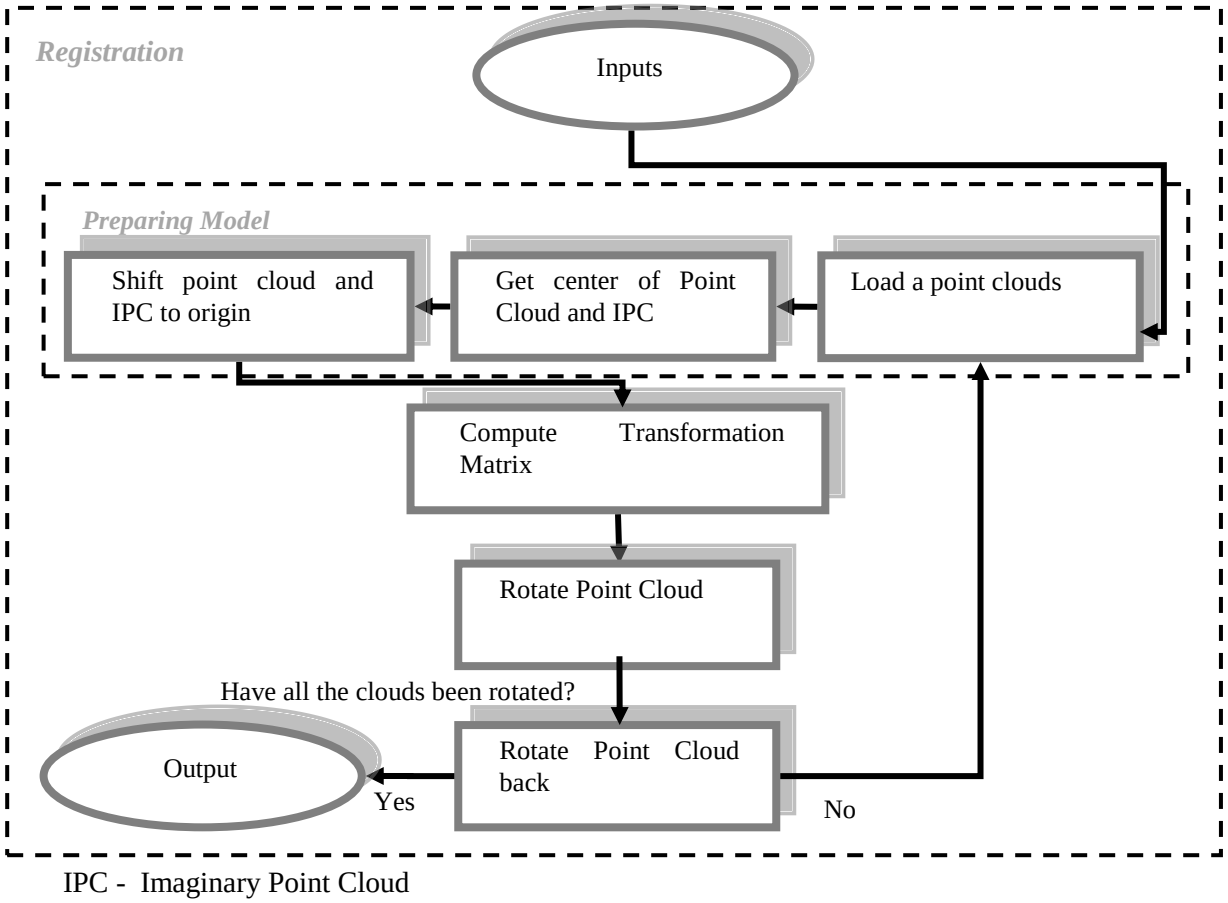


Figure 7.8 Simplified block diagram for the registration in the GICP

The first few step in Figure 7.8 after the inputs is to choose which points of the Imaginary Point Cloud need to be selected. As mentioned before, the connections are taken from CM_{ICP}, which is a very large matrix with two columns. This matrix saves all the connections from all the point clouds to the imaginary point cloud. However, the raw point clouds are loaded in one at a time. Therefore, the first step is to extract only the useful information from the CM_{ICP}.

When the point cloud which needs to be aligned is point cloud one, only the lines with red numbers in the first column are chosen. In the example given here colors are used to differentiate between point clouds, however in the algorithm, the exact number of points in each point cloud is remembered in a separate variable. This means that the algorithm knows directly how many

points to take and from which line. All of this takes place in the part of Figure 7.8 denoted: "Load a point clouds". By the end of this step, there are two point clouds in the registration (raw point cloud and partial Imaginary Point Cloud) module, and both point clouds have the same number of points.

The next two steps happen exactly as described in Chaper 5. The centers of both point clouds are computed and then used to position the point clouds in the origin of the coordinate system. This allows the alignment process to begin.

Afterwards, the two matrices which contain the x, y and z positions of the point clouds are multiplied to obtain a 3x3 matrix which is used to obtain the required rotations. The rotation is obtained with the help of an SVD function. More information on this function can be found in Appendix B: SVD.

Having computed the rotations, it is now possible to compute the translations by rotating the raw point clouds and computing the difference between the centers of mass which were obtained previously.

However, in this registration algorithm, all the point clouds move. This means that there is no initial reference frame preserved. This is a problem for the reconstruction algorithm because, as mentioned before, when a new point cloud is added, the accumulating errors of not saving the reference frame will cause the model to diverge. In order to prevent this, all the models will be transformed with the inverse transformation of the first point cloud.

Assume a vector T which contains the transformations for an m number point clouds which is shown in Equation (37).

$$T = [T_1 \quad T_2 \quad \dots \quad T_m] \tag{37}$$

To move all the point clouds back in the reference frame of the first point cloud, it is sufficient to multiply all the point clouds by T_1^{-1}. This results in Equation (38).

$$T = [T_1 T_1^{-1} \quad T_2 T_1^{-1} \quad \dots \quad T_m T_1^{-1}] = [I \quad T_2 T_1^{-1} \quad \dots \quad T_m T_1^{-1}] \tag{38}$$

This proves that the first point cloud remains in its own reference frame, as its transformation matrix is the identity matrix which means that it does not translate at all.

This way, all the relative transformations are preserved and all the point clouds are moved to the reference frame of the first point cloud. Having this consistency allows the algorithm to keep track of all the point clouds and to ensure that as new point clouds are being added the model still converges.

7. 6. MODULES

7. 6. 1. POINT DELETION

One of the first things that the algorithm does is it deletes a certain percentage of the points from the input point clouds. This is because searching kd Trees is still time consuming and it is much easier to search for fewer points than for the full point cloud.

There are two different point deletion methods being applied in this module. First, the total size of the point cloud is reduced. The reduction is done based on a certain percentage which depends on the size of the cloud. Then, the second reduction is a reduction of the points which are used to generate the kd tree.

It is also possible to leave the clouds intact in order to verify whether there is an actual improvement or not by reducing the points. But experiments have shown that reducing the points has a beneficial effect because it helps the algorithm by reducing the number of local minima. This in turn, reduces the probability of running in a local minima.

A diagram of the reduction process can be seen below in Figure 7.9.

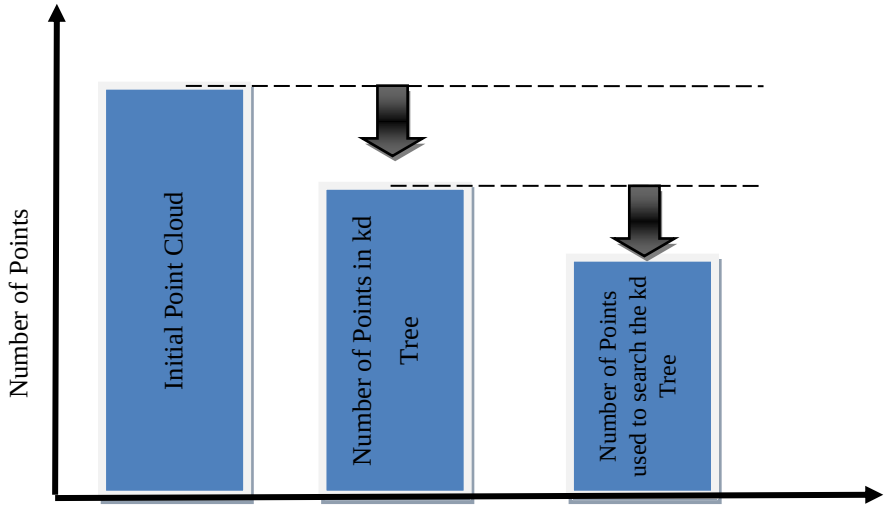


Figure 7.9 The diagram shows that the number of points used to search the kd Tree can be at most equal to the number of points in the kd tree, and the number of points in the kd tree can be at most equal to the number of points in the point cloud.

As can be seen in the figure above, the kd tree can be the same size as the initial cloud or smaller only. This is because the original number of points can only be reduced but not increased. The third column in Figure 7.9 can have a maximum size that is only as large as the column before it.

For example, it is possible to have the second column be 50% of the first column, but then the third column must be 50% or smaller than the first column. This is because if only a portion of the points are used to generate the kd tree, then the same number of points or fewer must be used to search the kd tree in order to obtain reasonable results.

For the alignment of several point clouds, a maximum of 500 points is sufficient for the alignment to be successful. If a model has 5000 points in each point cloud, and it is composed of 50 point clouds, then the complete model will have 250 000 points which would normally take a very long time to compute, but if 10% of the points are used, then the number of points which are computed is only 25 000. This does not mean that the model has fewer points. In the end, it is still possible to use all the points to display the model. The only difference is that fewer points are used to generate the model, which results in a lower processing time.

This module is being activated in each iteration of the GICP. This ensures an even distribution of the selected points. Since the points are reselected each time. The result is a generalization of point selections which are different each iteration which results in “different” point clouds that are aligned in each iteration. This whole process increases the probability of reaching a global minima while reducing the probability of a local minima, and thus improving the results as a whole.

7. 6. 2. KD TREE GENERATION

After points are deleted from the input point cloud, the points need to be organized in a kd Tree. The kd Tree generator is a third party algorithm which is used to generate an easily searchable format for points in space.

This module is accessed at the beginning of each iteration just after points have been selected, and it uses the second column of points which is represented in Figure 7.9.

Figure 7.10 shows a simple 2D, kd tree and how it is organized in 2D space. The kd tree is visible on the left side of the image, while the actual distribution in 2D space can be seen on the right side of the image. The root of the tree is the line denoted as "1", as can be seen on the left of the image. From this line, there are multiple branches extruding out to the sides.

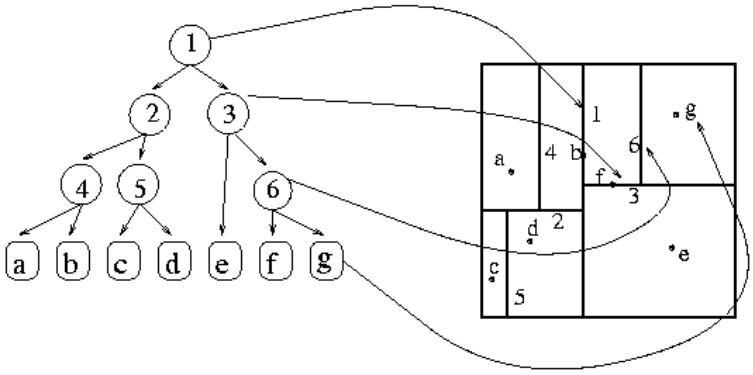


Figure 7.10 2D distribution of space for a 2d Tree (a kd tree in 2 dimensions). It can be seen that each branch of the tree corresponds to a different separating line in the space division. (Couto & Napoli, 1998)

A more complicated kd Tree can be seen in Figure 7.11. Although this tree is 2D, it resembles the trees used in the algorithm more closely in terms of complexity. There are hundreds of points which are all connected by branches to facilitate the searching algorithm. Although this method can also run slow if there are thousands of points, it is still much faster than calculating the distances between all the points every time.

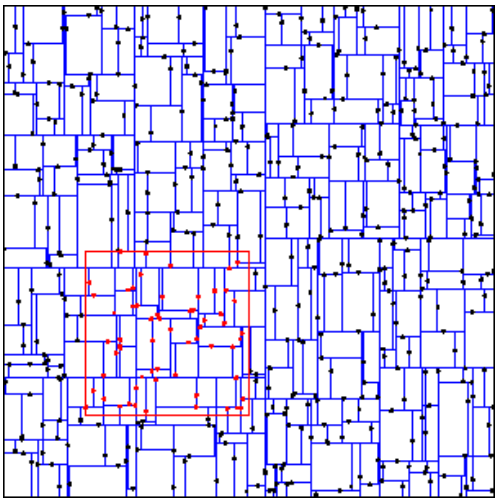


Figure 7.11 A larger version of a 2d Tree (sourceForge.net, 2012)

7. 6. 3. POINT REDUCTION

This module is activated just after the correspondence finding component of the code. The input of it is the correspondence matrix which is outputted by the correspondence finding algorithm. What this module does is it reduces the correspondence matrix by eliminating points which have no correspondences and duplicates.

This is done in order to reduce the size of the correspondence matrix and to make the algorithm run faster by reducing the number of points. In each iteration, there can be as many as several hundred points which need to be taken out.

The example below gives a better understanding of the function and how it works. The correspondence matrix is denoted as "CM", and for the purpose of this example, a "0" means that there is no connection. The format of the correspondence matrix has been explained in Chapter 7. 2. 4 and will not be explained again here.

Two points can have no connections if:

- The distance between the two points is too large.
- The distance between the two points is too small.
- Another criteria is used when selecting points and this criteria forbids the connection between the points.

In this case, the algorithm just ignores the connections between the points. All of these can improve the final result of the model if used correctly, however, if not used correctly, the point clouds could diverge instead of converge and give a bad model reconstruction.

The example contains three point clouds, the first two containing 4 points and the fifth being composed of 5 points. As described earlier in the chapter, the diagonal of the correspondence matrix is written in bold in order to make it easier to distinguish the clouds from each other. The correspondence matrix given here is the same as equation (33). However, it is assumed that some of the connections will not be formed. The connections which are not formed have a "0".

$$CM = \begin{bmatrix} \mathbf{1} & 1 & 1 \\ \mathbf{2} & 0 & 2 \\ \mathbf{3} & 0 & 0 \\ \mathbf{4} & 4 & 4 \\ 0 & \mathbf{1} & 0 \\ 3 & \mathbf{2} & 3 \\ 0 & \mathbf{3} & 0 \\ 4 & \mathbf{4} & 5 \\ 0 & 1 & \mathbf{1} \\ 1 & 0 & \mathbf{2} \\ 3 & 2 & \mathbf{3} \\ 0 & 0 & \mathbf{4} \\ 4 & 4 & \mathbf{5} \end{bmatrix} \quad (39)$$

The algorithm now looks at equation (39) and deletes all the lines which contain only one non-zero value. In this case, lines 3, 5, 7 and 12 are deleted from CM. The algorithm does this by counting the number of non-zero elements in each line of equation (39). Thes number of non-zero elements is then stored in a vector V which is shown in equation (40).

$$V = [3 \quad 2 \quad 1 \quad 3 \quad 1 \quad 3 \quad 1 \quad 3 \quad 2 \quad 2 \quad 3 \quad 1 \quad 3]^T \quad (40)$$

Vector V shows that there are 3 non-zero elements in the first line, 2 non-zero elements in the second line, one non-zero element on the third line, etc. Now vector V is used to decide which lines of the algorithm need to be deleted. Since the number "one" appears on positions 3, 5, 7 and 12 in vector V, these are the lines which will be deleted from CM.

After deleting lines 3, 5, 7 and 12 from CM, equation (41) is obtained. This equation is now saved as CM'.

$$CM' = \begin{bmatrix} \mathbf{1} & 1 & 1 \\ \mathbf{2} & 0 & 2 \\ \mathbf{4} & 4 & 4 \\ 3 & \mathbf{2} & 3 \\ 4 & \mathbf{4} & 5 \\ 0 & 1 & \mathbf{1} \\ 1 & 0 & \mathbf{2} \\ 3 & 2 & \mathbf{3} \\ 4 & 4 & \mathbf{5} \end{bmatrix} \quad (41)$$

This resulting correspondence matrix (CM') is then outputted as an input for the part of the algorithm which has the task of generating the imaginary point cloud.

7. 6. 4. UPDATING VALUES

This module is a simple one which updates values and prepares the algorithm for a new iteration step. It serves the crucial role of saving the progress that has been made so far and ensuring that none of the data has been lost.

It also ensures that the data which is saved is consistent with the progress of the algorithm and that the correct transformations and point clouds are saved.

The most crucial values that are saved are:

- The global transformation values
- The point clouds
- Errors
- Housekeeping values

The global transformation values which are saved are several values which are stored in a vector. Each element of the vector represents the transformation of one point cloud from its initial, raw, state to the current state.

When the algorithm finishes aligning the point clouds, the transformation vector will save the transformation from the raw cloud to the final aligned position for every point cloud.

The point clouds are also saved, in order to be used by the algorithm later for visualization of the results. Normally the point clouds are saved in their raw and unaltered format which means that they are saved before the algorithm manipulates them. Another version of the clouds is saved at the end. This version is the one showing the complete reconstructed model for easy visualization.

Errors are one of the most results of the algorithm. The errors are also computed in this module and are given as outputs at the end. It should be noted that the errors are only computed for virtually generated data. This is because they require Ground Truth (GT) data which is only available from the simulator. In

reality there is nowhere to obtain this data. More information on how the errors are computed is given in the Evalutaion chapter.

Housekeeping variables are required to store essential data that the algorithm needs to run.

Page intentionally left blank

8 RECONSTRUCTION

Reconstruction of the target satellite model and the trajectory of the orbiter around it are the main goals of this algorithm. Because of this, it can be concluded that the pieces of code that actually compute the rotations are the most important ones. However, most of the code does not compute the rotations required for model reconstruction. That is because most of the code consists of filters and automated procedures that choose points and features which the algorithm should consider when transforming different point clouds.

Because of all of this extra code, it is easy to lose track of the actual part of code which deals with the actual model reconstruction. This chapter will investigate the reconstruction on a much deeper level and explain in detail how the point clouds are aligned. To make it easier to understand, all the filters are ignored for the sake of simplicity and understanding.

First, a roadmap of the algorithm is given with emphasis on the model reconstruction parts, secondly, these parts will be described in more detail and their functions will be elaborated more. And lastly, the mathematical principles and formulas will be given to show how the results are reached.

8. 1. DIAGRAM

Because recreating a recognizable model from Figure 8.1 is quite a challenge even for a human, it is even harder for a computer which has no concept of what it has to do or what the model is supposed to look like. The reason why Figure 8.1 is such a challenge for a human being is because it is very hard to distinguish different point clouds and then to determine where each of them needs to be placed with respect to the next one. Basically, a human has no idea what the final product needs to look like, and neither does a computer. The reason this is much harder for a computer is because it does not have the notion of “object” or “geometry” the only thing that the algorithm does is it follows instructions without knowing why it is following them or what the final goal is. This proves especially difficult when it has to reconstruct an object that a human must be able to recognize, when even a human does not know what the object is supposed to look like when completed.

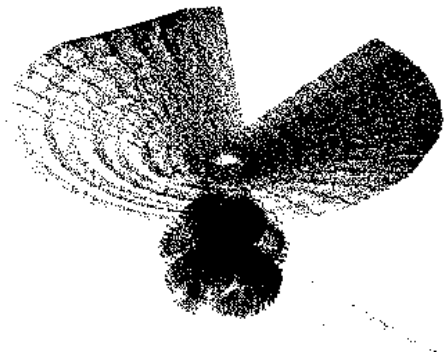


Figure 8.1 a number of raw scans of an object

Therefore, to simplify the task for the computer, there are multiple instances in the algorithm which align different parts of the model to different extents. This ensures that the work is divided amongst multiple agents which are used together to reconstruct the model. This makes the entire process much simpler to

understand for a human. It is also a more logical approach, since attempting to reconstruct the whole model in one step is highly unrealistic and improbable due to the complexity of the task at hand.

In the Global Iterative Closest Point algorithm there are many parts of code which compute the transformation matrices for the model reconstruction. These parts of code are located throughout the algorithm depending on their function. To make it easier to understand the order in which they are used a diagram is drawn that shows the exact relation of each part of code to the next one. The diagram showing these positions is given below in Figure 8.2.

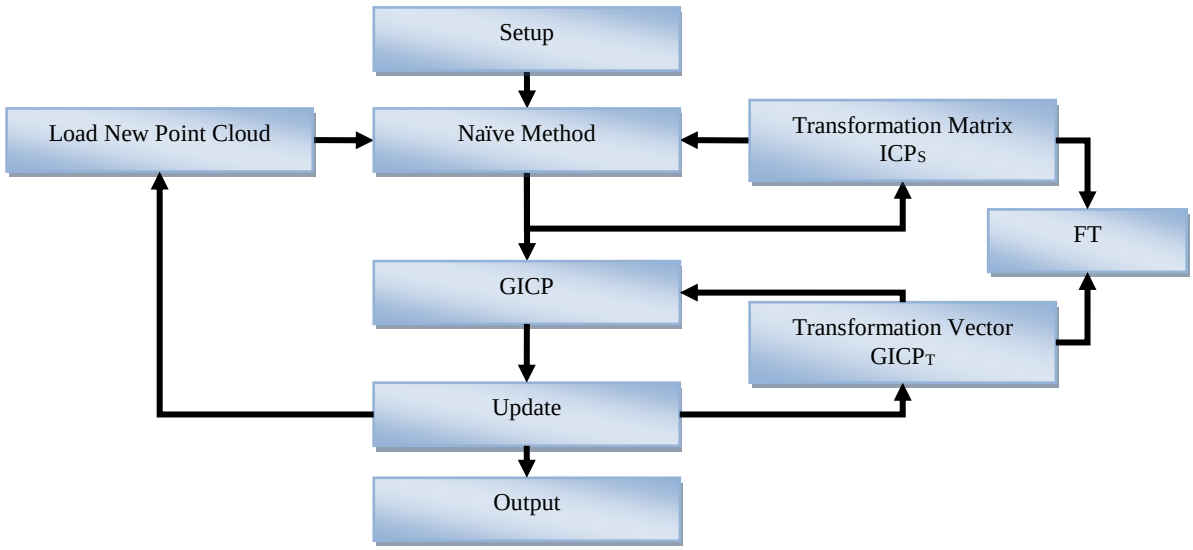


Figure 8.2 Information Flow Diagram

Figure 8.2 shows the steps of the algorithm which are directly involved in the model reconstruction process. An important note is that the diagram shown in Figure 8.2 is not a different algorithm from the diagram shown in Figure 7.1. In fact this is a diagram showing the same algorithm but concentrating on different aspects of it.

The "Setup" section is the exact same section in both figures and it has the same function. It generates the absolute same variables which then go into the algorithm.

The next section called "Input Clouds" is a subsection of the Input section from Figure 7.1. In the initial image, there are more parameters that are used as inputs including filtering parameters which are ignored in this chapter for the sake of simplicity. Therefore, only the clouds which are taken as inputs are considered for this chapter.

"Naïve Method" is the basic implementation of the iterative closest point algorithm. This code takes as input two clouds and tries to match them as well as possible. It is also the workhorse of the entire program. This is the first alignment that the point clouds have to go through before they are handed over to the GICP part of the algorithm.

Next, the Global Iterative Closest Point is activated to further align the model based on the clouds that come out of the ICP step and the clouds which are already in the memory of the algorithm. This step is more of an optimization step which is supposed to bring the model together and make it recognizable. It is a crucial step for reducing the total errors of the model.

"Update" is a step in which the results of the ICP and the GICP steps are brought together and combined as well as saving the progress so that nothing is lost in case of an unexpected crash. The results of the previous processes are combined in order to give a final result at the end of each iteration. This step sends the algorithm back to take another point cloud and then repeat all the steps until all the clouds have been taken and aligned.

The final step is the output which is the step that prints the final model and the transformations which bring the original raw scans into their final positions as a model.

8. 2. MODEL RECONSTRUCTION

This section describes how the algorithm approaches the alignment problem and how is solves different aspects of the problem at hand.

The emphasis is on how the tasks shown in the flow diagram from Figure 8.2 function to generate the transformation matrices and how the model is being reconstructed.

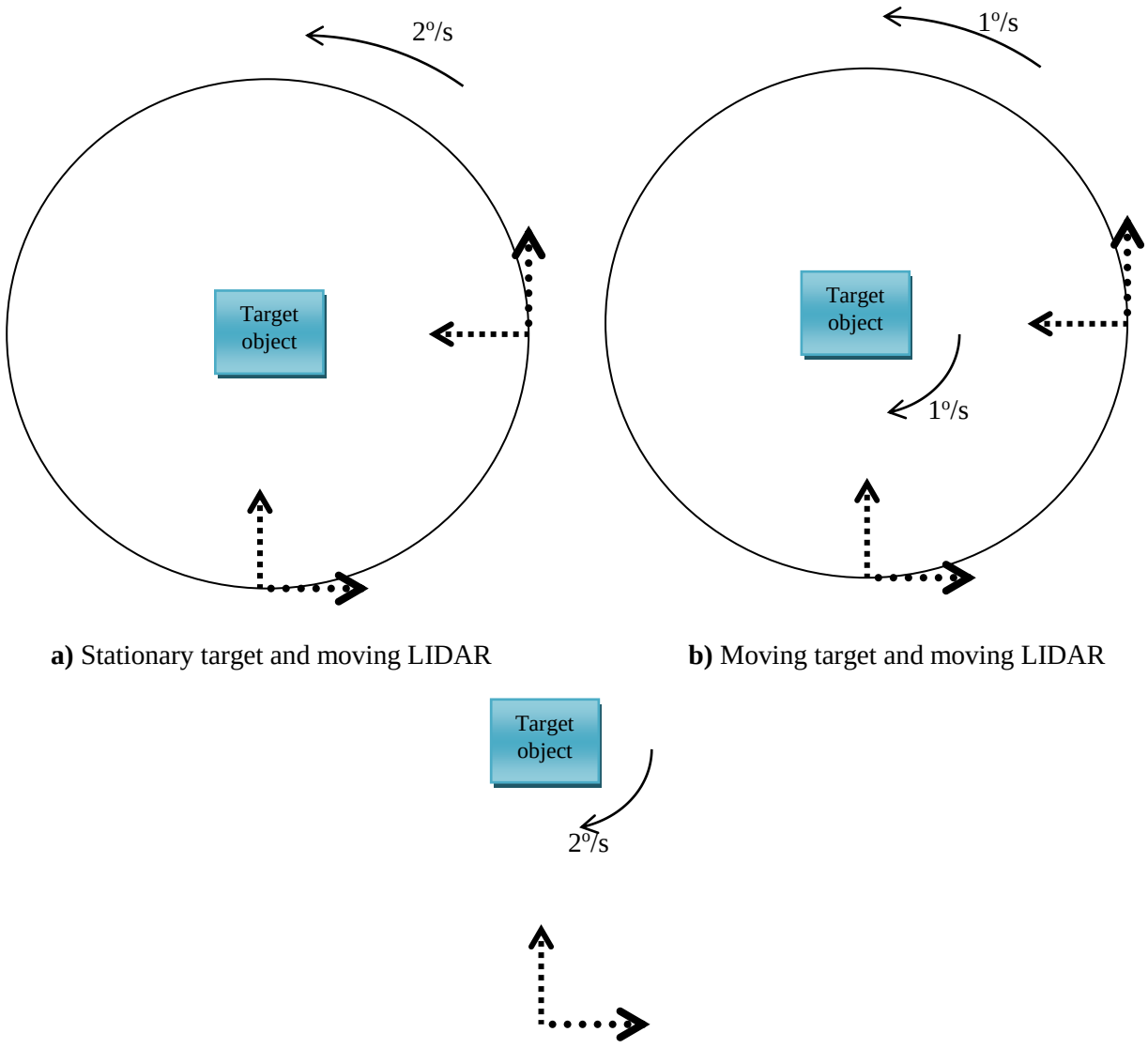
8. 2. 1. REFERENCE FRAMES

It is important to note that since the entire process of aligning points happens in outer space, there is no stationary reference frame. This makes alignment very difficult for the algorithm. Therefore, in order to make it possible to have alignment of the point clouds, a common reference frame needs to be defined. At the beginning, the reference frame is the LIDAR device which takes the measurements of the points. However, this reference frame is not ideal because the LIDAR is continuously moving and this results in a different reference frame every second of operation. A stationary reference frame needs to be defined to have consistency.

Therefore, to have a stationary reference frame, the algorithm defines the first scanned point cloud as the universal reference frame of the model. This reference frame is the one which is used when computing all the transformations and all the alignments as the algorithm runs.

Figure 8.3 shows these reference frames as the LIDAR turns around the object in a hypothetical ideal reference frame. It is immediately clear that as the LIDAR rotates around the object, so do the reference frames of the LIDAR. To make the problem even more complicated, the target object can have a rotation around its own reference frame.

To solve all these problems and to simplify the tasks, the first scan of the LIDAR is considered to be the initial reference frame. By defining this as the initial reference frame, all the previous problems can be simplified. Now the reference frame is attached to the object and it is independent of the rotation of the LIDAR sensor or of the target object.



a) Stationary target and moving LIDAR

b) Moving target and moving LIDAR

c) Moving object and stationary LIDAR

Figure 8.3 Different reference frames as in the problem at hand

The different reference frames can be seen in Figure 8.3 where the exact same situation is shown from three different perspectives. First, in Figure 8.3 a), the system shown is a target object and a LIDAR instrument rotating around the object with 2º/s. This reference frame is located in the body of the target and it is moving with the target. In the second image, Figure 8.3 b), there is a system as seen from an outside observer. The observer reference frame is the Earth reference frame. In this frame it becomes apparent that not only the LIDAR turns around the object, but that the object also rotates around its own axis with a rate of 1º/s, while the LIDAR rotates around it with 1º/s as well. Finally, Figure 8.3 c) shows the system from the perspective of the LIDAR. In this reference frame the LIDAR is immobile, while the object rotates around its axis at a rate of 2º/s.

None of the reference frames are wrong, and none are right, however, it is very important to understand these frames in order to solve the problem.

The inputs are generated from the reference frame of the LIDAR (Figure 8.3 c)). These inputs can be seen in Figure 8.1 where the point clouds appear to be rotating around a central axis. This is caused by the rotation of the model as the LIDAR is standing still (according to the LIDAR reference frame). However in order to show the model as clearly as possible it is best to use the reference frame from Figure 8.3 a). Furthermore, in order to reconstruct the trajectory and represent it in a way that is easy for humans to understand, it is necessary to use the reference frame from Figure 8.3 b).

8. 2. 2. APPROACH

The algorithm works by adding point clouds as they are being scanned to its memory and analyzing them as new clouds are being added. The point clouds are added one at a time as this is what would happen in reality as well. The sensor will be rotating around a target satellite and it will take images. As the images are taken, they are added to the algorithm.

SETUP

Once a point cloud has been added inside the algorithm's memory, the alignment process can begin. There need to be at least two point clouds for the alignment to take place, therefore when there is only one point cloud, there will be no alignment and the algorithm will not be activated.

INPUT CLOUDS

The point clouds are saved in a vector which contains all the point clouds and stores the positions of their points in 3D space. As new point clouds are being scanned, they will be added at the end of the vector making it larger and larger. These point clouds are the main input to the algorithm.

Once there are two point clouds in the memory of the algorithm, the alignment process can begin. The alignment process begins with a simple ICP algorithm. This algorithm takes as input the first two points and tries to align them. This is where the first transformation matrix is computed.

ICP

The first transformation matrix transforms point cloud B to the reference frame of point cloud A, while not moving point cloud A at all. After point cloud B is transformed, it is transferred further through the algorithm to be further aligned. The transformation matrix computed in this part is remembered for future use.

For example, in Figure 8.4 two point clouds can be seen with an initial error of 15^{o}. This error is decreased by applying the ICP algorithm to the two point clouds and moving point cloud B toward point cloud A. It can be seen that the correction of the ICP is 5^{o}. The variable ICP_S from the figure is the same variable that appears above in Figure 8.2. Note: This example is used only to explain a phenomenon, the numbers are not actual results.

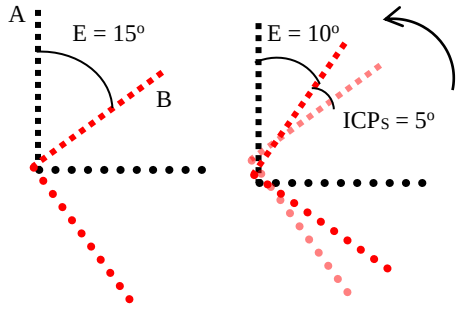


Figure 8.4 Alignment of two point clouds using only the ICP algorithm

Where:

- ICP_S is the transformation matrix obtained from the ICP algorithm. (can also be noted as ICP_{BA}, transformation from B to A)
- E is the error of the alignment.

If there are more than just two point clouds in the algorithm, the previously computed transformation is used for updating new results. This helps when aligning subsequent point clouds because the ICP can only align two point clouds with each other, this means that in the case of three (or more) clouds, point cloud 1 cannot be in the ICP alignment computation. This also means that the reference frame will be the reference frame of one of the other point clouds in the computation instead of the correct reference frame (frame of cloud 1) This can be seen in Figure 8.5.

However, the transformation of point cloud A to the original reference frame is known from the previous iteration, therefore by multiplying the second transformation from point cloud B to point cloud A by the transformation from point cloud A to point cloud 1 it is possible to obtain the transformation matrix for point cloud B to point cloud 1. This process is then extrapolated to all other steps in the exact same manner in order to bring all the point clouds in the same reference frame.

A and B are used for the last two point clouds because A and B can be any two consecutive point clouds. However, point cloud 1 is fixed and the rest of the point clouds will always be aligned to point cloud 1. In Figure 8.5 it can be seen that there is still a small error left at the end. This error will not be discussed here, but it is important to notice how it accumulates because of the errors generated in the ICP alignment process. The magnitude of the error is exaggerated for clarity.

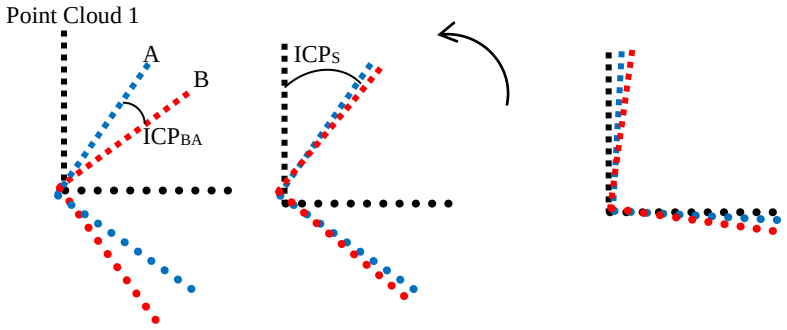


Figure 8.5 Alignment of three point clouds

Once the transformation from point cloud B to A has been computed, the resulting transformation matrix will be multiplied by the transformation matrix that translates A to the reference frame of the first point. This is done using Equation (42).

$$ICP_S \leftarrow ICP_{B1} = ICP_{A1} \cdot ICP_{BA} = ICP_S ICP_{BA} \quad (42)$$

Where ICP_S is the transformation from A to the frame of reference of point cloud 1, and ICP_{BA} is the transformation matrix which transforms point cloud B onto point cloud A.

The result of the multiplication is saved in variable ICP_S again in order for this to be used in the next iteration, and thus keep track of the algorithm progress as new point clouds are being added.

A more general form of the above Equation (42) would be Equation (43).

$$ICP_{n1} = ICP_{(n-1)1} ICP_{n(n-1)} \quad (43)$$

The transformation process now becomes apparent as the process can be scaled to infinitely many point clouds to bring them in the reference frame of the first point cloud. However, in the picture it is also seen that the errors also stack up. This accumulation of errors can be so large that the result of the algorithm can be uninterpretable after the addition of several point clouds.

In reality, the problem is slightly more complicated than described above. That is because there is more than just the ICP at work in the entire algorithm. There is also a Global Iterative Closest Point alignment section that works in between the addition of point clouds. This means that by the third cloud, the ICP will have run once and the GICP will have also run once, thus further improving the results of the target model.

The influence of the GICP has not been described above, but in practice, this appears as an extra step in the ICP alignment, simply an extra alignment which needs to be considered when multiplying the alignment from the new point cloud by the alignment from the previous iteration.

GICP

Having computed the ICP just above, now it is time for the results to be passed to the Global Iterative Closest Point for further processing. The way in which the GICP functions is quite different from the normal ICP because, all the point clouds are transformed in the same iteration.

As described in the “Global ICP” (Chapter 7) the GICP transforms one point cloud at a time while keeping the rest of the point clouds stationary. This means that is starts off by transforming the first point cloud with respect to every other point cloud, then it transforms the second point cloud with respect to every other point cloud and so on.

The transformation values are saved in the GICP Transformations variable which is described above in Figure 8.2. Each element of $GICP_T$ (GICP Transformations) holds one transformation matrix which corresponds to a certain point cloud.

The contribution of the GICP to the error of the algorithm is to further improve the results obtained from the ICP. Continuing on the example given above in Figure 8.4, the results shown in the figure are sent to the GICP after having been improved by the ICP algorithm. Figure 8.6 shows the contribution of the GICP as well as the contribution of the Naïve Method (ICP) in the same figure. It can be easily seen that the starting error is 15º, which, as mentioned above is reduced by 5º due to the ICP algorithm. This means that the GICP starts with an error of 10º and it corrects it by a further 5º. It should be noted that the ICP causes the first point cloud to move as well, as seen in the figure below. By the end of the GICP the total

error of the alignment is only 5º. As in the previous example, the numbers given here are purely for demonstration purposes and not actual results.

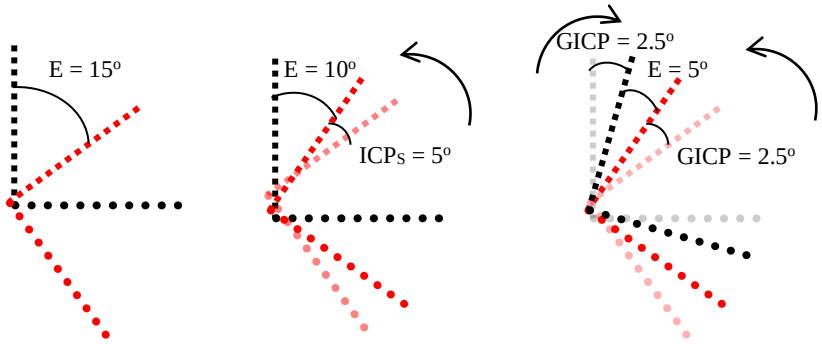


Figure 8.6 GICP contribution to the alignment of two point clouds

The GICP functions by not treating any of the point clouds as stationary point clouds, which means that all point clouds move with respect to every other point cloud. This causes other issues such as the fact that by the end of the GICP none of the point clouds are aligned in the first reference frame. Each of them having diverged from the original position. This can be seen in Figure 8.6 where it is clear that both point clouds move by 2.5º instead of having just one point cloud moving 5º. (as is the case for the ICP algorithm)

This causes a ripple effect which in turn causes the ICP to have a larger error and that brings it back in the GICP which increases the error even more. Therefore, the point clouds need to be brought back in the reference frame of the original point cloud before they are allowed to pass further through the algorithm.

By transforming the point clouds back to the reference frame, there will be no misalignment of the first point cloud in the next iteration of the ICP code block. This means that the alignment will again be in the first reference frame which can be used by the GICP to further improve the results and pass them on again to the next iteration.

In order to bring back the clouds to the reference frame it is important to know in what frame the clouds are in after the GICP. Before the GICP begins, all clouds are in the reference frame of the first point cloud because they have been brought to this reference frame by the ICP algorithm. Therefore, at the end of the GICP algorithm, the clouds are all still in the first reference frame but with a different transformation applied to each of the clouds. In order to fix this, the reference frame needs to change back to what it was before (the first point cloud needs to be back in the origin). In order to achieve this while maintaining the relative pose of all the point clouds is by bringing them back to the original point cloud by using the inverse transformation of the first point cloud. What this means is that all the clouds retain their relative positions, but at the end, the first point cloud will be back in the origin of the reference frame. This process can be seen as a “scaling” of the positions and rotations. The relative “size” or position remains the same, but the coordinates are different for each component.

Figure 8.7 shows how the clouds are first realigned using the GICP algorithm and then finally brought back to the first reference frame.

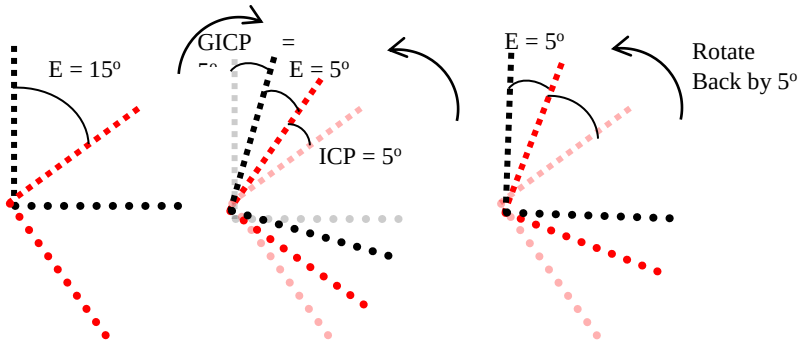


Figure 8.7 Two point clouds returning to the original frame of reference after the GICP algorithm

Assuming that $GICP_T$ is a vector of n elements (model has n point clouds), it is possible to write it as Equation (44).

$$GICP_T = (T_1 \quad T_2 \quad \dots \quad T_n) \tag{44}$$

Where: T_1, T_2 and T_n are transformation matrices corresponding to point clouds 1, 2 until n respectively.

Now the point clouds need to be brought back in the reference frame of the first point cloud, and this is done by multiplying everything with the inverse of T_1 as shown in Equation (45).

$$GICP_T = (T_1 \quad T_2 \quad \dots \quad T_n)T_1^{-1} = (T_1T_1^{-1} \quad T_2T_1^{-1} \quad \dots \quad T_nT_1^{-1}) = (I \quad T_2T_1^{-1} \quad \dots \quad T_nT_1^{-1}) \tag{45}$$

Where: I is the identity matrix. It now becomes clear that after the GICP, the first point cloud did not move at all, since its transformation matrix is just the identity matrix.

UPADATE

After both the ICP and the GICP have been calculated for all the point clouds that are currently available in the model, the information needs to be combined. If the information is not combined, then there is no interaction between the two parts. This part of the code has the goal of combining the information from both transformations.

There needs to be no information lost in this process or the new values of GICP and ICP will have errors which will propagate throughout the model destroying the alignment and the trajectory reconstruction.

One difficulty in this step is that the two methods affect different parts of the algorithm. Table 3 compares several important aspects of the two approaches and describes the differences between them.

Table 3 A basic comparison between the ICP and GICP methods

	ICP	GICP
Input clouds	2	n
Computation time	Fixed	Scales with nr of PC
Advantage	When differences between clouds are large	When cloud differences are small
Transforms	1 cloud	All clouds
Input FOR	2	1
Disadvantage	Cannot close loops	Takes long to compute even for few clouds
Complexity	Low	High

From Table 3 it is clearly visible that while the normal ICP algorithm transforms one point cloud, the GICP transforms all the point clouds. This results in multiple transformation matrices coming out of one algorithm. The number of transformation matrices is actually equal to the number of point clouds that go into the algorithm.

The ICP however, only has one output, and "update"(see Figure 8.2) needs to combine the single output of the ICP with the multiple outputs of the GICP. This can be done because it is known that the ICP only affects the final point cloud while the GICP affects all point clouds. The solution therefore involves multiplying all the outputs of the GICP by their respective clouds and then multiplying only the final cloud by the ICP in order to have a proper combination between the two methods. This prevents any of the information generated by the two processes from being lost or overwritten.

As described above, the value which stores all the progress is called Final Transformations (FT).

FT is a vector which has the size equal to the number of point clouds in the model. Assuming a model with n point clouds, FT would look like Equation (46).

$$FT = (FT_1 \quad FT_2 \quad \dots \quad FT_i \quad \dots \quad FT_n) \tag{46}$$

Where: FT_1, FT_2, FT_i and FT_n are transformation matrices corresponding to point clouds 1, 2 and n respectively.

It should be noted that this vector grows as the algorithm progresses. It starts with a size of one (when there is just one point cloud) and increases with the addition of every point cloud.

The values stored in FT can bring the point clouds from their positions in the raw state to the positions in the final state in one multiplication. That is why having this variable is so convenient, because at the end of the algorithm this is the only variable which is needed to completely transform the raw point clouds to their final positions.

Each time that FT increases in size, the new value will start off with the newest ICP matrix, and as new point clouds are being added, the GICP contributions are also saved.

The formula for FT_i is given below in equation (47), where n is the number of point clouds, and ICP_{i1} is described above in page 77:

$$FT_i = ICP_{i1} \prod_{j=i}^{n} T_i^j T_1^{j^{-1}} \tag{47}$$

Given a model with three point clouds, then FT is given in equation (48).

$$FT = (FT_1 \quad FT_2 \quad FT_3) \tag{48}$$

And the elements shown in equation (48) are computed in equation (49).

$$\begin{cases} FT_1 = ICP_{11}\prod_{j=1}^{3} T_i^j T_1^{j^{-1}} = ICP_{11}T_1^1 T_1^{1^{-1}} T_1^2 T_1^{2^{-1}} T_1^3 T_1^{3^{-1}} = I \\ FT_2 = ICP_{21}\prod_{j=2}^{3} T_i^j T_1^{j^{-1}} = ICP_{21}T_2^2 T_1^{2^{-1}} T_2^3 T_1^{3^{-1}} \\ FT_3 = ICP_{31}\prod_{j=3}^{3} T_i^j T_1^{j^{-1}} = ICP_{31}T_3^3 T_1^{3^{-1}} \end{cases} \quad (49)$$

After checking the equations obtained above, it immediately becomes clear that the first point cloud is in the same place because all the multiplications give the identity matrix. For the second point cloud, it becomes apparent that the final transformation matrix is composed of the result from the Naïve Method which aligns the point clouds to the first reference frame, the GICP result from when there are only two point clouds, and finally the GICP result of when the final point cloud is added.

OUTPUT

The final step is also the simplest. After all the calculations have been done and the transformations have been computed correctly, the algorithm writes down all the transformations in a text file.

8. 3. TRAJECTORY RECONSTRUCTION

Reconstructing the trajectory is one of the main objectives of this algorithm. However, most of the discussion is about reconstructing the model. In order to transform the trajectory it is necessary to first have a reconstructed model. As mentioned before, the trajectory is reconstructed based on the model. The model reconstruction is used merely as a tool to determine the trajectory of the observer around the target satellite.

It has been mentioned before, that in order to reconstruct the trajectory, it is necessary to use the reference frame shown in Figure 8.3 b). It has also been mentioned in Chapter 3 that the position of the LIDAR will be used to reconstruct the trajectory. In each frame, the position (0,0,0) is the position of the LIDAR, and therefore the position of the satellite.

After the object has been reconstructed, the whole reference frame is transformed from Figure 8.3 c) to Figure 8.3 a). This is done by multiplying all the raw scans by the values stored in the vector FT. (The main output of the algorithm) Following this principle, it can be deduced that the origin of all the other point clouds also shifts around the target object. The shifted origins represent the position of the observer satellite over time. By connecting the points it is possible to reconstruct the trajectory of the observer satellite around the target satellite.

This means that equation (50) gives the reconstructed trajectory in the reference frame depicted in Figure 8.3 a):

$$Trajectory = (0 \quad 0 \quad 0) \cdot FT \quad (50)$$

In order to transform this trajectory to the reference frame shown in Figure 8.3 b), it is necessary to know the movement of the target object with respect to the observer. To transform the trajectory from the equation above to the reference frame of Figure 8.3 b) equation (51) needs to be applied:

$$Trajectory = (0 \quad 0 \quad 0) \cdot FT \cdot T_T \quad (51)$$

Where:

- T_T represents the transformation matrix of the target satellite.

9 EVALUATION

In this chapter, the calibration and validation of the algorithm is being discussed.

First, the ICP and GICP components of the algorithm will be verified. Statistics and important notes about their performance will be elaborated.

Next, the reconstruction of different models will be discussed and the performance while reconstructing the models will be analyzed. This is where the validation of the algorithm takes place.

Finally, a comparison between the different results will be made.

9. 1. EVALUATION PROCEDURE

The procedure that is followed in this chapter will be explained so that the reader has a better idea of how the results shown were obtained.

In this chapter, there are many results which were obtained from the algorithm. The results are then compared in order to determine how different parameters/target objects influence the final reconstruction.

In order to do these evaluations, three different aspects have been tested. Each of these aspects is discussed in a different section of this chapter. The way the experiments are set up is as follows:

- Data is obtained from the LIDAR simulator
- Data is ran through the algorithm with different settings
- The results are compared

Each of the three aspects which are mentioned earlier are different parts of the algorithm. It is known by now that the algorithm consists of a "Naïve Method" and a Global Iterative Closest Point component. Each of these will be tested below in different sections, finally the complete model reconstructions will be shown.

The three aspects being tested are:

- Naïve Method (ICP) performance alone
- Performance of the GICP (With Naïve Method as this cannot work alone)
- Model reconstruction

The errors are computed by comparing the Ground Truth values which were described in Chapter 3 to the FT values described in Chapter 8.

For each aspect, different parameters will be tested. It is known that the algorithm has many different input parameters which influence the results of the alignment. These different parameters will be tested so that the exact magnitude of the influence of each is known. The results are displayed in tables and graphs which make comparisons much easier.

First, in Chapter 9. 3, the Naïve Method is tested, and the errors are computed. This method is tested by varying the parameters which influence the ICP portion of the algorithm. And, as mentioned before, the errors are computed by comparing the output from the algorithm to the Ground Truth.

Secondly, in Chapter 9. 4, the Global Iterative Closest Point algorithm is tested. It is tested by varying the parameters which influence only this part of the algorithm, and the errors are computed using the same method as above.

Finally, in Chapter 9. 5, the resulting models are displayed and compared. These models are created using various configurations of the algorithm.

9. 2. ERROR COMPUTATION

In order to be able to properly determine the accuracy of the reconstruction of the model, there needs to be a way to calculate the resulting error of the algorithm. Since there is no actual way to obtain realistic data, unless the satellite is sent to space to gather real data, another method is used.

It has been mentioned in previous chapters that a LIDAR simulator is used to obtain data. This is a very good decision because, not only can the parameters of the simulated data be changed very precisely, but the simulator also gives the exact result as output. This output is a vector which contains transformation matrices for each point cloud that the algorithm simulates. If the point clouds are multiplied by these transformation matrices, a perfect model is obtained. The outputted transformation matrices are known as the Ground Truth (GT) of the model.

In the ideal case, the result of the algorithm is the same as the ground truth and the error is zero, however, this can never be the case in reality due to inaccuracies. Since the ground truth is known before hand, it is possible to compare the output of the algorithm with the ground truth and compute the exact error of the algorithm.

There are six errors which are important to determine the accuracy of the algorithm. These errors can be divided into two groups:

- Translational Errors: Errors in the X, Y and Z planes
- Rotational Errors: Errors in the Roll, Pitch and Yaw of the satellite

These errors will be shown after each experiment session for comparison with other experiment sessions.

It should be noted that the following errors are all trajectory errors and not model errors. Which means that the values given represent the errors in the trajectory reconstruction capabilities of the algorithm and not 3D model reconstruction. These errors have been chosen to represent the performance of the algorithm because the main goal of the algorithm is to reconstruct the trajectory, and reconstructing the model is merely a tool used to achieve this goal. It therefore makes more sense to use the trajectory errors to reconstruct the model.

9. 2. 1. PARAMETER SELECTION

In order to have a good comparison between the influencing parameters, the parameters will be varied one at a time, while keeping the rest constant. This is done by selecting a value for each parameter, and this value will be called the “base value” or the “standard value”. After the standard values have been selected for all parameters, one parameter is selected and varied, while the rest of the parameters are kept constant.

This allows the influence of the parameters on the algorithm to be compared against each other in order to determine which parameter is most crucial to the success of the algorithm. The results of varying various

parameters will be displayed in tables and in plots which show the evolution over multiple runs of the algorithm.

In order to have consistent results, only one target satellite model is used for the following experiments.

9. 2. 2. READING THE PLOTS

In order to read the plots it is necessary to understand the different outputs that the algorithm gives. When running some input data through the algorithm, the output is a vector of transformation matrices FT. This vector is described in more detail in Chapter 8. The vector stores values for the rotation and translation required to bring the raw point clouds from the initial position to the final "aligned" position.

The resulting errors from one run of the algorithm are plotted in Figure 9.1 and Figure 9.2. These two figures will be used to explain the results in all the other cases.

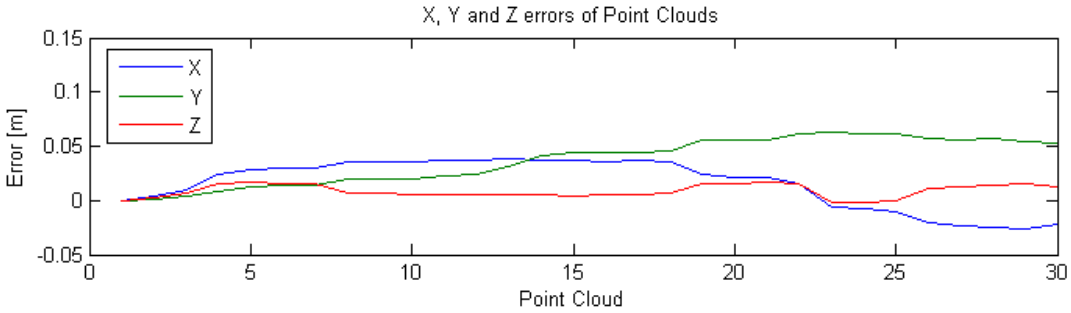


Figure 9.1 Translation error for one result of the algorithm

It can be seen that in Figure 9.1, the x-axis represents the point cloud number. Since the maximum number is 30, it means that there are thirty point clouds in this model. The values on the y-axis represent the errors in meters. The errors are trajectory errors which are computed by the algorithm. Because each scan represents a different position of the satellite, there are thirty errors in total, because the target satellite was in 30 positions while taking all 30 of these measurements.

These two images show just the result of one experiment. In order to compare this result with other results, the values are averaged.

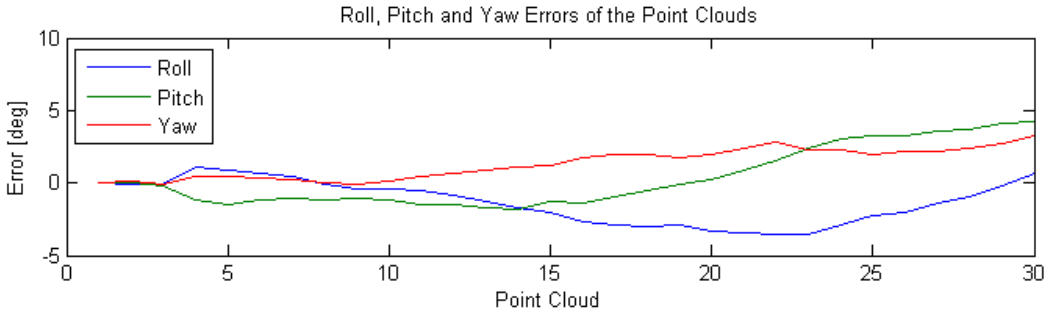


Figure 9.2 Rotational error of one of the results from the algorithm

These results belong to just one set of parameters. However, in order to combine the results, there need to be multiple sets of parameters, therefore, in order to make these comparisons easier to understand, the entire figures above will be compressed to just one point. This is done by averaging all the values of the

errors first across the x-axis. This reduces the x-axis to just one value, and the entire graph is reduced to a single line with three dots. Each dot represents the average errors in x, y and z directions for the translation figure, and the roll, pitch and yaw for the rotation figure. To further simplify the graphs, the remaining three points are also averaged out. In the end, the result is one point which gives an average error of all directions and all the point clouds. This error can then be plotted on a graph and compared with other errors, by varying certain parameters in the inputs of the algorithm.

9. 3. ICP

The ICP component has several parameters which can be chosen to affect the way in which it produces results. These parameters have to do with the points available in the point clouds and how the ICP treats them to make the clouds converge. The modifiable parameters available in the ICP portion of the algorithm are:

- Number of Iterations
- Number of Points
- High Tolerance

The number of iterations represents the number of times that the algorithm runs through the alignment process before it stops. The second property represents the number of points that are used in the alignment of the point clouds.

Finally, the last parameter is closely related to the weighing function of the algorithm, and can have different meanings depending on the weighing function.

9. 3. 1. PARAMETER TESTING

In order to test this algorithm, each of the parameters was tested independently in order to determine their influence on the model error. Note: the GICP algorithm was not activated for these tests. The results represent only the output of the ICP component.

In order to have clear tests showing the influence of the parameters three series of tests have been made. Each series of tests is meant to determine the influence of one of the parameters on the model error.

In order to have comparable results, a “standard” configuration has been determined in order to have a starting point for the test sessions. The standard configuration of the ICP algorithm has been chosen to be:

Table 4 Standard configuration of the ICP component

Parameter	Number	Units
Iterations	15	Integer
Nr. Of Points	0.025	%
Tolerance	0.3	m

When testing the ICP component, two of the variables from Table 4 will be kept constant while the third is changed to obtain different results. The results are then plotted in a graph in order to make it easier to see trends in the data.

SESSION 1

In the first session the number of iterations will be varied. The numbers tested are shown in the table below:

Table 5 Table representing the numbers of iterations used to obtain the results

Test Nr.	**1**	**2**	**3**	**4**	**5**	**6**
Value	5	**15**	25	50	75	100

In Table 5, the column marked in bald text represents the standard configuration of the algorithm, which is taken from Table 4.

Figure 9.3 and Figure 9.4 show the X, Y, Z and Roll, Pitch Yaw errors of the trajectory respectively. The errors are averaged out so that multiple experiments can fit in one graph. This means that the value that is shown in the position graph is the mean error of all three directions x,y and z. The same is done for the errors in the rotation graph.

From the first figure, it is not clear whether the increased number of iterations has a beneficial effect or a bad effect on the ICP algorithm. This is because the values hover between 0.12 and 0.06 meters of errors.

The second Image however clearly shows and improvement in the error as more iterations are being used. The error drops from 16° to approximately 8°. It can therefore be said that an increase in the number of iterations has a beneficial effect on the ICP component.

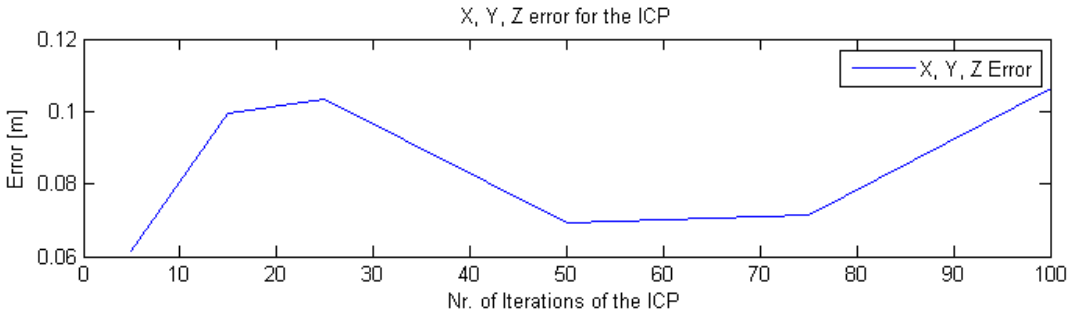


Figure 9.3 Average results of varying iterations for the translation component

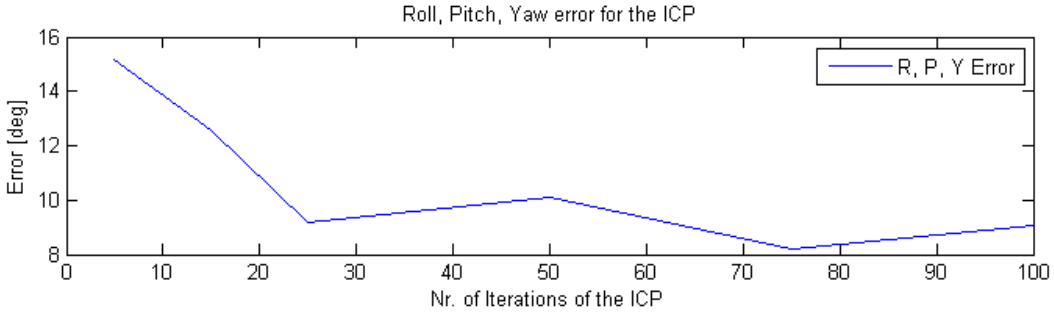


Figure 9.4 Average results of varying iterations for the rotation component

SESSION 2

In the second session, the varied variable is the number of points used to search in the kd tree. The tested values are given in Table 6

Table 6 Table representing the percentage values of points used to obtain the results

Test Nr.	1	2	3	4	5	6	7	8	9
Value[%]	1	**2.5**	5	10	20	40	50	70	100

As stated above, the bald column represents the standard configuration of the algorithm, which is taken from Table 4.

Unlike the results from the previous section, the results obtained for varying the number of points to use during the ICP does not seem to have a big effect. Figure 9.5 and Figure 9.6 show that there is no significant improvement in the error for using more than 5% of the points. This is because, as stated in previous chapters, using more points results in a higher probability of hitting a local minima. The second advantage of using fewer points is that the algorithm performs faster as it needs to do fewer computations.

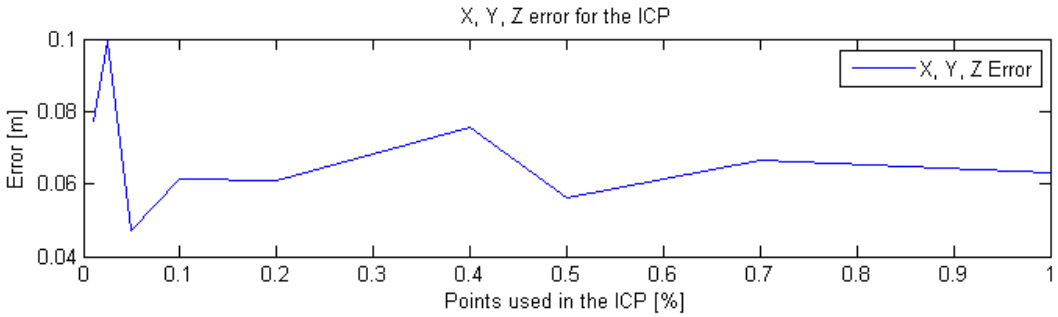


Figure 9.5 Average results of varying the nr of points used in the ICP algorithm in the x, y and z direction

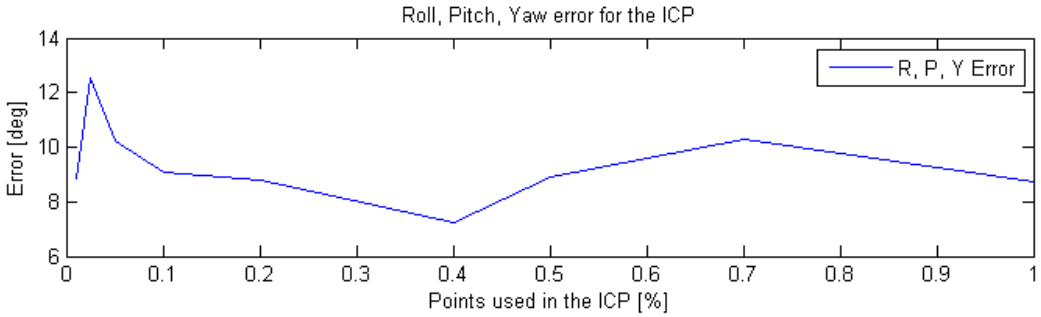


Figure 9.6 Average results of varying the nr of points used in the ICP algorithm in the roll, pitch and yaw direction

SESSION 3

Lastly, the tolerance was changed in order to determine its influence on the algorithm The values used are given in Table 7

Table 7 Table that represents the different tolerance values which were used in the algorithm

Test Nr.	1	2	3	4	5	6	7
Value[m]	0.1	0.2	**0.3**	0.4	0.6	0.8	1

The bald column represents the standard configuration as in the previous tables.

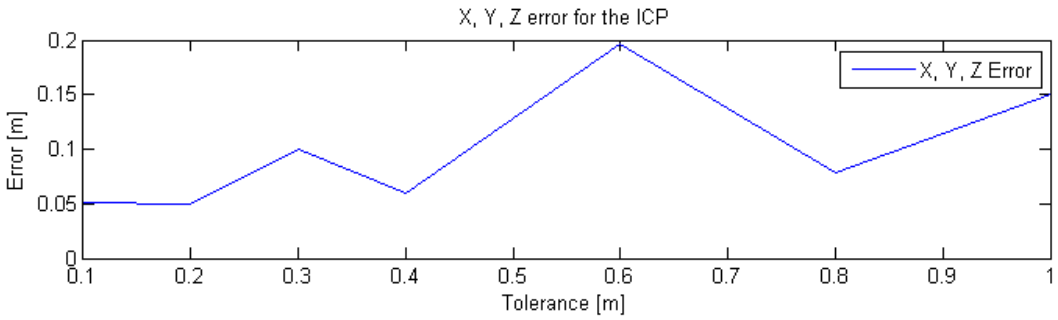


Figure 9.7 Average results of varying tolerance in the translation

Figure 9.7 and Figure 9.8 both show a clear increase in the error as the tolerance increases. This is because points with wrong matches have a higher chance of forming at larger distances which causes the model to be misaligned.

However, a tolerance which is too low can prevent enough connections to form, and can also prevent new connections to form once old ones have been established. This can cause the model to "freeze" before it is aligned.

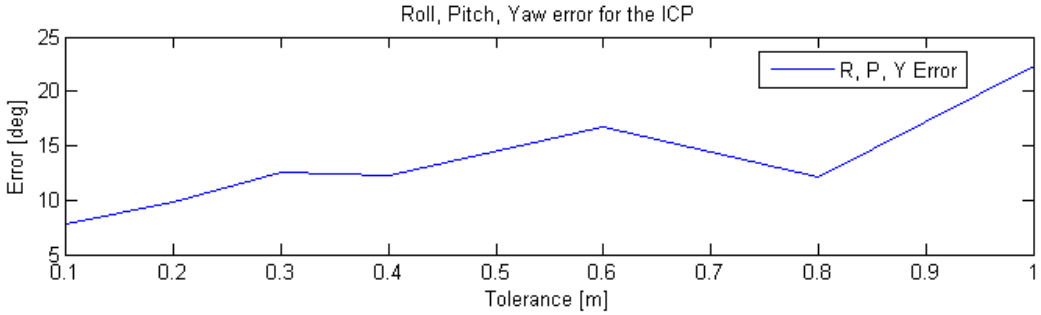


Figure 9.8 Average results of varying tolerance in the rotations

PARAMETER RESULTS

The results from the Figure 9.3, Figure 9.5 and Figure 9.7 are given in numeric form in Table 8

Table 8 Translation results

Test Nr.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Iteration (xyz)	0.062	**0.01**	0.1	0.069	0.071	0.11	NA	NA	NA
Point Nr. (xyz)	0.077	**0.01**	0.047	0.061	0.061	0.076	0.06	0.66	0.63
Tolerance(xyz)	0.051	0.05	**0.01**	0.06	0.2	0.079	0.15	NA	NA

The results of Figure 9.4, Figure 9.6 and Figure 9.8 are given in Table 9 below

Table 9 Rotation results

Test Nr.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
Iteration (rpy)	15.17	**12.55**	9.17	10.13	8.18	9.07	NA	NA	NA
Point Nr. (rpy)	8.87	**12.55**	10.20	9.07	8.8	7.23	8.91	10.29	8.71
Tolerance(rpy)	7.86	8.8	**12.55**	12.29	16.77	12.17	22.31	NA	NA

It can be seen that the bald cells contain the same values. This is caused by the fact that the bald cells contain the results of the standard configuration which appears in every experiment in order to make it easier to compare the results.

From the results given above, it becomes evident that the parameter with the most influence is the number of iterations followed closely by the tolerance. While the number of points used has almost no effect on the results.

9. 3. 2. ERROR ACCUMULATION

Further tests have been made on the accumulation of the errors in the ICP component of the algorithm. Since the ICP algorithm is not perfect, each point cloud that is being added has a small error with it. The more point clouds are being added, the larger the accumulated error will be at the end.

In order to test the accumulation of errors, a test has been set up. The test involves running only the ICP component of the algorithm over a series of point clouds. The first batch of point clouds is taken at a rate of three per second, while the second batch is taken at only 0.3 per second. This means that the first batch contains exactly 10 times the number of points of the second batch. Now the ICP is run over both batches, and the errors are recorded with the help of the Ground Truth.

Figure 9.9 and Figure 9.10 show the translational errors of the algorithm in the X, Y and Z axis. It immediately becomes clear that the errors of the scans which were taken at 3Hz are five times larger than the error of the scans taken at a lower rate. This is because there are more scans to contribute to the error addition. The fewer scans of the second batch contribute much less to the error causing it to be smaller than the error of the first batch.

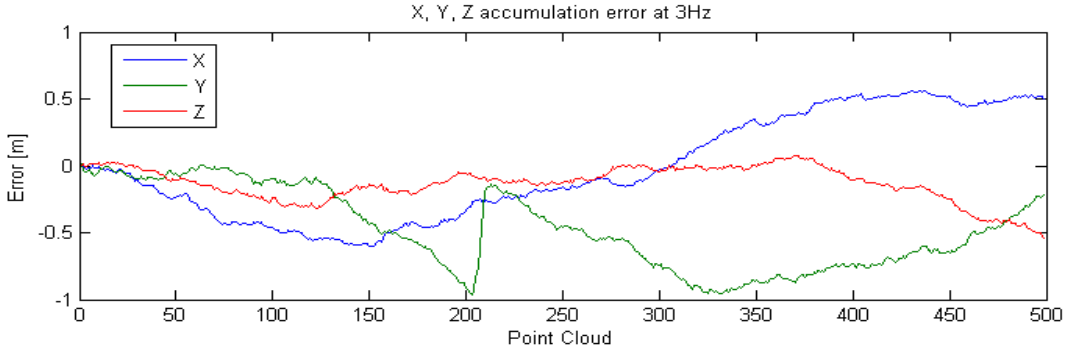


Figure 9.9 X, Y and Z errors of the first batch of scans.

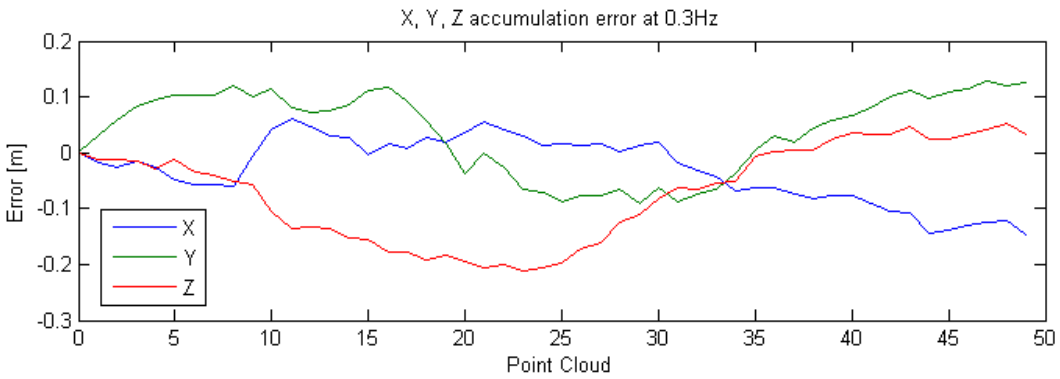


Figure 9.10 X, Y and Z errors of the second batch of scans

Note the sinusoidal waves which can be seen in some of the dimensions. This is caused by the rotation of the LIDAR around the target object. It also shows that one complete rotation of around the object is made in this period. Another interesting aspect which can be noted is that at the beginning, the first batch starts with a very small error that grows slowly, while the second batch immediately begins with a large error. This is caused by the fact that there are more clouds in the first batch and it is easier to align therefore causing a slowly increasing error. However, in the second batch 10 scans are skipped which causes a loss of information in the algorithm. Since less data is used, it is much more likely that a larger error can happen sooner. This can be seen as a discretization problem where the first batch has a high discretization and the second one has a low discretization.

Figure 9.11 and Figure 9.12 show the behavior of the roll, pitch and yaw of the point clouds. As was the case in the translational errors, the error increases slower in the first batch. However, there is a steep climb around the 200th frame. This steep increase is caused by a rapid change in geometry such as the appearance or disappearance of some features in the model.

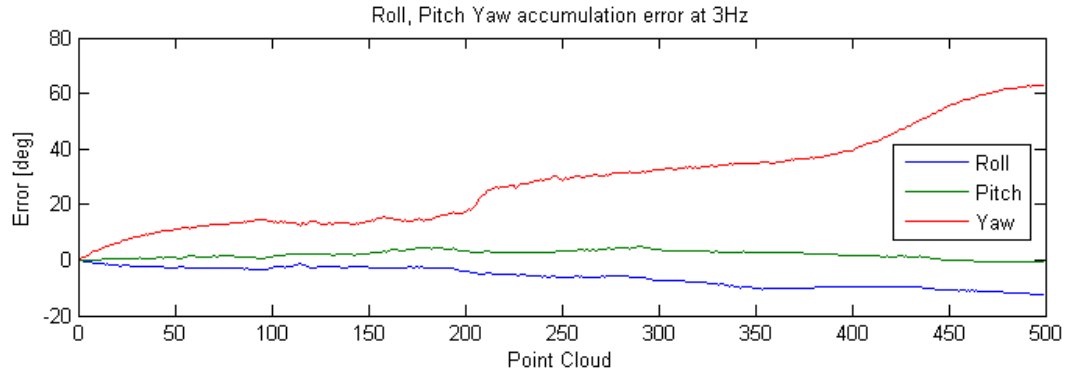


Figure 9.11 Roll, Pitch and Yaw errors of the first batch of scans.

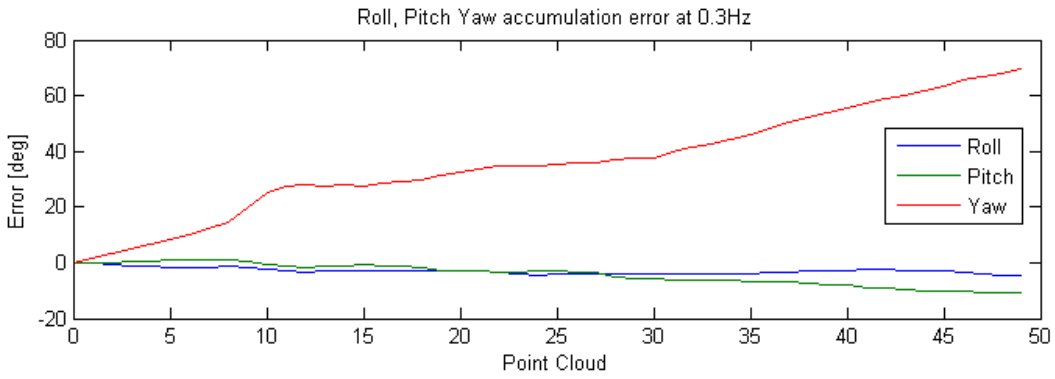


Figure 9.12 Roll, Pitch and Yaw errors of the second batch of scans.

In Figure 9.12 it can be seen that there is a large increase in the error from point cloud 9 to 10. This is the drawback of using fewer point clouds, since there is a higher chance of the points not matching. However, it is clearly visible that the large increase in error in Figure 9.11 does not appear in Figure 9.12, this is because the point clouds which generate this error are not considered in the second batch of examples. This shows that fewer point clouds can also mean avoidance of several errors. By the end it becomes apparent that no matter how many point clouds are being used, the final error is almost the same in both cases, going up to approximately 70º.

9. 4. GICP

Similarly to the ICP component of the algorithm, the GICP also has several parameters which affect the quality of the results. These parameters are:

- Point Cloud Number: Number of point clouds which the algorithm should take in. It will not take into account more point clouds than this number specifies.
- Similarity of the clouds: A percentage value which shows how different two point clouds need to be in order for all the connections between the two to be ignored. The difference between the point clouds is calculated based on the number of formed versus the number of connections that did not form.
- Iterations between PC: Number of iterations that the algorithm needs to do with the old point clouds before accepting a new oint cloud as input.
- Optimization Iterations: Number of iterations that the algorithm needs to perform at the end in order to improve the results before giving them out. His step can greatly improve the results of the algorithm.
- Kd tree points: Number of points from the point clouds which will be used to reconstruct the kd tree. A lower number, results in faster computation times, however, it also results in a greater loss of information.
- Search points: The number of points which are used to search in the kd tree for correspondences. Similarly to the previous value, a lower number results in faster computations, however it also results in a lower accuracy.
- Tolerance: A tolerance value which determines the distance at which connections can be formed and from which connections between points can no longer be formed.

As can be seen, there are clearly more parameters in the GICP than in the ICP component. Similarly to the procedure for the ICP, the variables above will also be varied in order to obtain results and the errors will be compared against each other to determine the most influential parameters.

9. 4. 1. PARAMETER TESTING

The standard configuration of the GICP is given in Table 10. However, unlike the ICP component, the GICP cannot function without previously running the ICP for pre-alignment. Therefore the standard configuration used for this set of experiments is the same as it is in Table 10(ICP Section).

Table 10 Parameters for the GICP algorithm

Parameter	Number	Units
Nr of Point Clouds	30	Integer
Similarity to other clouds	0.8	%
Iterations between PC	15	Integer
Optimization Iterations	150	Integer
Kd Tree Points	40	%
Search points	20	%
Tolerance	0.1	m

Of all the parameters above, the tolerance was not tested. This is because the tolerance is very closely related to the performance of the algorithm. If the tolerance is off by even a small amount, the algorithm will fail completely and give very wrong results, thus making comparisons for this value impossible

SESSION 1

First, the number of scans is being varied. For this session, three different values have been tried. The values tried are 30, 40 and 50 point clouds for the entire reconstruction. In this experiment, the rate at which scans are being taken is the only variable. The total time duration between the first scan and the final scan is kept the same, for consistency. The scans are also taken for a complete turn-around of the model. What this means in terms of angles between the scans is that for the 30 point cloud model the scans are taken every 12º while for the 40 and 50 point cloud models the scans are taken every 9 and 7.2 degrees respectively.

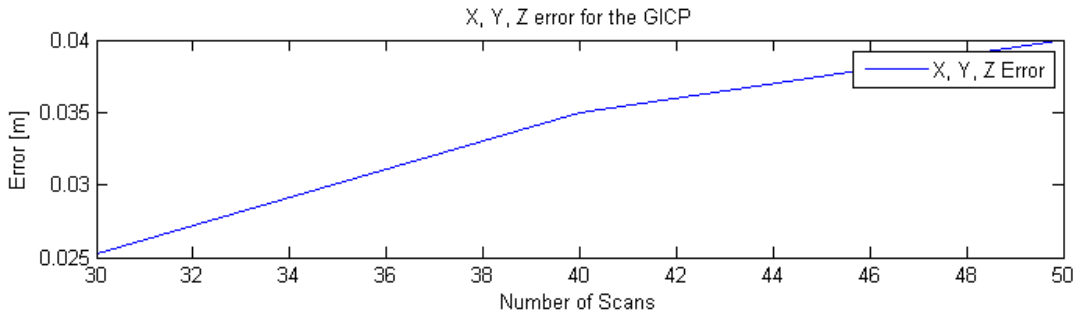


Figure 9.13 Translational errors of the GICP algorithm resulting by changing the number of point clouds

As Figure 9.13 and Figure 9.14 show the accuracy decreases with increasing numbers of scans. This is easily understandable since there is an increased amount of information in the algorithm. Also, for the 50th point cloud result the algorithm begin with an approximate error of only 7º compared to the 12º of the 30 point cloud version.

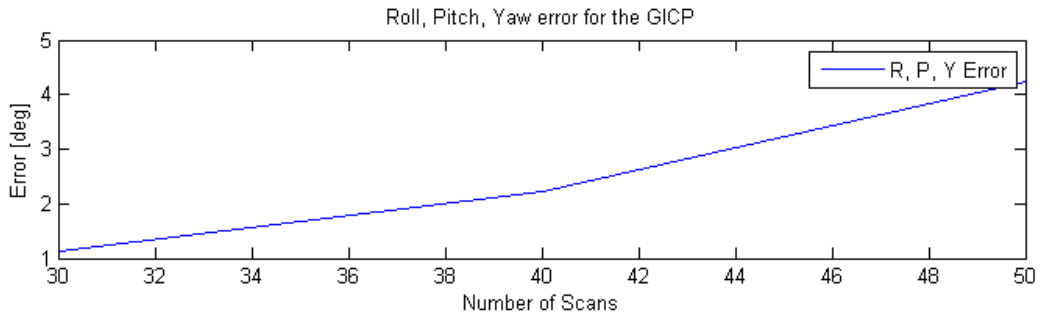


Figure 9.14 Rotational errors of the GICP algorithm resulting by changing the number of point clouds

As mentioned before in Chapter 9. 3. 2, as the number of scans increases, there is also a greater error accumulation. This in turn means that the final result will have more error accumulated. In order to fix this, there need to be more iteration after each point cloud is added for the algorithm to have more time to align all the point clouds. However, since the time is insufficient, the accumulated error does not have time to be taken out. In the end, this results in a slightly larger error.

However, it should be noted that there is a minimum of scans which can be used successfully to reconstruct an object. This number of scans depends greatly on the geometry of the target object, but a value as low as 24 scans can sometimes be enough.

Note, the additional error from the extra scans is not very significant to the overall quality of the model. The added error is very small and this aspect can be considered to not influence the quality of the trajectory reconstruction. However, the computational time required to compute everything is significantly increased.

SESSION 2

For this test session the similarity to other point clouds has been varied. This value tells the algorithm to ignore point clouds based on the similarity between the two point clouds. If the point clouds are very different, the algorithm will ignore the connection between these points. This allows the algorithm to classify its point clouds and to better form the image between the point clouds that have enough information to share.

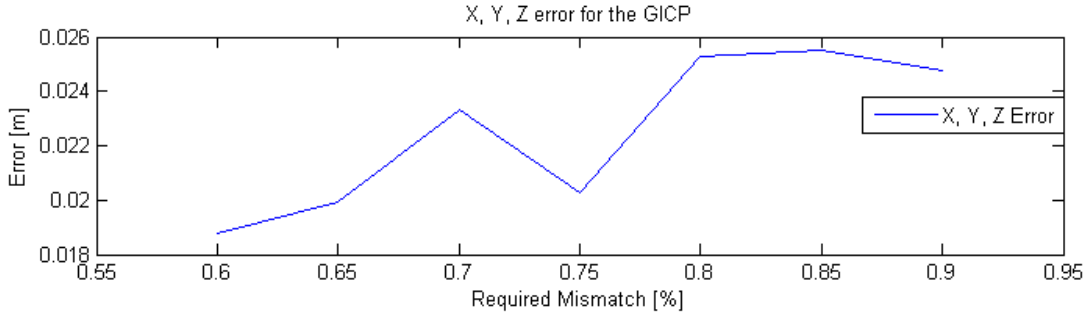


Figure 9.15 Translational error of the similarity of point clouds

Figure 9.15 and Figure 9.16 show the evolution of the error based on the difference between the points. The X-axis gives the percentage that is used in the algorithm. This percentage tells the algorithm that if the points are more than a certain percentage different, the point clouds will ignore each other and only form connections with other point clouds. It can be seen in Figure 9.15 that the more point clouds are

being ignored the more the error decreases. This is because too many connections can cause errors to stack up at higher rates.

However, the downside of this procedure is that if the value of the difference is too low, there could be point clouds that have no connections and thus resulting in crashes. This value needs to be balanced very carefully against the value of the tolerance and the numbers of points used in the algorithm in order to avoid crashes.

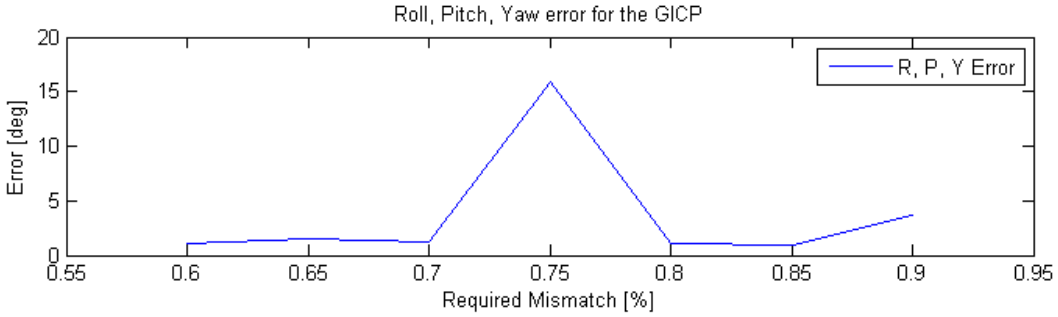


Figure 9.16 Rotational error of the similarity of point clouds

One interesting feature that can be seen in Figure 9.15 and Figure 9.16 is the spike which appears when the Required mismatch value is 75%. However, the spike is very different in the two images. In Figure 9.15 is appears to have a good effect, of reducing the error, while in Figure 9.16, the spike appears to drastically increase the error.

The spike is caused by the randomness when selecting point clouds. This can be traced back to the properties: Kd Tree Points and Search Tree Points. It can be seen in Table 10 that these parameters have values of 40% and 20% respectively. Moreover, since it is already known how the algorithm uses these values, (check Chapter 7. 6. 1) it is possible to come to the conclusion that randomness can indeed influence the final results. Because the points which will be used are selected randomly, it is possible to have points which, on average, provide a very bad representation of the point clouds. This can cause large mismatches between the point clouds and cause a poor alignment to form.

Moreover, it should be noted that the rotation and translation of the point clouds is not tightly correlated. This means that even though the model can experience an improvement in the rotation of the point clouds, it could experience a larger error in the translation of the clouds and vice versa.

If one looks at the y-axis of Figure 9.15 it is becomes clear that the improvement in translation is small when compared to the retrogression of the rotation. Also, when checking the y-axis of Figure 9.16 it becomes clear that (excluding the spike) the overall improvement is only 3^{o} which is not very significant. It can therefore be concluded that the Required Mismatch parameter does not influence the quality of the reconstruction. However, a large value will lead to small increases in error.

A large value for the Required Mismatch also results in more computation time. Therefore it is best to keep this value as low as possible in order to have a better accuracy and computation time. It is important to mention that this value, if too small, can cause the model to diverge very significantly. This means that there is a minimum limit which it needs to have. On average, a value of 60% gives good results and any value lower than this has been found to diverge.

Note: geometry and number of points influence the minimum possible value for this parameter. If more points are used, the value can go below 60%, but there is a risk of diverging. For some geometries, this

value needs to be larger than 60% as a value of 60% is insufficient and will cause a divergence in the model reconstruction and the trajectory.

SESSION 3

The third session varied the number of iterations done after adding one new point cloud in the algorithm. This number represents the number of iterations that the algorithm dies after adding one point cloud and before adding the next one. This alignment is crucial for reducing the global error and obtaining a better alignment from the next point cloud.

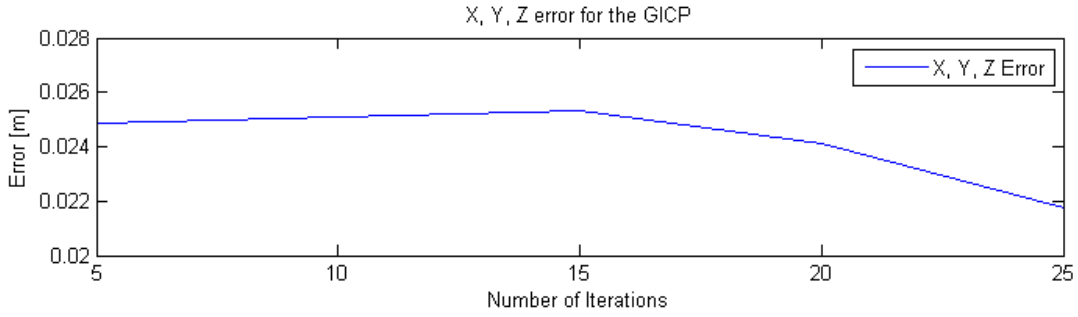


Figure 9.17 Translational error of the number of iterations

Figure 9.17 and Figure 9.18 show the error evolution of the algorithm as the number of iterations between the point cloud additions is changed.

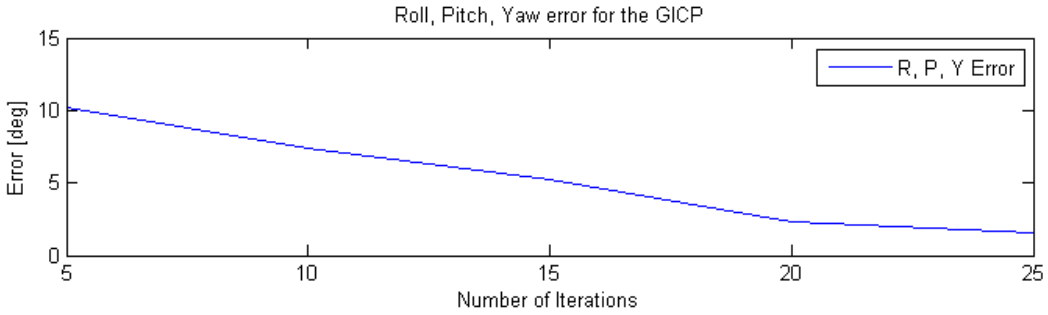


Figure 9.18 Rotational error of the number of iterations

Both figures show results which are expected. The errors decrease with an increasing number of iterations of the algorithm. Since the algorithm is running through more iterations it has more time to process the information and therefore reach a better solution.

SESSION 4

After all the point clouds have been added and aligned once the algorithm does one final optimization step. Because the point clouds that make up the model are still quite loose by the end of the last point cloud, an additional step is needed to tighten the point clouds. This last step has the goal of optimizing the model and improving the quality of the point cloud alignments.

In order to determine just how useful this part of the algorithm is, a test has been conducted.

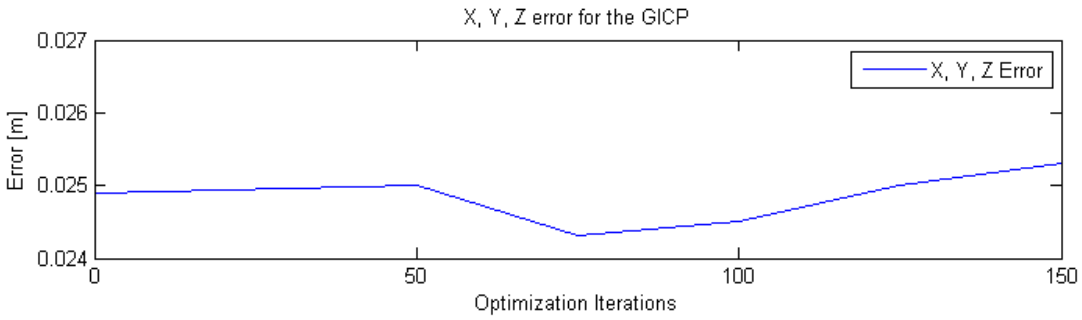


Figure 9.19 Translational error of the number of iterations at the end of the algorithm

Figure 9.19 and Figure 9.20 show the error evolution for which results by modifying the number of optimization steps that are done at the end of the algorithm. It should be noted that this is not done in real time and it is in fact a form of post processing which has the sole purpose of improving a trajectory which has already been obtained.

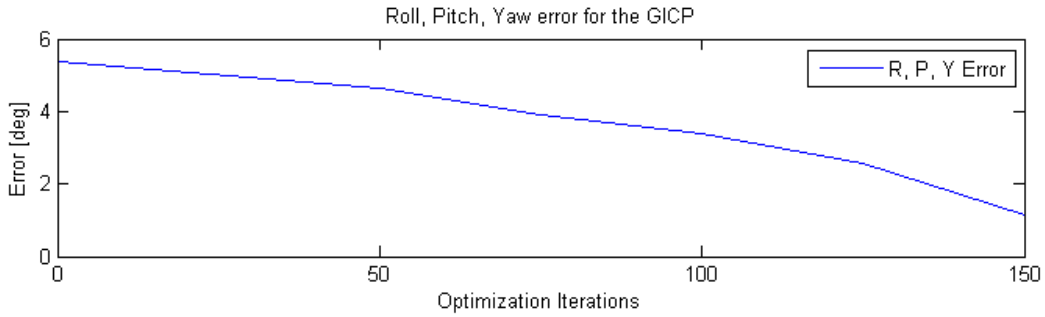


Figure 9.20 Rotational error of the number of iterations at the end of the algorithm

Figure 9.19 shows that the translational error is not affected by the change in the number of post processing simulations. This is because the translational error is very easy to correct by the algorithm. By the end of the algorithm the translation has already converged enough for the post processing steps to not influence the final result.

Figure 9.20 on the other hand shows that post processing does influence the final rotation. The rotational elements are harder for the algorithm to correct and therefore this aspect benefits from the post processing the most. The graph in Figure 9.20 shows a clear decrease in the error as the number of iterations increases.

SESSION 5

In the fifth experiment, the number of points used to form kd trees has been changed. The number of points used in for kd tree generation is a percentage value of the total number of points in the point cloud. The exact values used can be seen on the x-axis of the figures below while the y-axis shows the magnitudes of the errors.

It should be noted that with more points in the kd tree there is a very sharp increase in the time required for computation.

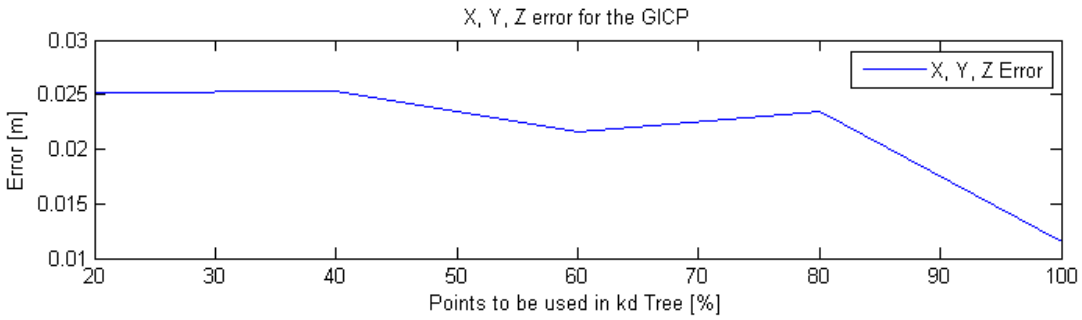


Figure 9.21 Translational error of the number of points used in the kd tree generation

The results of modifying the percentage of points to be used in the kd tree are shown in Figure 9.21 and Figure 9.22. The definition of this parameter can be seen in Chapter 7. 6. 1

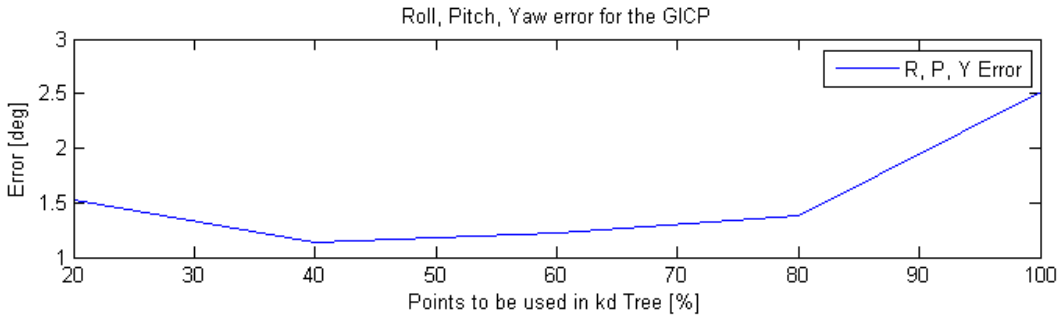


Figure 9.22 Rotational error of the number of points used in the kd tree generation

It can easily be seen in Figure 9.21 that the average error of the translation decreases. This decrease is caused by the increase in the number of points used. A larger number of points means that there is more information which can be used by the algorithm to align the points and make the translational error decrease.

However, unlike Figure 9.21, Figure 9.22 shows a different result. The rotational error appears to first decrease and later increase as the number of points increases. It is easy to explain why at first the error is large when only 20% of the points are used. This is caused by the fact that there is not sufficient information in the algorithm to achieve a good result. However, as the number of points increases, so does the error. This is because rotation is computed in a very different way than the translation. This means that the more points that there are, the higher the probability of running into a local minima. By having fewer points, the algorithm averages out the errors and this allows it to achieve relatively good results. The exact number of points that is best to use depends greatly on the geometry of the target satellite, however good values have been found to be between 20% and 50%.

It should be noted that a higher number of points in the kd Tree results in a significant increase in the computational time of the algorithm.

SESSION 6

The final experiment session is done by varying the number of points used to search the kd Trees. This number is expressed as a percentage and can be seen on the x-axis in Figure 9.23 and Figure 9.24. It should be noted that the values of this value depend on the number of points used to generate the kd Trees. This process is described in more detail in Chapter 7. 6. 1.

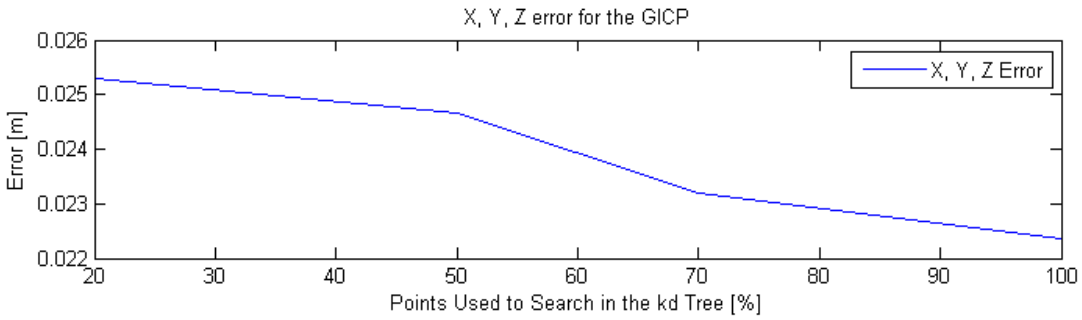


Figure 9.23 Translational error of the number of points used in the kd tree search

As can be seen in Figure 9.23 and Figure 9.24, the values of the errors do change as the percentage of points is changed, however the changes affect the translation and rotation in different ways as the two are not very tightly related.

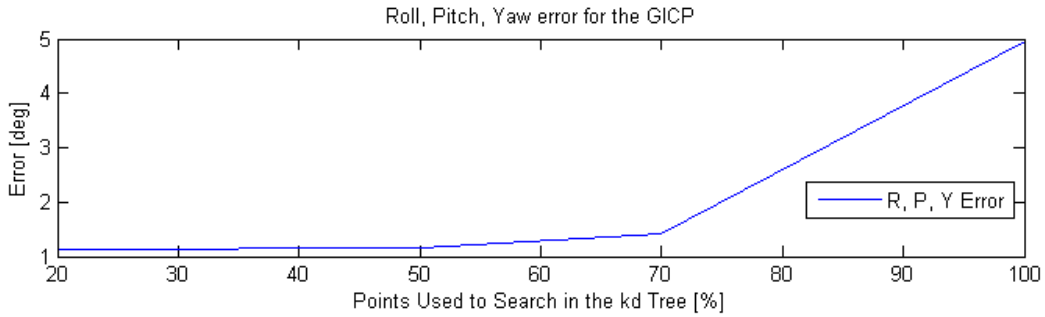


Figure 9.24 Rotational error of the number of points used in the kd tree search

Figure 9.23 shows the translational errors which result from modifying the percentage of points which are used to search the point clouds. It can be seen that in what concerns translations, the error does seem to improve. However, the improvement is very small, since the error starts off with 2.5 cm of error and in the end the error becomes approximately 2.2 cm. The improvement is thus only 3 mm which is insignificant when it comes to distances in space.

In Figure 9.24 it can be seen that the error of the rotations of the trajectory are affected much more than the translations. Moreover, the error increases instead of decreasing. At first this might be counterintuitive, because there is more information being used by the algorithm, which means that the result should be better. However, because all the points are used, it also means that the algorithm has a higher chance of reaching a local minimum. This prevents it from achieving a good result. By using fewer points the information is averaged out and the algorithm has a better chance of reaching a better solution. The number of points only influences the rotational aspect of the algorithm in a negative way by preventing a good solution, however in order to achieve a good translational error it is important to have a large quantity of points. For translation it is much more important to have more points. That is why the translational error improves with more points.

NOISE TOLERANCE

The last test that has been done for the algorithm is a test for the tolerance of the algorithm to different noise levels in the model.

When doing most of the tests, ideal conditions were used in order to have a degree of consistency. In an ideal case, there is no error in the measurement, which makes it much easier to analyze the data. If errors had been included, it would be impossible to correctly estimate the performance of the algorithm because the errors would be a random interference that could cause misinterpretation of the results.

On a normal mission, the average results which are expected are around ±4 cm. What this means is that the points which make up the point cloud can have an error of 4 cm in any direction.

The additional error of on the point clouds did not influence the result significantly. More tests have been done in order to test other error levels. The largest error levels which have been tested are ±20 cm with the results still very good comparable to tests that had no errors.

In some rare cases, the error was found to have improved the results. This is explained by the fact that by having a fuzzier surface, the point cloud actually has more features and it becomes easier for the algorithm to align than a point cloud with no surfaces.

9. 5. MODEL RESULTS

It is important to ensure that the algorithm can solve more than just the example given above. For this purpose, multiple objects have been taken and reconstructed by the algorithm. This section gives the results of four of the models used to verify if the algorithm works on a wider variety of satellites. The models used are of:

- DEOS Target: a specially designed vehicle on which the servicer will perform several actions including refuel, maintenance and deorbiting.
- Envisat: Is a satellite designed in Europe meant to study weather and to observe the Earth. It is now defunct and stuck in orbit. Because of its size and mass, it is important to decrease the possibility for the satellite to be destroyed.
- TerraSAR: is a satellite designed by Airbus Defense and Space to acquire high-quality radar images of the Earth.
- An Asteroid: a generic shape of an arbitrary asteroid meant to see if the algorithm can be used to reconstruct more natural shapes.

9. 5. 1. DEOS

The DEOS target satellite which can be seen in Figure 9.25 is specially designed to prove the concept of the servicer. It is meant to be a target satellite that needs to be repaired by the servicer, refueled and finally deorbited. It is designed to be approximately 1.5 meters in length in every direction which does not make it too large for the servicer to prove the mission concept.

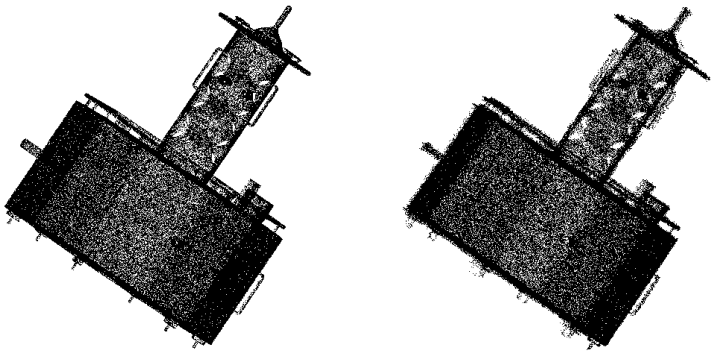


Figure 9.25 Two images of the DEOS target satellite. Left: the original Ground Truth of the target satellite. Right: The model reconstructed using the algorithm.

The reconstruction of the target satellite is reasonably accurate. The average error is 0.0126 m in the translational aspect and only 1.6385 degrees in rotation. This means that the trajectory of the target satellite can be reconstructed reasonably well from the reconstruction of this model.

One challenge with this model is that the shape (from above) is a hexagon. Because of this, there have been instances where the algorithm incorrectly aligned some of the point clouds to a wrong corner. However, the error rate for this particular shape is very low and insignificant if the parameters for the algorithm are chosen correctly.

9. 5. 2. ENVISAT

Launched in 2002, this satellite has been observing the Earth for many years until its mission has ended in 2012 just 10 years after launch. The huge satellite is now considered a ticking time bomb because of its size and its importance to preventing (or initiating) the Kessler syndrome. For this purpose, this satellite is one of the main targets of such a mission.

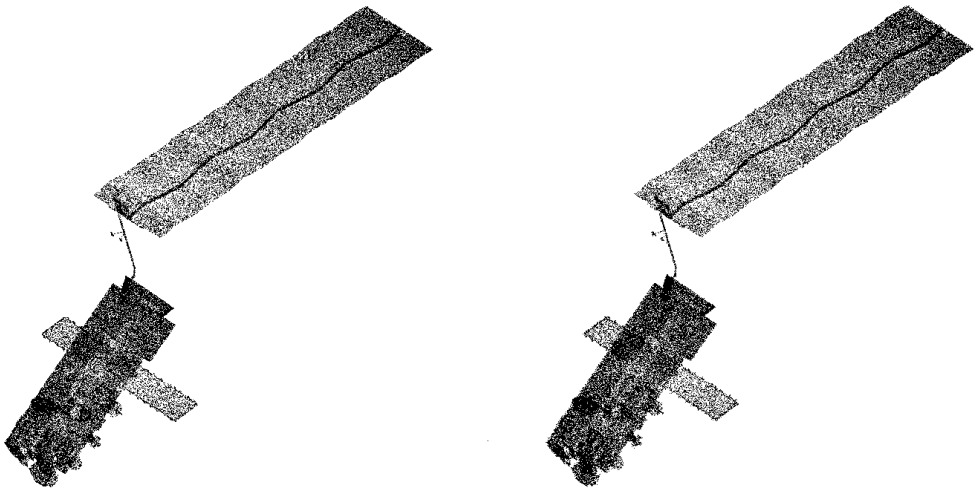


Figure 9.26 Two images of the Envisat satellite. Left: the original Ground Truth of the target satellite. Right: The model reconstructed using the algorithm.

Figure 9.26 shows the original model of the Envisat as well as the reconstructed version. There are no differences visible to the naked eye in these pictures.

The reconstructed model of the satellite has an average error of 0.0486 m in its translation. Although quite high, at 5 cm, this error is so large because the number of points that make up this model is significantly lower than the number of points that make up the point clouds of the DEOS target above. Moreover, the Envisat is considerably larger than the previous satellite. It measures 18 meters from one end to the other and contains a huge solar panel which can be seen from the top or bottom, but it is almost invisible from the left or right side of the satellite. This causes further errors to appear in the translational element of the problem.

The rotational error for this model has been computed to be 0.107 degrees. This value is extremely low and it means that the reconstruction, in terms of rotation, is very accurate. This is caused by the complex shape of the satellite and the many instruments on the bottom side which are excellent features for the algorithm. And features are extremely important for obtaining a good rotational accuracy, while the number of points is very important for obtaining a good translational error for the model reconstruction.

In the end this good model reconstruction will lead to a very good trajectory reconstruction around the target satellite.

9. 5. 3. TERRASAR

Built by Airbus Defense and Space, this satellite takes high resolution radar images of the Earth. It is an arbitrary satellite which has been chosen to determine how good the reconstruction of the satellite and the orbit can be using this algorithm.

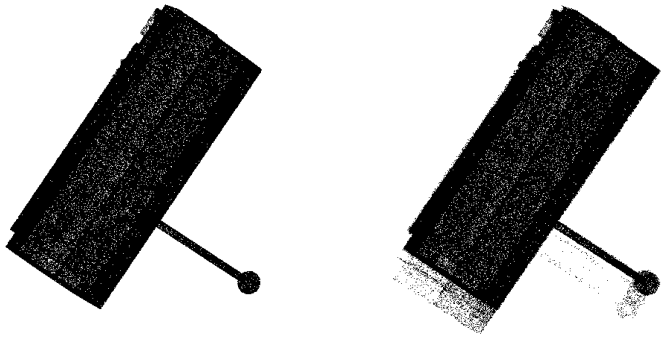


Figure 9.27 Two images of the TerraSAR satellite. Left: the original Ground Truth of the target satellite. Right: The model reconstructed using the algorithm.

As can be seen in the left image of Figure 9.27, there is a clear missalginment of some of the point clouds in this model This misalignment translates to an error in the trajectory reconstruction.

The translational error of the TerraSAR satellite is on average 0.0321 meters. This small error is achieved because the number of point clouds which are misaligned is very small when compared to the number of point clouds which are aligned. Therefore the average is brought lower.

For the rotational error, the average value is 0.3663 degrees, which is very small and it is achievable due to the many flat surfaces on the body of the target satellite.

9. 5. 4. ASTEROID

The final shape which was tested was an asteroid. This shape is very circular if seen from the top and elongated at the ends. It also contains several holes and irregularities on the surface to recreate a more uneven surface of an asteroid. Shape is 15 meters high and approximately 7 meter in the width and length. It has been scanned by the LIDAR simulator in the exact same way that the other satellites have beens canned.

Figure 9.28 shows the Ground Truth and the reconstructed model of the asteroid model. It can be clearly seen that the reconstructed model looks quite different from the original model. This is because, of the four models tested, this one was the poorest reconstruction.

The translational error of this reconstruction is 0.2937 meters. It is clear that the error from this model reconstruction is one order of magnitude larger than the previous reconstruction. In terms of rotation, the asteroid has even poorer results. The rotational error is 22.165 degrees which is so large that a trajectory reconstruction would give a completely wrong trajectory.

The reason for this is that the asteroid is very circular and the features on its surface were not strong enough to prevent the point clouds from sliding over each other. This means that the point clouds are arranged incorrectly and that the final result contains large errors.



Figure 9.28 Two images of the Asteroid. Left: the original Ground Truth of the target satellite. Right: The model reconstructed using the algorithm.

In order to avoid this from happening, the algorithm needs to have more points in order to extract features. Because of the round shape, this types of objects will always pose a serious problem to the algorithm.

9. 6. ORBIT RECONSTRUCTION

The orbit reconstruction process has already been described in Chapter 8. 3. And now the process described in that chapter will be applied to the reconstructed models seen in Chapter 9. 5. The reconstructed trajectories will then be discussed. It is very important for the trajectories to be understandable because the servicer will use this reconstructed trajectory to get close to the target satellite.



Figure 9.29 Trajectory reconstruction for the DEOS target satellite.

Figure 9.29 shows the trajectory reconstruction for the DEOS target satellite. The orbit appears to be quite bent and not rough at some points. This is caused by the misalignment of some of the point clouds. The rotational error for the DEOS satellite was reasonably large and this in turn caused the orbit to appear wavy in the image above.

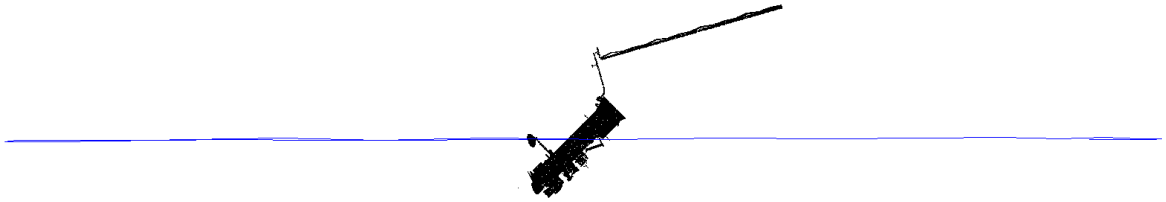


Figure 9.30 Trajectory reconstruction for the Envisat reconstruction.

Figure 9.30 shows the orbit reconstruction for the Envisat satellite. This is the best reconstruction which has been achieved of the four models. The orbit looks almost perfectly flat from the side and it clearly shows that the Enviat had the lowest error in the rotational aspect. Although the Envisat also had a large error in the translation, this appears not to affect the final outcome as much as the rotation error. This is of course caused by the fact that unlike the rotational error, the translational error is only added to the final trajectory position, while the rotation is multiplied.



Figure 9.31 Trajectory reconstruction for the TerraSAR satellite.

Figure 9.31 appears similar to Figure 9.30 in the sense that the result is quite clear and the orbit looks relatively smooth and well reconstructed.

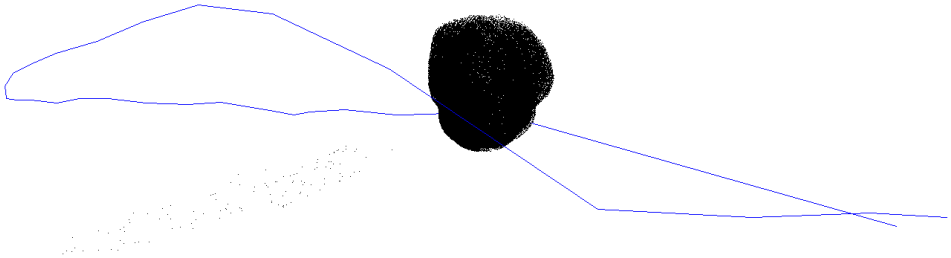


Figure 9.32 Trajectory reconstruction for the Asteroid.

Of all the models, the Asteroid had the poorest reconstruction. This is especially visible in Figure 9.32 where the trajectory is by no means a smooth reconstruction of the actual trajectory. This is caused by the enormous rotational errors of the model reconstruction.

9. 7. DATA PROCESSING AND TRANSFER

After the data is acquired it needs to be processed by the algorithm. However, the processing of the data is not necessarily done on the spacecraft itself.

There are two ways in which data can be processed after it is gathered by the LIDAR:

- The data can be processed on the ground by transferring the information from the LDIAR
- The data can be directly processed onboard the satellite and only the results sent back to Earth

Each of these solutions has advantages and disadvantages which make it attractive. In the case of transferring the data, the algorithm can benefit from extremely powerful hardware which is available on the ground and the calculations can take much less time. However, the satellite might not be able to communicate with the ground segment at all times.

The advantage of the onboard processing method is that it can be done in real time and there is no need to downlink the information. But this means that the onboard computer needs to do all the computation which, at times, could prove to be too much for the computer.

9. 7. 1. DATA GENERATION

It has been mentioned in previous chapters that the LIDAR functions at 3 Hz which results in three point clouds per second.

Considering that the average point cloud can contain up to 5000 points (depending on the LIDAR configuration), it is possible to compute the volume of data generated per second.

The values are saved in the double-precision floating-point format which means that each single value takes up 8 bytes of memory. (Kahan, 1997) It is common knowledge that in order to store a point in 3D space, three coordinates are needed, and since each of the coordinates takes up 8 bytes, one point will take up 24 bytes.

The average number of points in a point cloud is 5000, which means that one entire point cloud is 120000 bytes or 117.2 KB. This number is then multiplied by three because the LIDAR functions at 3 Hz and this results in 351.6 KB of data per second.

9. 7. 2. DOWNLINK RATE

The downlink rate is however significantly lower than the data generation rate. This is because not all the data needs to be used by the algorithm. In fact, because of the error addition it is better for the algorithm to use fewer point clouds.

It has been mentioned earlier that 24 point clouds is the minimum number of point clouds which is recommended for the algorithm. Using this principle, the downlink rate will be calculated in KB/turn around the target instead of KB/s.

Therefore, if the target is to be composed of 30 point clouds, then the downlink should be 3515.6 KB/turn or 3.4MB/turn.

Page intentionally left blank

10 CONCLUSION

The aim of this thesis was to determine whether it is possible to reconstruct the trajectory of a servicer satellite around a target satellite. The constraints were that only information obtained from the target satellite body can be used to determine the orbit of the satellite. This included scans of the target satellite which were obtained with the help of a LIDAR. By reconstructing the target satellite it was possible to derive the position of the servicer around the target satellite.

For the purpose of aligning the target satellite model, a generalized version of the Iterative Closest Point algorithm has been used. This algorithm allows the servicer to flexibly acquire data from its target and integrate it into the algorithm for processing. The algorithm then uses a global optimization method to align all the point clouds at the same time while taking into account the effect of every point cloud on every other point cloud.

In order to make this algorithm successful, several other methods of obtaining a reconstructed model have been tested. These algorithms are known as the Naïve Method and the Global Iterative Closest Point. These two methods had to be redesigned in order for them to function in such a way that the results of one of them would be used to improve the results of the other and vice versa. The resulting synergy allowed the algorithm to perform much better at the task of reconstructing a 3D model based on point cloud data.

Naïve Method

- The naïve method is a very simple implementation of the ICP algorithm which proved extremely valuable especially when prealigning the point clouds of the model. This method has been successfully used to align two point clouds at a time. Each time a new point cloud was added to the algorithm, the Naïve Method would reduce the error by a very large factor, thus making the job of the next step much easier.

Global Iterative Closest Point

- Taking the input from the Naïve Method, and from pervious runs of the GICP, this step would align all the point clouds at the same time. It does this by using an Imaginary Point Cloud to align all the other point clouds one at a time in a single step. The Imaginary Point Cloud is made from the connections of the other point clouds used merely as a helping object in the task at hand. The alignment process can even close loops and the result of loop closures can be clearly seen in the results. Another advantage is that the algorithm does not take into account the order in which the point clouds are taken, thus allowing it to close any number of loops equally well as long as there is enough information provided.

Processing Speed

- Because the algorithm gets slower with every point cloud added to it, eventually it becomes too slow to be considered useful. In order to be used in real time, the number of point clouds in the algorithm needs to be kept quite low with an average of 150000 points in the algorithm at any given time. This number also depends on the computer architecture providing the calculations.

Accuracy

- The algorithm has been shown to be able to perform very well on all the satellite targets which have been provided to it. Good results have been obtained regardless of the size of the target

satellite or the types of features which it contains. However, the algorithm has also been shown to have some weaknesses to certain shapes and when there is a lack of features. It is especially poor at reconstructing spherically shaped objects which do not have very prominent features on their surfaces.

Because of the nature of the algorithm, it is very flexible with respect to the input point clouds which go in. As long as there isn't a gap too large between two point clouds, and there is enough information stored in them, a reasonably good alignment can be achieved. The point clouds do not even need to have a fixed interval of time between being added to the algorithm. And the order in which these point clouds are added is irrelevant, as the algorithm will perform just as well, because the order is not a defining factor inside the algorithm. The algorithm is however quite dependent on the reconstruction parameters which need to be selected by the user. Without very good parameters selected the algorithm can very quickly diverge and give a very poor result in the end. However, with the proper parameters selected, the result can be very accurate.

In the end, this thesis was successful in creating a real-time algorithm which takes input data from a LIDAR senor and uses that data to reconstruct the target model. This was done using a novel way to represent the data inside the algorithm (imaginary point cloud) and through different selection methods. The algorithm can then reconstruct the trajectory of the observer based on the 3D reconstruction of the model. Other novel aspects in this thesis are the extreme flexibility of the algorithm and the possibility for the user to influence the convergence parameters.

Final conclusion: It is possible to reconstruct the trajectory of a servicer satellite by first reconstructing target satellite using a global version of the ICP algorithm in order to align different scanned images of the targeted satellite.

11 RECOMMENDATIONS

Since this is only the first implementation of an algorithm that performs trajectory reconstruction of an orbiter based on the model reconstruction of a target satellite, there are a lot of promising new implementations which can be coded in order to make the algorithm faster and more accurate. The recommendations given here are not the only possible improvements, but merely a collection of ideas, since the number of possible improvements/modifications for such an algorithm can be very large.

Most of the implementations discussed below are features which can be either added to the algorithm or they can be made to replace existing modules of the algorithm. Especially since most of them deal with forming connections and manipulating the existing connections between the points from the point clouds.

Some of the recommendations given below were once considered to be implemented in the algorithm described in this thesis, however due to different constraints such as time, computational resources and lack of certainty that they would provide significant improvement; it was decided to abandon them in order to have more time dedicated to the design of the actual algorithm.

The following are a few recommendations for modifying the algorithm in order to make it more efficient, and more flexible:

Use weights

The simplest way to achieve better results is by using a weighing function on the connections between points. The weighting function will result in a different distribution of forces in the model. Common ways to weigh connections is to base the weight on distance. Some functions will give more weight to two points which are close to each other, while other functions will give more strength to points which are far apart. Each of these two methods has its own advantages and disadvantages.

For example, the advantage of assigning higher weights to points which are closer to each other is that there is a much higher probability that wrong points will not have a strong enough influence on the model and thus will not spoil a good reconstruction. The disadvantage is that the clouds could become stuck together very fast because as soon as two points get very close, the connection strength is very large, and further points cannot be aligned because they have a weaker bond.

The advantage of having higher weights on points which are further apart is that the model has more movement between the point clouds. This is because points which are close have almost no influence, while points which are further away have a large influence. This means that there is a smaller probability of running into a local minimum, however, the disadvantage is that points which are too far away, and are therefore errors, will have a large influence.

A third probability is to have a varying weighing function which changes as the simulation progresses.

Use ICP more often than the GICP

The algorithm can also be easily modified to use one component more often than another component. Therefore, with minimal modification it is possible for the algorithm to use more point clouds. However, instead of applying the GICP over all of them, it is possible for the algorithm to be programmed to apply the GICP component on only half of the points.

Generally, more information means a higher chance of correct alignment. However, this also means that the algorithm runs slower. To avoid a very slow running algorithm, the task can be divided over the different components of the algorithm. It can load a high number of point clouds and align all of them using the Naïve Method (which is very fast) and it can apply the GICP only on every second point cloud. This means that there will be more information in the algorithm and that the computation time will not be much longer.

If the algorithm is designed to take in 30 point clouds, then this means that for a 360^o view of the target, the clouds need to be 12^o apart from each other. This large difference can cause some of the point clouds to be misaligned by the Naïve Method and almost impossible to correct later on. By using 60 point clouds, this starting error is reduced to only 6^o. However the increased number of point clouds means that the GICP will be very slow. Therefore, the Naïve Method can be applied to all point clouds and the GICP can be applied only every second point cloud to save time.

Improve corner overlaps

Corners have presented problems for the algorithm. This is because after reconstruction, it has been discovered in multiple instances that the corners are overlapping. This is an undesirable phenomenon since it causes the end result to be poorer than it would be possible.

This error can be as large as 10 cm depending on the size of the target satellite. Such a large error can be very important for trajectory reconstruction.

Automate parameter choosing process

Right now, the algorithm is dependent on approximately twenty input parameters which dictate how the algorithm will function. The number used to be larger until some of the parameter have been excluded. It is possible to hav fewer parameters by automating the selection for some fo them. The way to do this would be to choose the parameters based on properties of the target object. This can come from the average error of the measurements or the size of the target object.

The basic idea being to use simple object traits to determine values for the algorithm's input parameters in order to make it more independent of humans.

Use edge detection

Edge detection can be used to align the different point clouds from the model. It is known that two planes are very hard to align. However, if there is a wireframe, the edges would form excellent features which can guide the alignment process. This method has the added advantage that the corners will not overlap and therefore solving some of the problems encountered previously.

One disadvantage of this method is that an algorithm that finds edges is very computationally expensive and could use up a lot of resources. On the other hand, it will eliminate up to 90% of the points from the model, because very few points are on the edges, while most of them are on surfaces.

This approach might be unable to perform well for spherically shaped objects. This is because it will be very difficult to detect edges on this type of object.

Use a SLAM algorithm to optimize the path after the GICP algorithm

A SLAM algorithm can be used to optimize the results of the algorithm. Although this approach will not be able to completely resolve the problems, it can be used to improve the results and reduce the errors.

After the algorithm gives out the resulting transformation matrices, the trajectory needs to be computed. This algorithm will look at the trajectory and estimate the velocity of the satellite to try to "guess" the position of the next measurement. Using this guessed value, the algorithm will compare it to the actual position o the measurement and determine if the alignment can be improved or not. If it turns out that the measurement can be improved, the algorithm can try to realign the point clouds in question. Otherwise it moves to the next position on the trajectory.

In the end it will be possible to optimize the entire trajectory of the servicer around the target satellite.

Approximate basic shapes

It is possible to align the model by approximating its features using basic shapes and then aligning the shapes to fit as good as possible with each other. This method requires an algorithm which can reduce complex geometry to simple shapes in order to function. However, after it is completed, it could greatly simplify the task of aligning different point clouds.

Eliminate bottlenecks

A sure way to make the algorithm perform faster is to determine the bottlenecks of the entire algorithm. After detecting these bottlenecks, they have to be optimized to perform faster. This can be done by using more optimized code or by parallelizing the process.

The kd Tree searcher described above can be easily parallelized because it repeats the same action tens of thousands of times. By parallelizing this process, the speed of the algorithm can be increased by entire orders of magnitude.

Other bottlenecks can also be discovered and improved accordingly to obtain a better performance.

Use patches of point clouds instead of full Point Clouds

It is possible to use only a small part of the point cloud to search for correspondences instead of using the entire point cloud. This has the advantage of using a very small number of points instead of the entire point cloud; however, one disadvantage is that information is being lost. Moreover, the patches selected should contain enough information to correctly guide the alignment of the model.

If the patch selected is a simple plane, then the translation are impossible to derive, however, if the selected patches contain features, this will make the alignment much more straight forward, therefore a method of selecting good patches needs to be selected.

This method is different from excluding random points from the point clouds because it keeps all the information in that patch. On the other hand, if random points are removed from the point cloud, there is a loss of information overall because points which represent features are also removed. However, by selecting patches with features, the points which make up the features are preserved and therefore enough information is preserved to have a good alignment.

Better feature selection

It is incredibly important to have correct features selected. A feature which is unique to one part of the target object will ensure that that part is aligned correctly. Wrong features can cause the object to be incorrectly aligned and to have overlaps of regions which should not overlap.

Having a better feature selector will therefore allow the algorithm to devote its resources to important aspects of the problem.

12 BIBLIOGRAPHY

Agarwal, P. K., & Arge, L. (2006). From Point Cloud to Grid DEM : A Scalable Approach 1 Introduction, 1–15.

Airbus-Defence & Space. (2012). Airbus Defence and Space wins DEOS contract to demonstrate in-orbit servicing. Retrieved March 01, 2014, from http://www.space-airbusds.com/en/press_centre/-kx9.html

Amberg, B., & Vetter, T. (2003). *Optimal Step Nonrigid ICP Algorithms for Surface Registration*.

Astos, S. G. (2014). Astos Website. Retrieved May 13, 2014, from https://www.astos.de/

Astrium, C. S. (2004). Agrani 2 (ex Thaicom 4) / 2003 - cancelled. Retrieved March 25, 2014, from http://cs.astrium.eads.net/sp/spacecraft-propulsion/heritage/page-139.html

Astronautica, E. (2014). Envisat. Retrieved March 01, 2014, from http://www.astronautix.com/craft/envisat.htm

Breitenmoser, A., & Tishchenko, I. (2010). Surface Reconstruction from Point Clouds.

Brown, B. J. (2004). *Non-Rigid Range-Scan Alignment Using Thin-Plate Splines*.

CLARK, S. (2008). Europe's new space freighter prepares for maiden voyage. Retrieved March 25, 2014, from http://www.spaceflightnow.com/ariane/v181/preview.html

CNRS. (2013). Planification de missions pour ramasser les debris spatiaux. Retrieved February 27, 2014, from http://images.math.cnrs.fr/Le-site-MPT-propose-une-breve-1595.html

Couto, D. De, & Napoli, J. (1998). kD Trees. Retrieved June 09, 2014, from http://groups.csail.mit.edu/graphics/classes/6.838/S98/meetings/m13/kd.html

Cracknell, A. P., & Hayes, L. (2007). *Introduction to Remote Sensing (2 ed.).* Boca Raton, FL: CRC Press. Retrieved from http://books.google.nl/books?id=0acsAQAAMAAJ

CSIRO. (2013). Light detection and ranging (lidar). Retrieved March 25, 2014, from http://www.csiro.au/Outcomes/Food-and-Agriculture/LightDetectionLidar.aspx

Dawson, J., & Hu, G. (2013). *Asia Pacific Reference Frame (APREF)*. Manila, Philipines.

Delaunay. (2006). Wikimedia. Retrieved July 16, 2014, from http://upload.wikimedia.org/wikipedia/commons/3/38/Delaunay_triangulation_small.png

Delaunay, I. (1993). Incremental Delaunay Triangulation.

Diep, F. (2012, June 8). Have lasers located lost city of gold sought by conquistadors in 1500s? *Tech News Daily*. Retrieved from http://www.foxnews.com/scitech/2012/06/08/lasers-help-discover-lost-city-gold-sought-by-conquistadors-in-1500s/

DLR. (2012). Defekte Satelliten im All einfangen: DLR vergibt Auftrag an Astrium. Retrieved March 01, 2014, from http://www.dlr.de/dlr/desktopdefault.aspx/tabid-10212/332_read-5173/#/gallery/7516

Doc-Ok. (2013). Meet the LiDAR Viewer. Retrieved March 25, 2014, from http://doc-ok.org/?p=432

Eaton, M. (1983). Multivariate Statistics: a Vector Space Approach, 116–117.

ESA. (2014). ESA declares end of mission for Envisat. Retrieved February 27, 2014, from http://www.esa.int/esaCP/SEM1SXSWT1H_index_0.html

ISR. (2012). 4D Facial Dynamics. Retrieved March 27, 2014, from http://dynface4d.isr.uc.pt/database.php

Jeffrey S. Evans, & Hudak, A. (2012). Lidar Measures 3-D Vegetation Characteristics. Retrieved March 25, 2014, from http://www.forestry.gov.uk/forestry/INFD-6S8LQS

Jia, Y. (2013). Singular Value Decomposition, 1–9.

Kabsch, W. (1976). A solution for the best rotation to relate two sets of vectors, (32), 922.

Kahan, P. W. (1997). Lecture Notes on the Status of IEEE Standard 754 for Binary Floating-Point Arithmetic Introduction : Table of Formats ' Parameters :, (May 1995), 1–30.

Kessler, D. ~J., & Cour-Palais, B. ~G. (1978). Collision frequency of artificial satellites: The creation of a debris belt, *83*, 2637–2646. doi:10.1029/JA083iA06p02637

Kjer, H. M., & Wilm, J. (2010). *Evaluation of surface registration algorithms for PET motion correction Bachelor thesis*.

Lu, F., & Milios, E. (1997). *Globally Consistent Tange Scan Alignment for Environment Mapping*. Ontario, Canada.

Marks, P. (2009). Satellite collision "more powerful than China"s ASAT test'. Retrieved February 27, 2014, from http://www.newscientist.com/article/dn16604-satellite-collision-more-powerful-than-chinas-asat-test.html

NASA GSFC. (2006). SCANNING RAMAN LIDAR. Retrieved March 25, 2014, from http://ramanlidar.gsfc.nasa.gov/instruments/scanning raman lidar/

Nelson, P. (2011). 3D Mapping for Robotic Search and Rescue, (May).

NOAA. (2013a). LIDAR—Light Detection and Ranging—is a remote sensing method used to examine the surface of the Earth. Retrieved June 04, 2014, from http://oceanservice.noaa.gov/facts/lidar.html

NOAA. (2013b). What is LIDAR? Retrieved March 25, 2014, from http://oceanservice.noaa.gov/facts/lidar.html

NOAA. (2014). National Oceanic and Atmospheric Administration. Retrieved May 06, 2014, from http://oceanservice.noaa.gov/facts/lidar.html

Oudot, S. (2002). Delaunay Triangulation.

Page, G. S. (2010). NIGCOMSAT 1, 1R. Retrieved March 25, 2014, from http://space.skyrocket.de/img_sat/nigcomsat-1__1.jpg

Page, G. S. (2011). Kopernikus 1. Retrieved March 25, 2014, from http://space.skyrocket.de/img_sat/dfs-kopernikus__1.jpg

Page, G. S. (2014). EchoStar 1, 2. Retrieved March 25, 2014, from http://space.skyrocket.de/doc_sdat/echostar-1.htm

PCL. (2014). Point Cloud Library. Retrieved May 28, 2014, from http://pointclouds.org/

Princeton University. (2005). 1 Derivation of Point-to-Plane Minimization 2 Analysis of Stability, 1–4.

Propshop Ltd. (2012). 3D Scanning. Retrieved March 25, 2014, from http://www.propshop.co.uk/3d-scanning/

Pulli, K. (1998). *Multiview Registration for Large Data Sets*.

Rossi, A. (2011). Space debris, *6*(1), 10595.

Rusinkiewicz, S., & Levoy, M. (2001). Efficient variants of the ICP algorithm. *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, 145–152. doi:10.1109/IM.2001.924423

Sapiro, G., Hall, V., E, C. S. S., & Mémoli, F. (2004). IMA Preprint Series # 1978 (.

Science, C. (1980). Two Algorithms for Constructing a Delaunay Triangulation 1, *9*(3).

Shafarevich, I. R. (1961). BORIS NIKOLAEVICH DELONE (on his seventieth birthday). *Russian Mathematical Surveys*, *16*(3), 151. Retrieved from http://stacks.iop.org/0036-0279/16/i=3/a=M04

Sorkine, O. (2009). Least-Squares Rigid Motion Using SVD, (February), 1–6.

sourceForge.net. (2012). JavaGeom: geometry with Java. Retrieved June 08, 2014, from http://geom-java.sourceforge.net/demos/drawRangeKDTreeDemo.html

Sox, L., & Wickwar, V. B. (2012). Results from an Extremely Sensitive Rayleigh- - Scatter Lidar. *NASA Space COnsortium*, 1–6.

SpaceTech. (2014). SpaceTech GmbH Immenstaad. Retrieved May 29, 2014, from http://www.spacetech-i.com/deutsche-orbitale-servicing-mission.html

Toldo, R., Informatica, D., Beinat, A., Crosilla, F., & Cotonificio, V. (2007). *Global registration of multiple point clouds embedding the Generalized Procrustes Analysis into an ICP framework*. Verona, Italy.

USGS. (2013). USGS Science for a Changing World. Retrieved March 25, 2014, from http://lidar.cr.usgs.gov/

USGS. (2014). Observing Earth Today and Tomorrow: A National Plan. Retrieved March 25, 2014, from http://www.usgs.gov/blogs/features/usgs_top_story/observing-earth-today-and-tomorrow-a-national-plan/

Young-Hoo Kwon. (1998). Computation of the Rotation Matrix. Retrieved July 16, 2014, from http://www.kwon3d.com/theory/jkinem/rotmat.html

APPENDIX A: TRANSFORMATION MATRICES

The point clouds need to be rotated and translated in order to align them to different frames of reference. For this purpose, a transformation matrix is used. Another use of transformation matrices is to change the frame of reference of objects to other frames of reference.

A transformation matrix can have different sizes, but for a 3-dimensional world, a transformation matrix is normally a 3x3 matrix which contains values for rotation. However, since translation is also a very important factor in a transformation, the translational component will also be included in the transformation matrix. Although scaling can also be represented, it is ignored in this paper as it provides no usefulness to the problems at hand which involve point clouds.

The goal is to represent both the rotation and the translation in a single transformation matrix. This provides a simple and easy to use variable. The rotational component of the transformation matrix is a 3x3 submatrix, while the translational component is a 1x3 submatrix. For this master thesis, the transformation matrices will be written in the same format as the matrix given below in Equation (52).

$$T = \begin{pmatrix} R & 0 \\ t & 1 \end{pmatrix} = \begin{pmatrix} R_1 & R_2 & R_3 & 0 \\ R_4 & R_5 & R_6 & 0 \\ R_7 & R_8 & R_9 & 0 \\ t_1 & t_2 & t_3 & 1 \end{pmatrix} \quad (52)$$

Where:

- R is a rotation subatrix
- t is the translation submatrix
- R_x represents an element of the rotation matrix
- t_x represents the x[th] element of the translation matrix

It is important to note the last column which contains three zeros and one 1. This column is present for the only purpose of having a square matrix. This allows the matrix to be easily invertible, which is very useful to do when it comes to transformations.

The rotation submatrix is obtained by multiplying three primary rotational matrices along each of the axes. The primary rotational matrices are given in Equation (53).

$$\left.\begin{aligned} R_x &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{pmatrix} \\ R_y &= \begin{pmatrix} \cos(\theta_1) & 0 & \sin(\theta_1) \\ 0 & 1 & 0 \\ -\sin(\theta_1) & 0 & \cos(\theta_1) \end{pmatrix} \\ R_z &= \begin{pmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 \\ \sin(\theta_2) & \cos(\theta_2) & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{aligned}\right\} R = R_x R_y R_z \quad (53)$$

Where:

- θ is an angle that the object is rotating around the x axis
- θ_1 is an angle that the object uses to turn around the y axis
- θ_2 is the angle that the object rotates around the z axis

Each of the matrices rotates the object about one axis (x, y or z) the multiplication of the three matrices results in the combined effect of all the rotations in a single 3x3 matrix.

The translation element consists of the positions in Cartesian coordinates: $t = (X \quad Y \quad Z)$

As can be seen in Figure A.1, in order to get from position A to position B matrix T_1 is used and to get from position B to C matrix T_2 is used. However, if one wants to go directly from A to C then the resulting multiplication is simply T_{12} which can be seen below:

$$T_{12} = T_1 T_2 \tag{54}$$

A representation of Equation (54) can be seen below in Figure A.1.

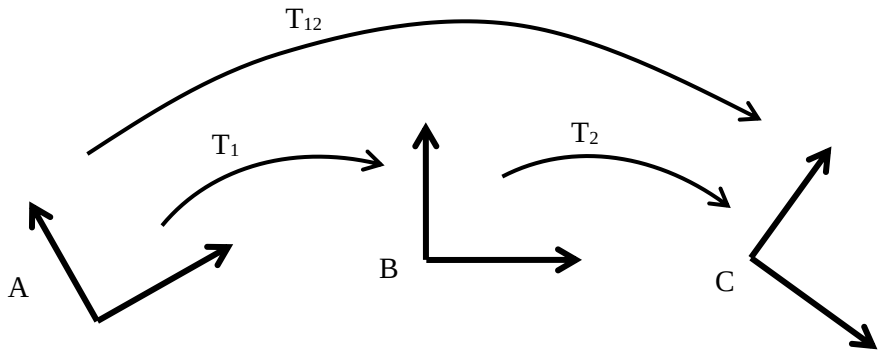


Figure A.1 Transformation between three reference frames

Furthermore, in some cases it will be necessary to reverse the transformation. For example, one might want to go from frame B to frame of reference A, or from C to B. In order to reverse a transformation, it is sufficient to invert the transformation matrix and multiply it by the result to get the initial state. For example, if one wishes to get from position C to position B in Figure A.1, it is sufficient to invert matrix T_2. The inversion can be seen in Equation (55).

$$T_2' = T_2^{-1} \tag{55}$$

Using these simple explanations it is possible to express the entire process described in the previous section in mathematical expressions.

APPENDIX B: SVD

The Singlaur (SVD) Value Decomposition is a means of factoring matrices into linear approximations which show the underlying structure of the matrix. The SVD has many useful applications in areas such as data analysis, signal processing, pattern recognition, image compression and even weather prediction.

It is the only main result about linear transformations between two different spaces. The SVD says that by choosing a suitable bases for the spaces, the transformation can be expressed in a simple matrix form, a diagonal matrix. Geometrically speaking, the Singular Value Decomposition means that circles in a domain are transformed into ellipsoids in the codomain. Since the transformations may not be injective, the dimension of the ellipsoid is at most the dimension of the sphere. Therefore, some distortion about the axes is obtained. This effect can be observed in Figure B.1.

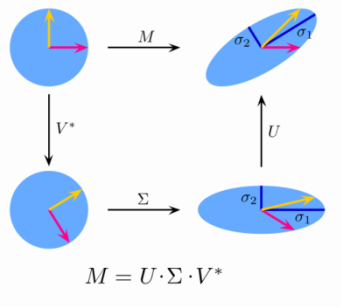


Figure B.1 A circle being transformed using a SVD

The symbols from Figure B.1 are explained in the equation given below.

The Singular Value Decomposition can be seen from three different points of view:

- It is a method for transforming correlated variables into a set of uncorrelated variables which do a better job of exposing relationships among the original data
- A method for identifying the dimensions along which data points exhibit the most variation
- It can be used as a method for data reduction as it is possible to find the best approximation of the original data using fewer dimensions

The SVD is based on a theorem from linear algebra which states that a rectangular matrix can be broken down into the product of three matrices:

- An orthogonal matrix U
- A diagonal matrix S or Σ
- The transpose of an orthogonal matrix V

Based on the above explanations, the formula for computing the SVD of an arbitrary matrix A is given below in Equation (56).

$$A = U\Sigma V^T \qquad (56)$$

Where the following hold:

- $U^T U = I$
- $V^T V = I$

- The columns of U are orthonormal eigenvectors of AA^T
- The ccolumns of V are orthonormal eigenvectors of A^TA
- S is a diagonal matrix containing the square roots of eigenvalues from U and V

The equation described above can be seen in matrix form in Figure B.2. This figure shows the matrices involved with all the dimensions in order to facilitate understanding.

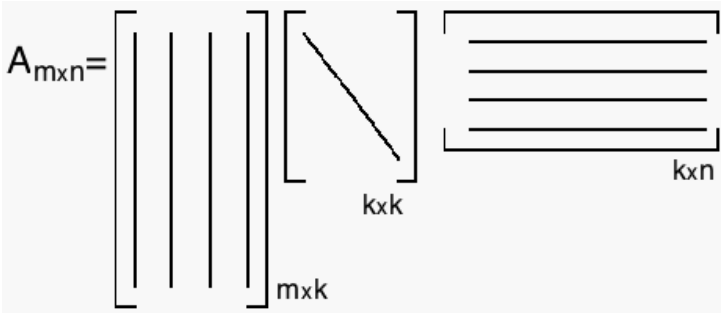


Figure B.2 The matrices which are involved in the SVD with the corresponding dimensions

Reduced Singular Value Decomposition is a technique which is used to retrieve information and determine similarities between different parameters in the matrix. One application of this reduced SVD is to analyze large quantities of data faster than it would take the full SVD to compute. One example of this is finding of similarities between articles. The reduced SVD functions by reducing the dimensions of the data. Assuming that the SVD matrix has the format: $word \times document$, it is possible to approximate the data using a reduced SVD matrix. An added benefit of using this process is that it is capable of making items that share a substructure appear more similar, while at the same time making features that are dissimilar appear even more dissimilar. This is a process which helps the classification process by sorting features of the matrix.

In a special case, where A is an $m \times m$ real square matrix with a positive determinant, all other matrices are also real $m \times m$ matrices. Σ is a scaling matrix, while U and V^T can be seen as rotation matrices. Thus, the equation given above can be seen as the multiplication between a rotation, a scaling matrix and another rotation. This effect is seen in Figure B.1, where it is clearly visible that the first matrix rotates the circle, while the second matrix scales it. The last matrix further rotates the scaled circle which has now become an ellipse.

APPENDIX C: DELAUNAY TRIANGULATION

In computing and mathematics the Delaunay Triangulation a kind of triangulation of a set of points in an k-dimensional space. Although the Delaunay Triangulation can be applied to any number of dimensions, it is most often used for points in two or three dimensional space. This is because these spaces have the most practical applications in the real world.

This triangulation is named after Boris Delaunay for his work in the field which started in 1934. He was a Russian mathematician who worked extensively on the geometry of numbers and modern algebra.(Shafarevich, 1961)

The main condition for the Delaunay triangulation is that after triangulation, there is no point in the triangles formed by the process. There are certain special cases for this type of triangulation. For example, a set of points on a line cannot form a Delaunay Triangulation. Moreover, four points which are equally spaced from each other can be connected in any one of two ways as both of them are solutions for the Delaunay triangulation. This example can be seen below in Figure C.1.

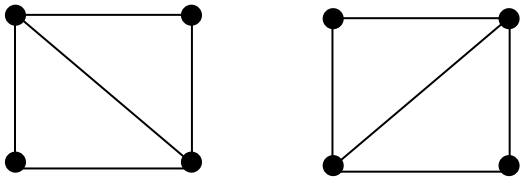


Figure C.1 two possible solutions to the Delaunay triangulation of four points which are equally spaced from each other.

The Delaunay triangulation is also very closely related to the Voronoi Diagram. The Voronoi diagram is a way of distributing space around points. This is done by computing the space around the points and dividing it evenly between different pairs of points such that each point is located at the center of a region which is surrounded by other regions that also contain points at the center. (Science, 1980)

The Voronoi diagram can be obtained from the Delaunay triangulation by connecting the centers of the circumcircles located in the Delaunay triangulation. This process is seen in Figure C.2.

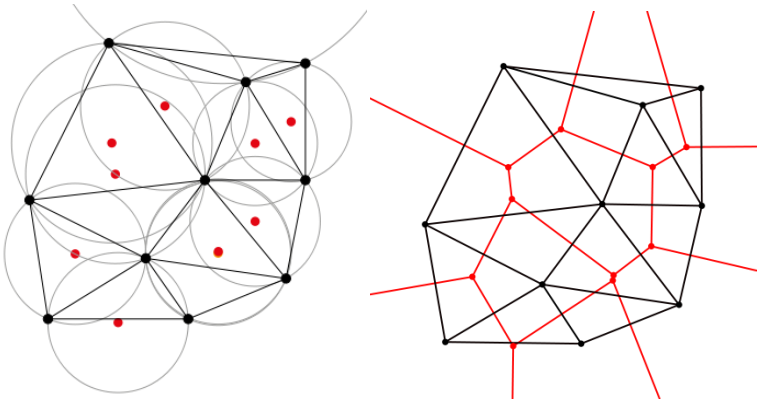


Figure C.2 The circumcircles of the Delaunay triangulation are located on the left image together with the Delaunay triangulation. On the right, the Voronoi diagram is seen drawn in red, and the positions of the intersections are the same as the position of the centers of the circles from the left.(Oudot, 2002)

The Delaunay triangulation is used for many different purposes in algorithms and computations. One of the most obvious uses for it is to reconstruct a mesh from a point cloud. In order to render a 3D model, the model needs to have surfaces, however, sometimes, 3D models are not available, and instead only point clouds are available.(Delaunay, 1993)

If the user wants to be able to display the point clouds with surfaces on them, then the Delaunay triangulation can be used to rebuild the model by creating edges between the different points of the point cloud. These edges can then be filled with a surface to create the 3D model of a point cloud.

Another use for the Delaunay triangulation is to find the nearest neighbor of a point. This is because, once a "map" of the points has been made through the Delaunay triangulation, it is very easy to find which point is close to which other points by searching directly in the triangulated mesh. This method is much faster than comparing all the distances for all the points, however, the Delaunay triangulation has been found to be slower than the kd tree method.