Università degli Studi di Napoli "Federico II"

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE

TESI DI LAUREA IN INGEGNERIA AEROSPAZIALE

CLASSE DELLE LAUREE MAGISTRALI IN INGEGNERIA AEROSPAZIALE E ASTRONAUTICA (LM 20)

LiDAR-based Global Registration Algorithms for Pose Acquisition of Non-cooperative Spacecraft

Relatori:
Prof. Michele Grassi
Prof. Roberto Opromolla

Candidato:
Clemente Tecchia
Matr. M53001593

Correlatori:
Dr. Alessia Nocerino
Dr. Margherita Piccinin
Dr. Ulrich Hillenbrand

ANNO ACCADEMICO 2023 – 2024

Table of Contents

Table of Contents **3**

Abstract **6**

List of Figures **7**

List of Tables **10**

Acronyms **11**

1. Introduction **13**

1.1. The Pose Estimation Problem 16

1.2. LiDAR Sensors 17

1.2.1. Scanning LiDARs 19

1.2.2. Detector Array LiDARs 20

1.2.3. Spatial Light Modulators 21

2. State of the Art **22**

2.1. Feature-Based Methods 24

2.1.1. Local-Feature-Based Methods 25

2.1.1.1. Local Feature Extraction/Description 25

2.1.1.2. Point Cloud Alignment 32

2.1.2. Global-Feature-Based Methods 37

2.1.2.1. Global Feature Extraction/Description 38

2.1.2.2. Recognition 47

2.1.3. Literature Analysis Conclusion 52

3. Methodologies **56**

3.1. Reference Geometry 57

3.2. Feature Extraction Analysis 58

3.2.1. Normal Estimation Analysis 58

3.2.2. FPFH Analysis 63
3.2.2.1. FPFH Distance Histogram 63
3.2.2.2. Persistence Analysis 65
3.2.2.3. Geometric Primitives Recognition 67
3.3. Hash Tables 75
3.3.1. Training Hash Tables 77
3.3.1.1. Point Cloud Classifier 77
3.3.1.2. Hash Table Construction for CS Scans in Label-Based RANSAC 79
3.3.1.3. Hash Table Construction for CS Scans in PA-Based RANSAC 81
3.3.1.4. Hash Table Construction for CS Scans in PPF-Based RANSAC 81
3.3.1.5. Hash Table Construction for FS Scans 84
3.3.2. Model Hash Tables 84
3.4. Overview of the Offline Phase of the Algorithms 85
3.4.1. Label-Based RANSAC 85
3.4.2. PA-Based RANSAC 86
3.4.3. PPF-Based RANSAC 87
3.5. Initial Pose Determination 87
3.5.1. Label-Based RANSAC 87
3.5.2. PA-Based RANSAC 88
3.5.3. PPF-Based RANSAC 89
3.5.4. Hash Table Lookup 90
3.5.4.1. Alignment Algorithms 91
3.5.4.2. Alignment Evaluation 94
3.6. Post-Processing 96
3.6.1. Iterative Closest Point 97
3.6.2. Ambiguity Reduction Process 98

4. Open3D Global Registration Algorithms 102
4.1. FPFH-Based RANSAC 102
4.2. Fast Global Registration 104
5. Experiments 105
5.1. Evaluation Metrics 105
5.2. Performance Analysis 106
5.2.1. Comparison of the Main Algorithms 106
5.2.1.1. Tuning Parameters 107
5.2.1.2. Comparison Results 110
5.2.2. Comparison of Algorithm Variants 113
5.2.2.1. Training Hash Table vs Model Hash Table 113
5.2.2.2. Nearest Neighbor vs Binary Matching 114
5.2.3. Performance of Open3D Algorithms without and with Downsampling 116
5.3. Autonomous Failure Detection 117
6. Conclusions and Future Works 121
References 124
Acknowledgements 131
Ringraziamenti 134

Abstract

This thesis work, developed in collaboration with the German Aerospace Center (DLR), is placed in the context of spacecraft pose determination, i.e., the problem of calculating the set of parameters that describe the relative position and attitude of an active satellite with respect to another space object, which is widely encountered in space missions such as On-Orbit Servicing (OOS) and Active Debris Removal (ADR), where algorithmic and technological solutions are essential to ensure the efficient execution of autonomous maneuvers of a chaser in close-proximity with respect to a designated target.

Specifically, the work carried out addresses the problem of pose acquisition of a known non-cooperative spacecraft, based on the use of target 3D point cloud scans produced by a LiDAR sensor, proposing a suite of feature-based algorithmic solutions, developed in Python environment, that leverage point-normal structures as local features (Fast Point Feature Histograms, FPFH) or as non-local primitives (Point Pair Features, PPF). Additionally, they exploit a Random-Sample-Consensus-based (RANSAC-based) strategy to perform the initial pose estimation and Hash Tables (HT) for fast and efficient matching.

The performance of the proposed architecture is tested using a dataset of synthetic point clouds obtained using a LiDAR data simulator developed by DLR and considering as target the Client Satellite of the DLR On-Orbit Servicing Simulator for Capture (OOS-SIM). The achieved performance is compared against standard approaches in 3D registration, namely FPFH-based RANSAC and Fast Global Registration (FGR), implemented in the Python Open3D Library. The obtained results demonstrate that these algorithms are promising alternatives to standard approaches, showing comparable accuracy, but with a slight disadvantage in computational time.

Finally, a description of an autonomous failure detection strategy is provided, which can be applied to increase robustness of the proposed pose estimation architectures.

List of Figures

Figure 1.1 - Reconstructed number of catalogued objects in Earth's orbit [1] 13
Figure 1.2 - Close-proximity scenario example [2] 14
Figure 1.3 - Different concepts of on-orbit servicing missions: (a) ETS-VII of JAXA, (b) orbital express of DARPA, (c) TECSAS of DLR/CSA/RKA and (d) DEOS of DLR [2] 15
Figure 1.4 - Taxonomy of LiDARs [4] 18
Figure 1.5 - Basic triangulation geometry [8] 19
Figure 1.6 - Simplified representation of a scanning LiDAR [9] 20
Figure 1.7 - Simplified representation of a detector array LiDAR [9] 21
Figure 2.1 - Logical scheme of the pose determination process [10] 22
Figure 2.2 - Classification of point cloud registration methods. Highlighted in blue is the method class of interest 23
Figure 2.3 - Local-feature-based methods pipeline 25
Figure 2.4 - Geometric Hashing pre-processing step [31] 26
Figure 2.5 - Local features exploited by PAH architecture [18] 27
Figure 2.6 - CTA pipeline [19] 28
Figure 2.7 - The influence region diagram for a Point Feature Histogram. In red: query point. In blue: k – neighbors [21] 30
Figure 2.8 - The influence region diagram for a Fast Point Feature Histogram [22] 31
Figure 2.9 - Illustration with 2D point sets. In blue: genuine correspondences; in red: spurious correspondences [34] 34
Figure 2.10 - Geometric Hashing recognition step. The diagram represents the continuation of Figure 2.4 [31] 36
Figure 2.11 - Global-feature-based methods pipeline 37
Figure 2.12 - Pipeline of silhouette image generation [27] 40
Figure 2.13 - Left: point cloud of a wine glass (black) with associated cluster $\boldsymbol{Ci}$ (green) and the SGURF reference frame. Right: the resulting OUR-CVFH histogram [24] 44
Figure 2.14 - Basis Point Set encoding for point clouds [17] 45
Figure 2.15 - Flow diagram of the Online TM algorithm [10] 48
Figure 3.1 - CAD model of the OOS-SIM Client Satellite and TRF representation 57

Figure 3.2 - Graphical visualization of normal correction using sensor viewpoint [49] 59
Figure 3.3 - Mean error plots for Scan 1 by varying r and $maxnn$ 60
Figure 3.4 - Scan 1 colored point cloud 61
Figure 3.5 - Mean error plots for Scan 2 by varying r and $maxnn$ 61
Figure 3.6 - Mean error plots for Scan 3 by varying r and $maxnn$ 61
Figure 3.7 - Normal estimation time performance for Scan 1 and Scan 2 62
Figure 3.8 - FPFH distance histograms for Scan 1 64
Figure 3.9 - Persistent points for model point cloud (left), Scan 1 (central) and Scan 2 (right) 66
Figure 3.10 - Persistent FPFH distance histograms for Scan 1 and Scan 2 66
Figure 3.11 - Segmented point cloud 68
Figure 3.12 - Comparison mean FPFH toroid and edges 69
Figure 3.13 - Comparison mean FPFH sphere and handles 69
Figure 3.14 - Comparison mean FPFH cylinder and planes 69
Figure 3.15 - Example of cylinder-points detection and terminology used 71
Figure 3.16 - Precision-Recall curves for cylinder geometry: cases 1 and 2 72
Figure 3.17 - Precision-Recall curves for edges geometry: cases 1 and 2 73
Figure 3.18 - Precision-Recall curves for handles geometry: cases 1 and 2 73
Figure 3.19 - Precision-Recall curves comparison between the geometries analyzed 74
Figure 3.20 - Working principle of a hash table 75
Figure 3.21 - Collision handling 76
Figure 3.22 - Two examples of CS scan normal distribution 78
Figure 3.23 - Two examples of FS scan normal distribution 78
Figure 3.24 - Graphical representation of a surflet and the defined LRF 82
Figure 3.25 - Label-based RANSAC offline block diagram 85
Figure 3.26 - PA-based RANSAC offline block diagram 86
Figure 3.27 - PPF-based RANSAC offline block diagram 87
Figure 3.28 - Label-based RANSAC online block diagram 88
Figure 3.29 - PA-based RANSAC online block diagram 89
Figure 3.30 - PPF-based RANSAC online block diagram 89
Figure 3.31 – HT lookup. This diagram is representative of both the CS and FS case 91
Figure 3.32 – Post-processing 97

Figure 3.33 - a) Case of correctly estimated pose: $\boldsymbol{T}ICP = \boldsymbol{T}SRF \rightarrow TRF$. b) Case of incorrect pose that has passed the HT lookup phase: $\boldsymbol{T}ICP = \boldsymbol{T}SRF \rightarrow FTRF$ 99
Figure 3.34 - Flips implemented. a) Flip check performed for all scans. b) FS Flip check: upper plane; c) FS Flip check: lower plane 100
Figure 5.1 - Comparison results between the algorithms. The metrics compared are ADD, ADI (top 4 plots), rotational error and translational error (bottom 4 plots), without post-processing (left column) and with post-processing (right column) 111
Figure 5.2 - Comparison of Label and PA-based RANSAC using training and model HT approaches, with post-processing. Left: ADD, ADI. Right: rotational and translational errors 113
Figure 5.3 - Comparison of Label and PA-based RANSAC using NN and BM approaches, without post-processing. Left: ADD, ADI. Right: rotational and translational errors 115
Figure 5.4 – Comparison of Open3D algorithms with downsampling (downs.) and without downsampling (no downs.), with post-processing. Left: ADD, ADI. Right: rotational and translational errors 116
Figure 5.5 – AFD results. Highlighted are the ranges of $f\tau$ such that the probability PT that the AFD algorithm tells the truth is maximum 119

List of Tables

Table 2.1 - Feature Extraction/Description pros and cons 53
Table 2.2 - Alignment/Recognition pros and cons 55
Table 3.1 - Parameters selection for normal estimation 63
Table 3.2 - Normal and FPFH estimation parameters for both LiDAR and model point clouds 66
Table 3.3 - Precision-Recall analyses: Cases 1 and 2 parameter setting 72
Table 3.4 - Filtering parameters resulting from Precision-Recall analysis 74
Table 3.5 - Sum of variances of normal components of point clouds shown as an example 79
Table 3.6 – Flip matrices $\boldsymbol{T}FTRF \rightarrow TRF$ defined 101
Table 5.1 - Tuning parameters selected for training HTs construction 107
Table 5.2 - HT-based normal, FPFH estimation and feature extraction tuning parameters 108
Table 5.3 - FPFH-based RANSAC and FGR normal and FPFH estimation tuning parameters 108
Table 5.4 – Alignment evaluation tuning parameters adopted 109
Table 5.5 - ICP tuning parameters adopted 109
Table 5.6 - Time performance 112
Table 5.7 – Runtimes of Label-based and PA-based RANSAC, with training and model HTs 114
Table 5.8 - Parameters selected for $\boldsymbol{HT}voxel$ 114
Table 5.9 – Runtimes of Label-based and PA-based RANSAC, with NN and BM approaches 115
Table 5.10 – Runtimes of Open3D algorithms, without and with downsampling 117
Table 5.11 – Optimal $f\tau$ values resulting from AFD analysis 120

Acronyms

ADD = Average Distance of model points for Distinguishable points

ADI = Average Distance of model points for Indistinguishable points

ADR = Active Debris Removal

AFD = Autonomous Failure Detection

BM = Binary Matching

BPS = Basis Point Set

CPD = Coherent Point Drift

CRH = Camera Roll Histogram

CS = Complex Structure

CSA = Canadian Space Agency

CTA = Congruent Tetrahedron Align

CVFH = Clustered Viewpoint Feature Histogram

CW = Continuous Wave

DARPA = Defense Advanced Research Projects Agency

DEOS = Deutsche Orbital Servicing Mission

DLR = Deutsches Zentrum für Luft- und Raumfahrt (German Aerospace Center)

DOF = Degree(s) Of Freedom

EO = Electro-Optical

ETS-VII = Experimental Test Satellite VII

FGR = Fast Global Registration

FOV = Field Of View

FPFH = Fast Point Feature Histograms

FS = Flat Structure

FTRF = Flipped Target Reference Frame

GH = Geometric Hashing

HT = Hash Table

ICP = Iterative Closest Point

ICS = Image Coordinate System

JAXA = Japan Aerospace Exploration Agency

KD = k-Dimensional

KL = Kullback-Leibler

LaDAR = Laser Detection and Ranging

LCS = LiDAR Coordinate System

LEO = Low Earth Orbit

LiDAR = Light Detection and Ranging

NDT = Normal Distributions Transform

NN = Nearest Neighbor

OOS = On-Orbit Servicing

OOS-SIM = On-Orbit Servicing Simulator for Capture

OUR-CVFH = Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram

PA = Persistence Analysis

PAH = Polygonal Aspect Hashing

PCA = Principal Component Analysis

PFH = Point Feature Histograms

RANSAC = Random Sample Consensus

RKA = Russian Space Agency

RMSE = Root Mean Square Error

SAC-IA = Sample Consensus Initial Alignment

SGURF = Semi-Global Unique Reference Frame

SLM = Spatial Light Modulator

SPFH = Simplified Point Feature Histogram

SR = Success Rate

SRF = Sensor Reference Frame

SVD = Singular Value Decomposition

TCS = Target Coordinate System

TECSAS = Technology Satellites for Demonstration and Verification of Space Systems

TM = Template Matching

TOF = Time Of Flight

TRF = Target Reference Frame

VFH = Viewpoint Feature Histogram

1. Introduction

The rapid expansion of space activities has resulted in a significant increase in the number of operational and defunct satellites, as well as orbital debris, in Earth's orbit. This growing population of space objects presents both opportunities and challenges. On one hand, the increase in operational satellites supports a wide range of applications, from communication and navigation to Earth observation. On the other hand, the accumulation of defunct satellites and debris poses risks to space missions, heightening the likelihood of collisions and the potential for a cascade effect known as the Kessler Syndrome. Figure 1.1 illustrates the reconstructed growth rate of catalogued objects in Earth's orbit [1].

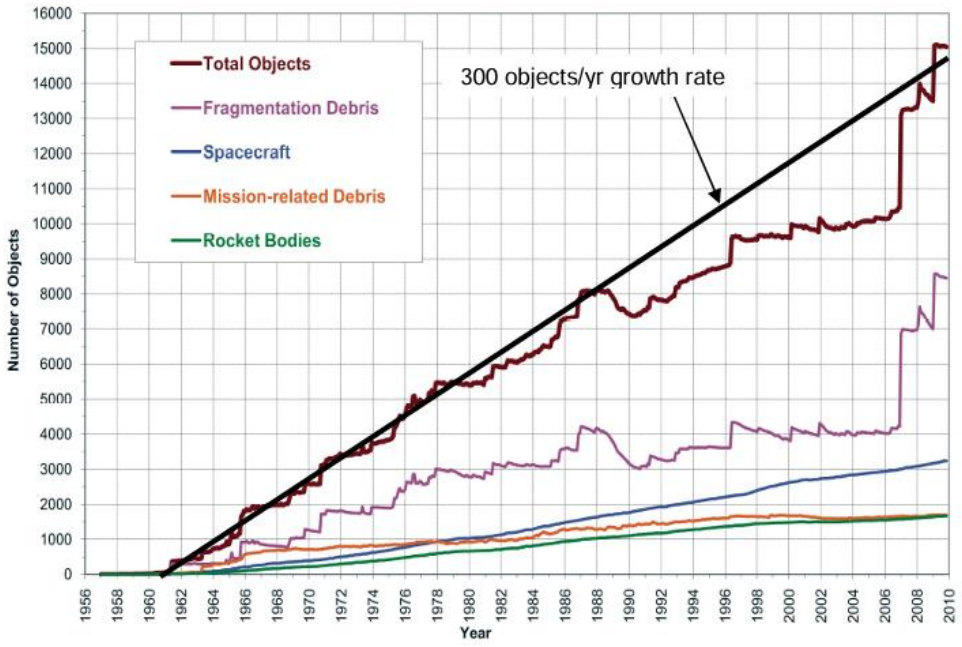

Figure 1.1 - Reconstructed number of catalogued objects in Earth's orbit [1]

This proliferation of space objects has led to an ever-increasing interest in developing strategies to manage both operational and non-operational assets in orbit. These strategies encompass On-Orbit Servicing (OOS) [2], which focuses on satellite maintenance and lifespan extension, and Active Debris Removal (ADR) [3], which targets the removal of defunct objects. Central to these scenarios is the implementation of autonomous maneuvers of an active satellite, called *chaser*, in close-proximity to a designated *target*.

Figure 1.2 shows an OOS scenario in which the chaser, equipped with a robotic arm, approaches the target for maintenance operations.

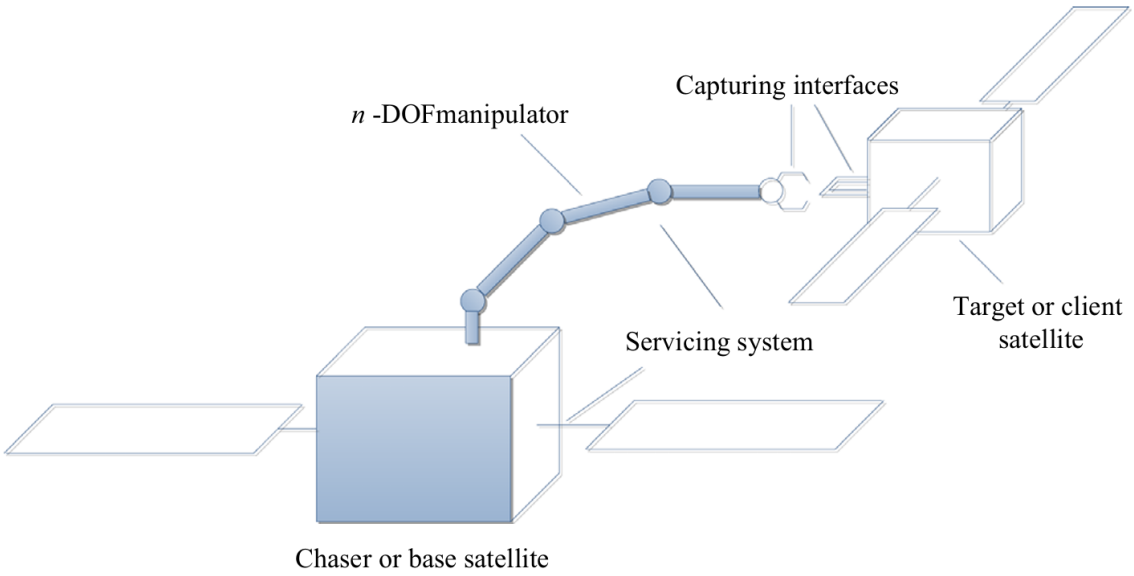

Figure 1.2 - Close-proximity scenario example [2]

Figure 1.3, instead, illustrates some real-world OOS mission concepts, which are briefly described below: (a) the Experimental Test Satellite VII (ETS-VII) of the Japan Aerospace Exploration Agency (JAXA), considered the first robotic OOS demonstration mission. This mission included robotic servicing tasks such as orbital replacement unit exchange, capture and berthing of a target satellite; (b) the Orbital Express mission of the Defense Advanced Research Projects Agency (DARPA), a demonstration mission in which autonomous rendezvous and docking, as well as in-orbit refueling operations, were performed. Additionally, during the mission, a robotic arm autonomously transferred a supplemental battery and backup computer to the target spacecraft; (c) the Technology Satellites for Demonstration and Verification of Space Systems (TECSAS) mission, developed by the German Aerospace Center (DLR), the Canadian Space Agency (CSA), and the Russian Space Agency (RKA). This mission included rendezvous, close approach, flying-around inspection, formation flight, capture and manipulation of the target satellite; (d) the Deutsche Orbital Servicing Mission (DEOS) of the DLR, which aimed to develop and evaluate procedures and techniques for rendezvous, capture, and deorbiting of a noncooperative spacecraft from its operational orbit.

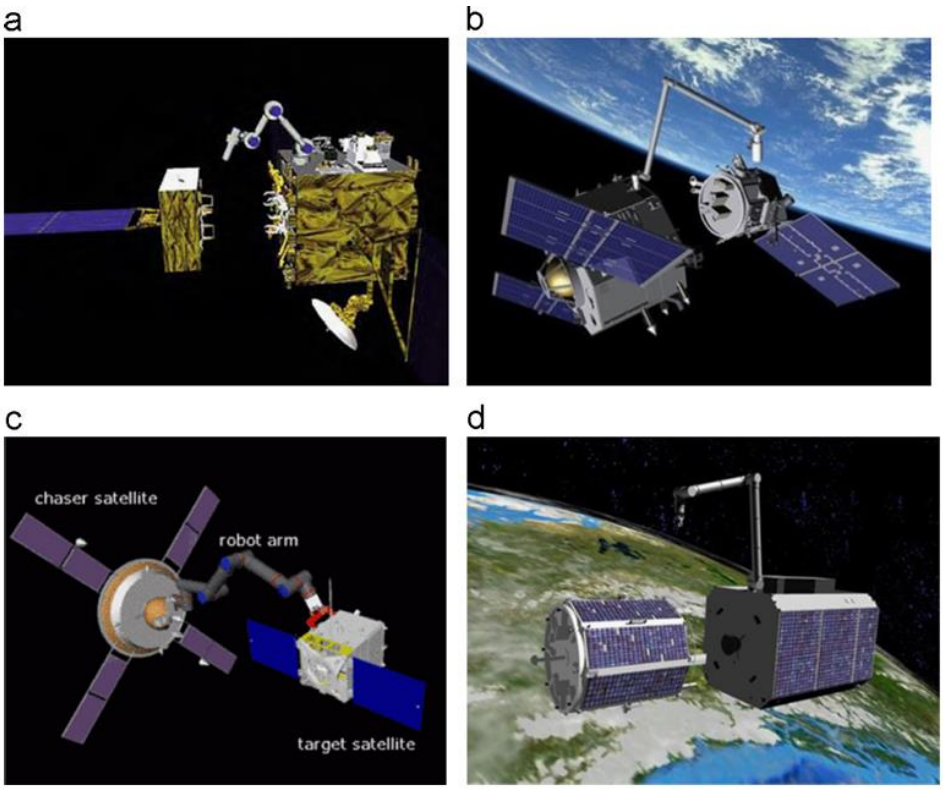

Figure 1.3 - Different concepts of on-orbit servicing missions: (a) ETS-VII of JAXA, (b) orbital express of DARPA, (c) TECSAS of DLR/CSA/RKA and (d) DEOS of DLR [2]

To properly perform these tasks, accurate knowledge of the relative position and attitude between the two space objects is essential: this problem is called *spacecraft pose estimation*.

The problem of pose estimation of space targets is approached differently depending on:

- *The type of target (cooperative or uncooperative)*. A target is cooperative if it is designed to provide information that simplifies the estimation of its pose with respect to the chaser. Examples are targets equipped with a dedicated communication link or with easily recognizable markers - typical in both Formation Flying (FF) and OOS applications. More challenging, however, is the estimation of the pose of uncooperative targets, which are not able to communicate with the chaser and are not equipped with markers. A further distinction can be made between uncooperative targets whose geometry is known (case that can occur in OOS and ADR scenarios) and targets of partially or totally unknown shape (case of some ADR missions as well as asteroid exploration scenarios, for example) [4];

- *The type of Electro-Optical (EO) sensor mounted on the chaser*. EO sensors represent the best option to ensure pose determination in close-proximity; many research efforts are currently dedicated to the design and development of vision-based pose estimation techniques relying on either monocular or stereo cameras [5, 6], in light of the relatively small size, weight, and low power consumption of passive imaging sensors. Nevertheless, active Light Detection and Ranging (LiDAR) systems are preferred as main relative navigation sensors, as demonstrated by the recent success of the Mission Extension Vehicle missions [7], given their advantages in terms of operative range, direct depth observability and robustness against unfavorable illumination conditions.

Given the relevance of this topic, it has been and still is the object of study by various universities and research centers, which over the years have developed ad-hoc techniques and algorithms for solving this problem in the various conditions previously mentioned.

In this context, this thesis work is placed, which focuses on the problem of LiDAR-based pose acquisition of non-cooperative spacecraft of known geometry. Specifically, this work presents novel feature-based pose estimation algorithms, whose performance is compared with well-known strategies in 3D registration.

In the next sections, the points briefly mentioned in this introduction are explored further, specifically concerning the problem of pose estimation and the characteristics of a LiDAR sensor.

1.1. The Pose Estimation Problem

The pose represents the set of parameters that characterize the translation and rotation between two reference systems. Therefore, given a Sensor Reference Frame (SRF), centered in the chaser, and a Target Reference Frame (TRF), centered in the target, the pose allows to define the transformation that aligns the two reference systems.

The SRF to TRF pose matrix is defined as a $4x4$ matrix that includes a $3x3$ rotation matrix $\boldsymbol{R}$ and a translation vector $\boldsymbol{t}$, where $\boldsymbol{R}$ aligns the TRF to the SRF while $\boldsymbol{t}$ represents the position of

the TRF origin with respect to the SRF origin, expressed in SRF. The pose matrix is defined by the Equation 1.1:

$$\boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix} \tag{1.1}$$

The determination of this transformation matrix is of great importance in the field of robotics in general, and is in fact widely studied to promote the implementation of close-proximity operations between a service satellite and a target, with ever-increasing precision, as already stated in the introduction to Chapter 1.

Pose estimation is performed thanks to the extraction of information from the observed scene, by a sensor, and the comparison of this information with a model. Clearly, the problem is approached differently depending on the sensor used. In Section 1.2, the characteristics of the EO Sensors investigated in this thesis work, i.e. LiDARs, are briefly described.

1.2. LiDAR Sensors

A LiDAR (Light Detection And Ranging), also called LaDAR (Laser Detection and Ranging) is an active 3D EO sensor. A sensor is called *3D* when it is able to produce a three dimensional representation of the observed scene, while it is called *active* when the source of the detected radiation is internal to the system. The main components of a LiDAR are: (1) the laser source, from which the radiation is emitted; (2) the optics, which is fundamental for determining the minimum range from which the sensor is able to measure the distance from objects observed; (3) the detector, which captures the radiation as it comes back.

LiDAR sensors use light (typically a laser) to illuminate the target and measure the time it takes for the emitted signal to return to the sensor. Since the light must travel from the source to the target, and back to the detector, the range R to the observed point may be computed as shown in Equation 1.2:

$$R = \frac{ct}{2} \tag{1.2}$$

where R is the distance from the sensor to the point on the target, c is the speed of light and t is the laser Time-Of-Flight (TOF). Hence, this sensor is able to provide multiple 3D vectors within its field of view. The set of these 3D position vectors can be interpreted as a *point cloud*. The above is just one example of how a LiDAR can extract information from the observed scene; in fact, those that use the principle just described are called TOF-based LiDARs.

There are a multitude of types of LiDAR, which can be classified according to: (1) The characteristics of the emitted laser beam; (2) The measurement principle; (3) The technological solution. A schematic of the above is shown in Figure 1.4.

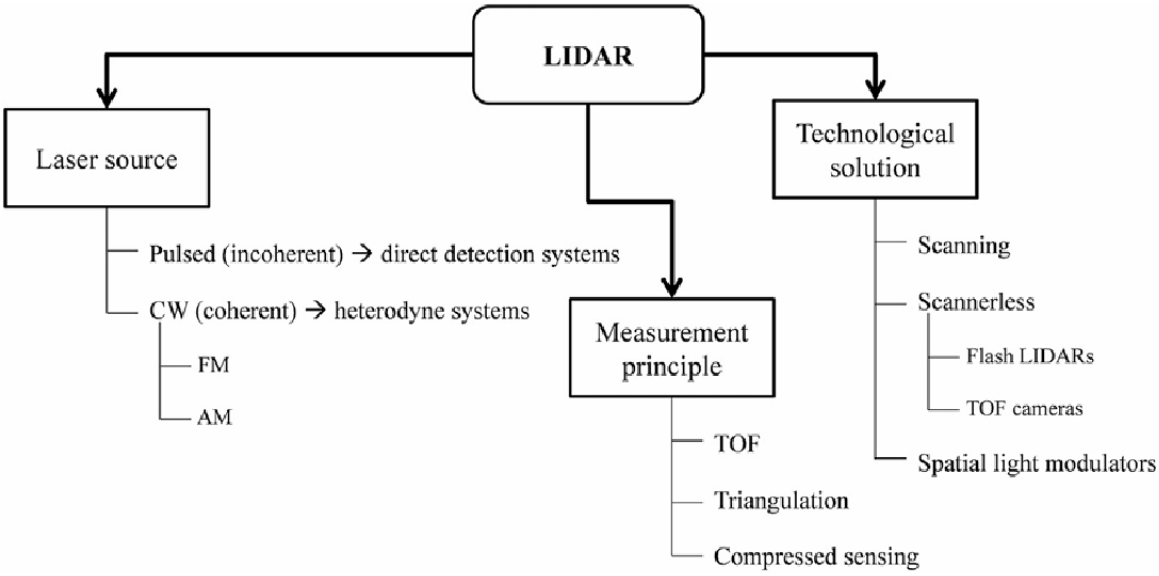

Figure 1.4 - Taxonomy of LiDARs [4]

As can be seen from Figure 1.4, regarding the laser source, LiDARs can be classified into Pulsed and Continuous Wave (CW) systems.

Pulsed LiDARs emit energy in discrete intervals (pulses), are less expensive than CW LiDARs and are typically used when long operating ranges are desired. They measure distance by measuring the time delay between transmitted and received laser pulses. These systems represent the already mentioned TOF-based LiDARs.

On the other hand, *CW LiDARs* emit energy continuously and are mainly classified according to the modulation type Amplitude-Modulated (AM) or Frequency-Modulated (FM). They

measure distance by measuring the phase difference between the emitted signal and the reflected echo.

CW LiDARs can also exploit the triangulation method as measurement principle. Basically, the same principle is adopted as for a stereo system, positioning the laser source with a certain offset with respect to the detector and therefore to the relative optics. The laser is projected onto a surface and the location of the reflected image on an offset detector depends on the range to the surface at that projected location, hence measurement of the image position indicates range [8]. In Figure 1.5, the basic triangulation geometry is shown.

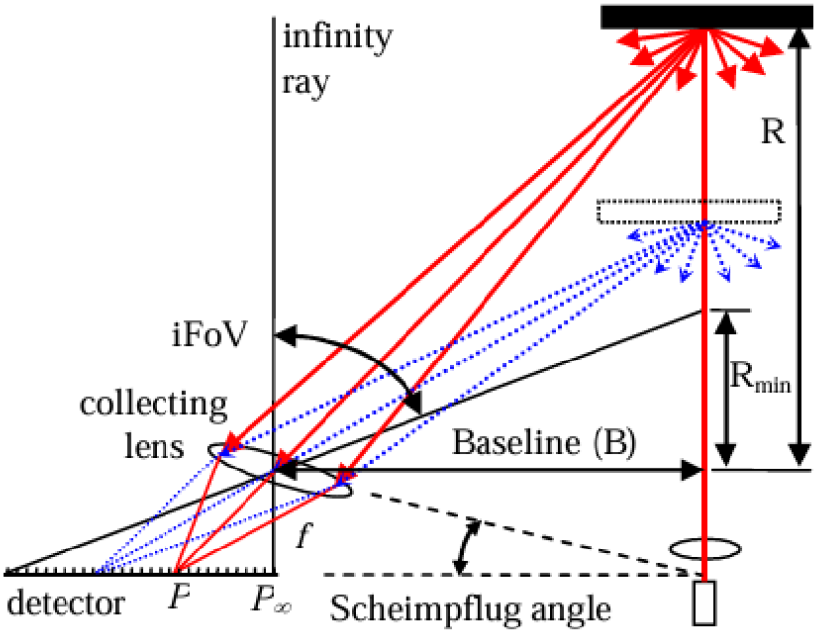

Figure 1.5 - Basic triangulation geometry [8]

Regarding the technological solutions, instead, LiDARs can be classified as: (1) Scanning LiDARs; (2) Detector Array LiDARs; (3) Spatial Light Modulator LiDARs.

1.2.1. Scanning LiDARs

Scanning LiDARs use a narrow laser beam that is swept over the sensor's Field Of View (FOV) according to a pre-established pattern to obtain range measurements to objects within the scene. The direction of the laser is changed using lenses, mirrors, or other devices. A simplified representation is shown in Figure 1.6.

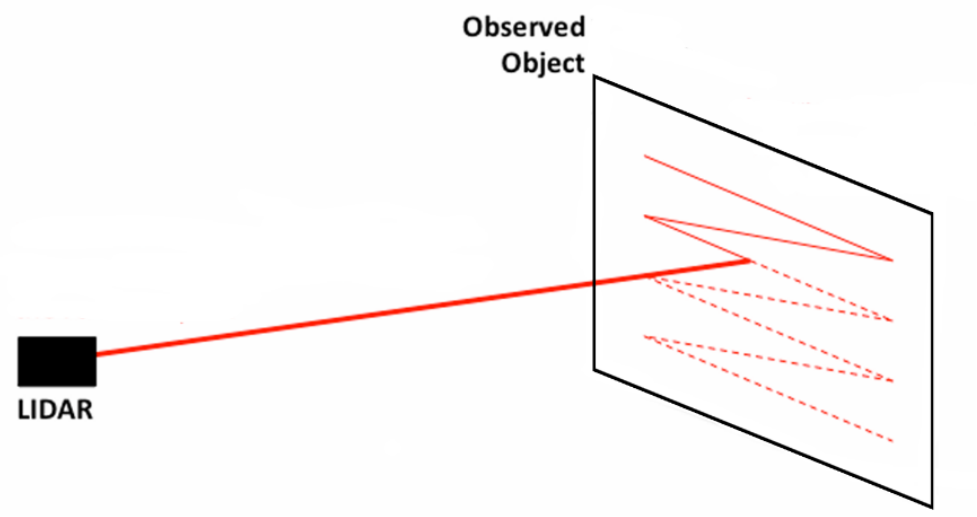

Figure 1.6 - Simplified representation of a scanning LiDAR [9]

There are several scanning patterns; Figure 1.6 shows the Raster pattern as an example, but there are also more complex ones (e.g. Lissajous pattern).

Since they only use one detector (or a very small number of detectors), these sensors are relatively easy to calibrate; in addition, scanning LiDARs can point the narrow laser beam very precisely and create very high-resolution point clouds. On the other hand, these devices contain moving parts that can potentially be a source of hardware failure; furthermore, these sensors are expensive, sensitive to motion blur and it is necessary to wait a certain amount of time for the entire scene of interest to be scanned.

1.2.2. Detector Array LiDARs

Detector Array LiDARs (or scannerless LiDARs) illuminate the entire scene with a single broad laser beam and use a detector array to detect the echoes backscattered in the pixel direction [4]. A simplified representation is shown in Figure 1.7.

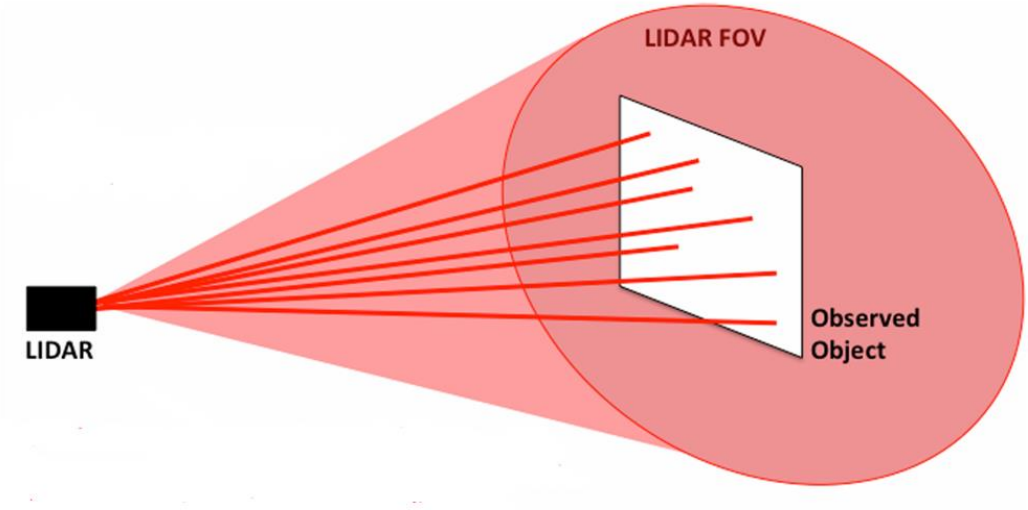

Figure 1.7 - Simplified representation of a detector array LiDAR [9]

Unlike scanning LiDARs, Detector Array LiDARs have no moving parts, are less susceptible to motion blur, and are cheaper. On the other hand, having multiple detectors makes them more difficult to calibrate and does not reach the accuracy of scanning LiDARs.

1.2.3. Spatial Light Modulators

Finally, Spatial Light Modulators (SLMs) illuminate the observed scene with a light pattern and detect the reflected signal through compressed sensing algorithms capable of reconstructing the scene observed from the time history of the reflected signal.

SLMs have the advantage of having no moving parts and a single detector. On the other hand, they require assumptions about the geometry of the observed scene; furthermore, they are still a developing technology [9].

2. State of the Art

In the literature there are different methods to face the challenge presented in Chapter 1; these pose acquisition methods from 3D point clouds fall into the broader field of Point Cloud Registration. To accurately follow the evolution of the relative pose of an object, two main steps are carried out:

- *Pose Initialization*. An initial estimate of the pose of the spacecraft is made, using the first set of data acquired by the sensor, without knowing any a priori information on its position and attitude. Pose initialization methods are also called *global methods* and are typically used to provide input information to pose tracking methods;
- *Pose Tracking*. An update of the pose parameters is carried out, using as input the pose information obtained thanks to global methods, in order to obtain a more accurate estimate at the output. Pose tracking methods are also called *local methods*.

A diagram of the pose determination process is shown in Figure 2.1.

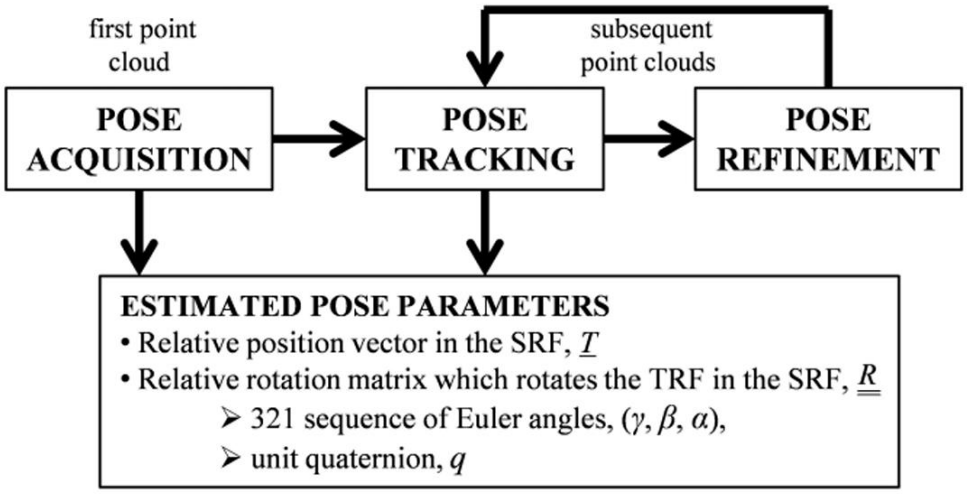

Figure 2.1 - Logical scheme of the pose determination process [10]

Point cloud registration methods are very varied and it is not trivial to find a way to effectively classify the types of existing methods. In Figure 2.2 a very simple diagram is shown which highlights the class of methods explored in depth in this work; in particular, the focus is on *global methods*.

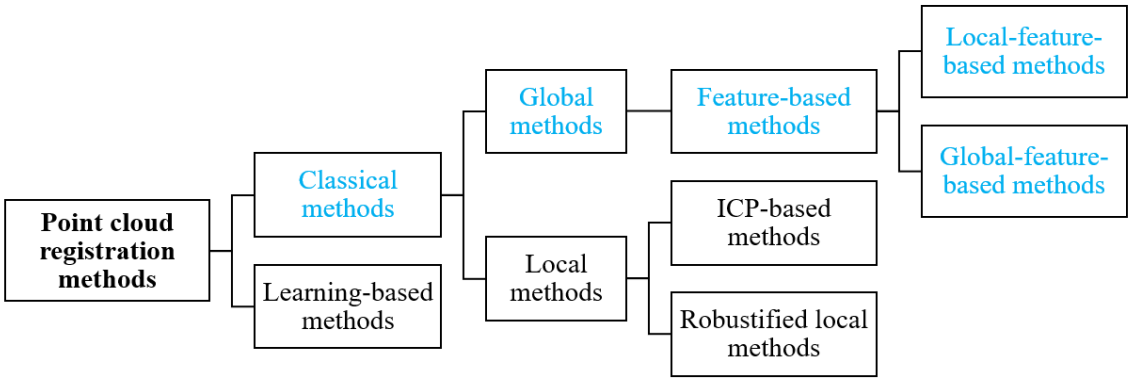

Figure 2.2 - Classification of point cloud registration methods. Highlighted in blue is the method class of interest

From Figure 2.2 it can be observed that a first major distinction is done between classical methods and learning-based methods. The substantial difference between these two classes of methods lies in the fact that the classical methods are generally based on developing an iterative optimization algorithm to estimate the rigid or, in general, non-rigid transformation, while the learning-based methods use machine learning and deep learning techniques to learn the transformation based on the geometric features of the point clouds. The learned descriptors may contain more detailed features than geometric characteristics or other classic descriptors [11, 12]. In this Chapter, only classical methods are covered.

In general, point cloud registration methods can be classified into coarse and fine approaches or, as shown in the diagram, *global* and *local methods*. The methods that are mainly used as global methods are the *feature-based methods*. These techniques are designed to extract distinctive and robust local geometrical characteristics and exploit them as correspondences to estimate a transformation between two point clouds. These extracted characteristics, called features, can be subdivided into *local features* and *global features*. Local ones are extracted from specific regions of interest of the point cloud, while global ones are generated by encoding the overall geometric properties of the entire point cloud.

Then, there are the *local methods*, which have the aim of refining the pose given as input by the global methods and making it as precise as possible. The most popular and widely used technique is the Iterative Closest Point (ICP), an algorithm which, given as input the two point clouds to be aligned and the initial guess, iteratively determines the correspondences between

them, attributes weights to them and performs outlier removal to handle noisy conditions and updates the transformation until convergence, set by a cost function to be minimized or by the maximum number of iterations. Numerous variants of this algorithm have been subsequently proposed to counteract the limitations affecting the general algorithm and improve its performance.

One of the main limitations of the ICP is that its performance strongly depends on the initial guess, i.e. how much it differs from the true pose of the target: in other words, the ICP is not very robust if the initial pose is poor. Some local methods overcome this limitation, demonstrating robustness even with poor initialization, and are classified as *robustified local methods* [13]. Common examples are probabilistic methods that represent data through specific probability density functions, i.e. Coherent Point Drift (CPD) [14] and Normal Distributions Transform (NDT) [15].

In the following paragraphs, some state-of-the-art feature-based methods are analyzed in detail, describing them by individually analyzing the logical blocks that constitute their pipelines, and finally highlighting their advantages and disadvantages.

Of the methods that are analyzed, only some have actually been used for space applications, while other algorithms have been initially developed more generally for object recognition purposes.

2.1. Feature-Based Methods

A *feature* is an individual and measurable property of an observed phenomenon. This is a discriminating characteristic with a high information content, which can be codified and converted into numerical form. In the case of interest, features can be used to extract information about the geometry of the point cloud under study, and they are a very useful tool for facilitating and speeding up classification processes, pattern recognition and so on. These features can be geometric primitives (such as points or lines) [16, 17], polygons [18], tetrahedrons [19], point-normal structures [20, 21, 22, 23, 24] or even the point cloud itself [10, 25, 26, 27, 28].

As anticipated, features can be grouped into two large categories: local and global ones: local features contain information about the local geometry of the point cloud, while global features are representative of the entire point cloud. This difference in approach has advantages and disadvantages. Local features are less discriminating than global ones, but they are more robust in occlusion and cluttered environment. On the other hand, global features are efficient in computation time and memory consumption. However, global-feature-based methods are affected generally by occlusion and clutter. In the next Subsections, both these classes are analyzed in more detail.

2.1.1. Local-Feature-Based Methods

In this Subsection, local-feature-based methods are illustrated, starting with a look at the pipeline these methods employ, shown in Figure 2.3. The objective, given two point clouds, i.e. the point cloud acquired by the LiDAR and a model point cloud, obtained for example from the known 3D CAD model of the target, is to align one to the other. Therefore, given the point clouds as input, the basic idea is to exploit the local features detected from both point clouds to generate sets of correspondences, which will be used for alignment.

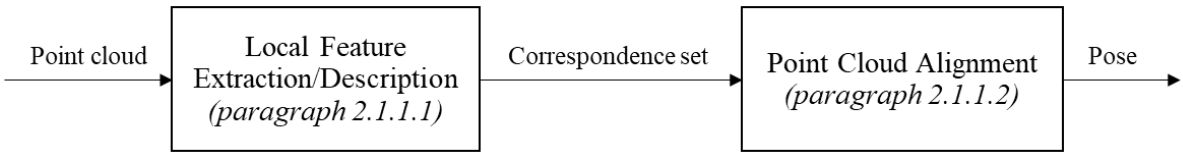

Figure 2.3 - Local-feature-based methods pipeline

Note that the first step is not just called *Feature Extraction* as some of the methods that are presented are point-based approaches that work directly with the raw dataset and do not require computationally expensive feature extraction steps [18, 19].

2.1.1.1. Local Feature Extraction/Description

The first step of a local-feature-based method is the extraction - not in all cases - and storage of the features. This paragraph illustrates the types of features and the methodologies adopted by some local feature methods to exploit them.

Geometric Hashing

Lamdan and Wolfson [16] proposed an indexing-based approach that exploits geometric primitives for model-based object recognition in occluded scenes and is Geometric Hashing (GH). These local features are encoded and stored in a database; the idea is to use it for storing pieces of information of known geometric objects in order to allow fast recognition of an unknown query object [29].

In a pre-processing step, n model's point features are extracted, with respect to a world coordinate system [30]. Then, using these points, a base is defined, made up of 2 points in the case of 2D objects, or 3 points in the case of 3D data (as in the case of interest) and a reference system is defined using this base, within which the coordinates of these point features are determined. Finally, this model information is stored in a large memory, a hash table. The contents of the hash table are independent of the scene and can thus be computed offline, not affecting the recognition time.

Figure 2.4 shows a summary diagram of the Geometric Hashing pre-processing phase. Note that this diagram refers to 2D objects, in the case of 3D data the hash table will also be three-dimensional, using the 3-point bases as indices.

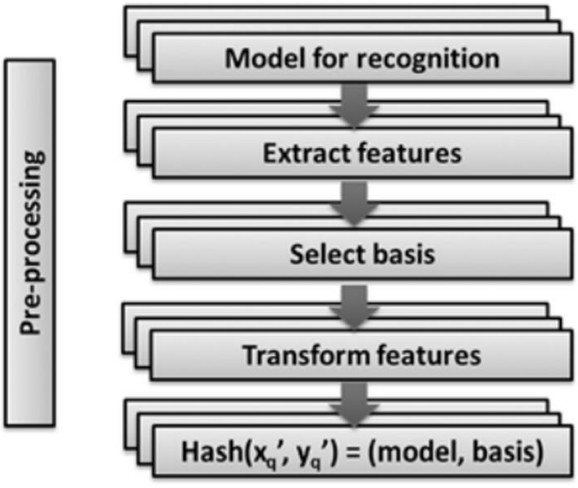

Figure 2.4 - Geometric Hashing pre-processing step [31]

Polygonal Aspect Hashing

In the wake of Geometric Hashing, a method for object localization was subsequently developed by Ruel et al. [18], specifically for point cloud processing: it is the Polygonal Aspect Hashing (PAH) and is a point-based method that works directly on the raw dataset and therefore is not based on computationally expensive extraction steps of features, an aspect that constitutes a great improvement compared to its predecessor. This method exploits one or more polygons selected from the input scan data as local features, as shown in Figure 2.5. The number of polygons selected for the subsequent matching phase constitutes a trade-off between computational cost and greater robustness to outliers.

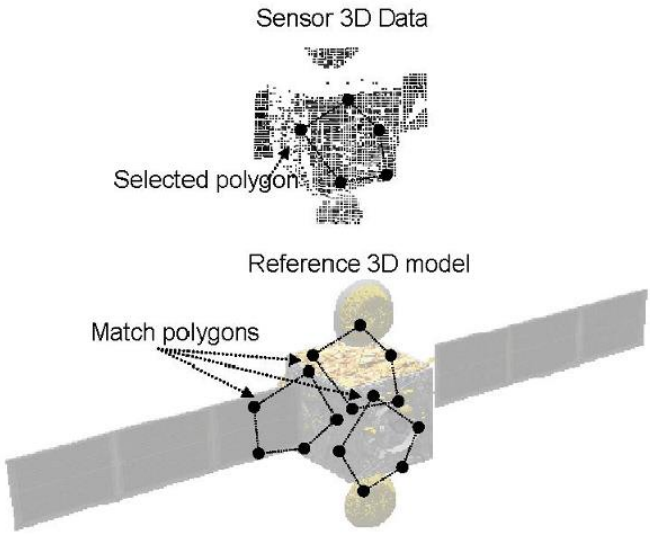

Figure 2.5 - Local features exploited by PAH architecture [18]

The first step of this method - offline processing - consists in generating a reference database (hash table) using a polygonal model of the object to localize, which has the aim of speeding up the polygon matching process. Let M be a polygonal model of the object to localize and S a sparse set of 3D points located on the model surface. The offline processing first generates the set of all segments that can be created on M from point pairs of S. The set of segments is then arranged in a hash table that efficiently stores length and connectivity information to allow direct segment lookup.

The first step of the algorithm is used to reduce the 6 dimensions pose search space. The search space is reduced by keeping (in the next step) the set of poses that have at least some overlapping surfaces between the input point cloud and the polygonal model M.

Congruent Tetrahedron Align

Another point-based approach, improved from PAH, is the Congruent Tetrahedron Align (CTA) algorithm, developed by Yin et al. and described in [19]. This method, instead of extracting features, finds the congruent tetrahedrons that are built on the scanning point cloud and on the model point cloud and, through the alignment of these tetrahedrons, performs the estimation of the pose. The scanning point cloud is the one acquired by the sensor while the model point cloud can be obtained from the 3D CAD model, which is known. When the scanning point cloud is obtained, a 3D convex hull will be constructed, through specific algorithms, to simplify it. Then, a tetrahedron with the largest volume is found in the vertices of the convex hull and, thanks to a hash table, the tetrahedron congruent to the latter is detected, making it possible to calculate the transformation between them. The algorithm pipeline is shown in Figure 2.6.

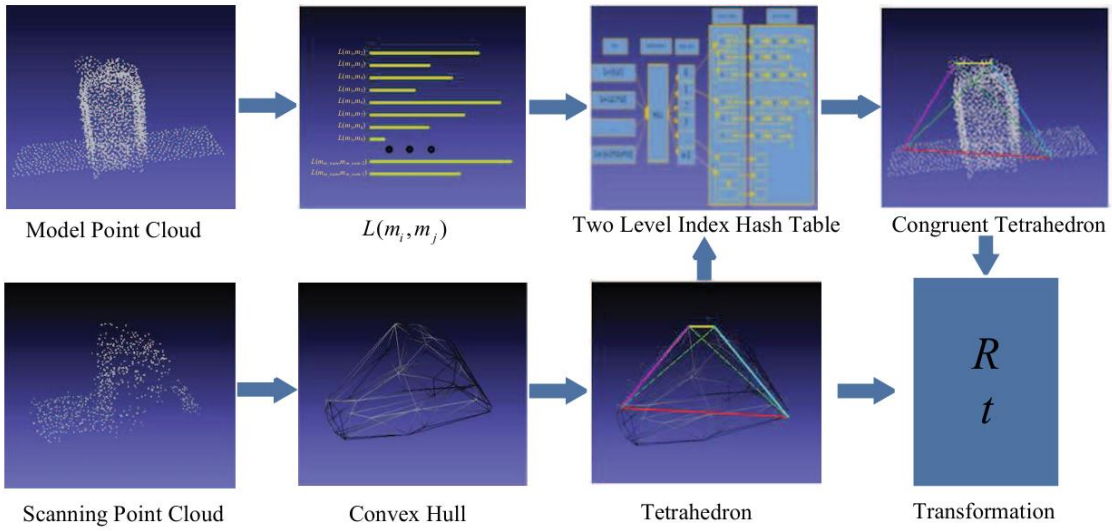

Figure 2.6 - CTA pipeline [19]

Based on the necessary and sufficient condition for two tetrahedrons to be congruent (according to which, if six sides of a tetrahedron are equal to the corresponding six sides of another tetrahedron, then the tetrahedrons are regarded as being congruent), the key idea of the CTA

algorithm is the search for the six corresponding sides. In order to find the correspondences in the next step more efficiently, in the first step of this method a Two Level Index Hash Table is built to store the information, which includes not only the point pair length of model point cloud but also the location topology information. To construct the hash table, a linear hash function is used to classify the various distance measures into a certain number of buckets.

Point Feature Histograms

A local feature that has been much studied in recent years is the Point Feature Histograms (PFH), introduced by Wahl et al. in [20] and then further developed by Rusu et al. in [21]. PFH is the first of a series of histogram descriptors, some of which are analyzed in Subsection 2.1.2, being global features. The PFH descriptor exploits the use of local and pose-invariant features and focuses on the use, for each point $\boldsymbol{p}_i$ of the point cloud, of the point $\boldsymbol{p}_i$ (query point) and a group of points close to it (k– neighborhood). The estimation of these features is based on the use of geometric relationships between the k-neighbors closest to $\boldsymbol{p}_i$ involving:

- The 3D coordinates of these points x, y, z;
- The normals to the surface at each point $\boldsymbol{n}_x, \boldsymbol{n}_y, \boldsymbol{n}_z$.

For each pair of points $\boldsymbol{p}_j$ and $\boldsymbol{p}_k$ in the k-neighborhood of $\boldsymbol{p}_i$ and their estimated normals $\boldsymbol{n}_j$ and $\boldsymbol{n}_k$, a source point $\boldsymbol{p}_s$ and target point $\boldsymbol{p}_t$ are selected and then a Darboux Reference Frame with the origin in the source point is defined as: $\boldsymbol{u} = \boldsymbol{n}_s$, $\boldsymbol{v} = (\boldsymbol{p}_t - \boldsymbol{p}_s) \times \boldsymbol{u}$ and $\boldsymbol{w} = \boldsymbol{u} \times \boldsymbol{v}$. Four features are estimated, which are shown in Equation 2.1:

$$\begin{cases} f_1 = \boldsymbol{v} \cdot \boldsymbol{n}_t \\ f_2 = \|\boldsymbol{p}_t - \boldsymbol{p}_s\| \\ f_3 = \boldsymbol{u} \cdot (\boldsymbol{p}_t - \boldsymbol{p}_s)/f_2 \\ f_4 = \text{atan}(\boldsymbol{w} \cdot \boldsymbol{n}_t, \boldsymbol{u} \cdot \boldsymbol{n}_t) \end{cases} \tag{2.1}$$

These features are then classified in the so-called bins, depending on the values they assume. In Figure 2.7 the diagram of the region of influence of a PFH is shown.

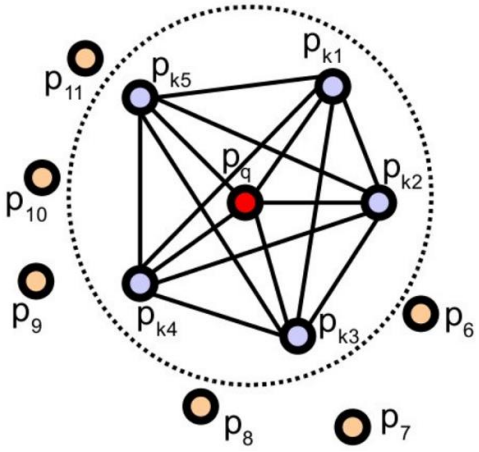

Figure 2.7 - The influence region diagram for a Point Feature Histogram. In red: query point. In blue: k – neighbors [21]

In this way, a multidimensional vector is obtained as output that encodes the local geometry of the point cloud, in the vicinity of the query point. The size of the vector depends on the number of features inserted (in this case four) and on the number of distinguishable cases depending on the value assumed by the single feature.

Therefore, for each point of the point cloud its PFH is estimated. Clearly, the more *particular* a PFH is compared to the others, the better and more robust the feature is for the correspondence determination phase. The technique that provides such PFH is called *Persistence Analysis*, which analyses the neighborhood of each point $\boldsymbol{p}$ of the point cloud, contained in a sphere whose center is $\boldsymbol{p}$ and whose radius is varied in an interval dependent on the size and density of the point cloud.

The main advantage of this feature is that it is invariant to position, orientation and point cloud density, and the histograms cope well with noisy datasets, but, a very important drawback of the PFH is the computational complexity: the theoretical computational complexity of the PFH for a given point cloud with n points is $O(n \cdot k^2)$, where k is the number of neighbors for each point $\boldsymbol{p}$ of the point cloud.

Fast Point Feature Histograms

The problem of the computational cost of PFH led to the development of a simpler and faster variant, called Fast Point Feature Histograms (FPFH), presented by Rusu et al. in [22], in which:

- Each query point is connected only to the nearest neighbors, then the features between the pairs of points are calculated, which this time are no longer 4 but 3, excluding the Euclidean distance f_2 (See Equation 2.1), obtaining for each point considered the Simplified Point Feature Histogram (SPFH);
- For each point considered in the previous step, the k-neighbors are considered, and therefore, exploiting the SPFH of the neighbors previously calculated, the final histogram (FPFH) relating to the point $\boldsymbol{p}$ is estimated, as shown in Equation 2.2:

$$FPFH(\boldsymbol{p}) = SPFH(\boldsymbol{p}) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\omega_k} SPFH(\boldsymbol{p}_k) \tag{2.2}$$

where the weight ω_k is the distance between $\boldsymbol{p}$ and $\boldsymbol{p}_k$.

In Figure 2.8 the diagram of the region of influence of a FPFH is shown. The great advantage of the FPFH descriptor is that it maintains most of the discriminative power as PFH, but in addition it is much faster; in fact, the theoretical computational cost of the FPFH is reduced to $O(n \cdot k)$, becoming able to estimate features almost in real time.

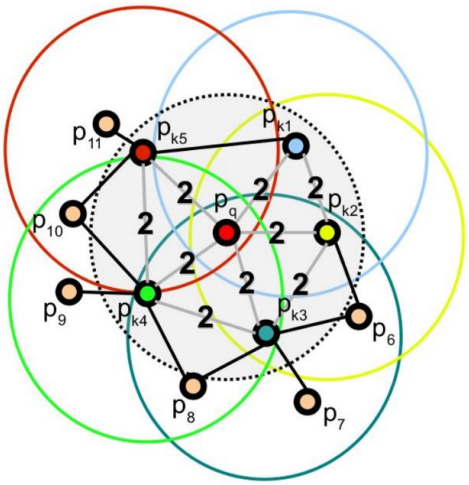

Figure 2.8 - The influence region diagram for a Fast Point Feature Histogram [22]

The output of this first phase is a set of candidate features which will be analyzed and compared with the features of the acquired point cloud in the next phase of correspondence search to find the optimal matching and thus align the two point clouds.

2.1.1.2. Point Cloud Alignment

Once the set of candidate features is obtained, the local features on one surface are compared with the local features on the other surface to obtain point-to-point correspondences. Finally, the transformation is estimated from the constructed correspondences for registering the two surfaces. This task is particularly challenging, as the generated correspondences may contain a large number of outliers due to symmetric structures, noise, clutter and occlusions. To address and solve this problem, various algorithms have been developed, and some of these are presented in this paragraph.

Sample Consensus Initial Alignment

A very popular algorithm in the field of computer vision, capable of counteracting this problem, is the Random Sample Consensus (RANSAC) algorithm, developed by Fischler and Bolles and presented in [32]. RANSAC is an iterative method to estimate the best parameters for a mathematical model that fits a dataset with outliers, which are ignored in the estimation. It is a non-deterministic algorithm in the sense that it produces a reasonable result only with a certain probability, which is greater the higher the number of iterations.

Starting from this method, a family of RANSAC-based algorithms has been developed for point cloud initial alignment purposes, one of which was presented by Rusu et al. in [22] and applied to the FPFH descriptor, the Sample Consensus Initial Alignment (SAC-IA). This algorithm iteratively carries out the following steps:

- *Keypoints Sampling*. At least three sample points are selected from the model point cloud while making sure that their pairwise distances are greater than a user-defined minimum distance;
- *Correspondence Search*. For each sample point, a list of K candidate points in the scene point cloud, with a similar local descriptor, is found. Then, one point from the candidates is randomly selected as the correspondence point;
- *Transformation Matrix Estimation*. With three correspondence pairs, it is possible to estimate the transformation matrix $\boldsymbol{T}_j$, where j is the iteration index;

- *Performance Evaluating*. The model point cloud is transformed using $\boldsymbol{T}_j$ and is compared to the scene point cloud. The quality of the comparison is measured using a Huber penalty measure as an error metric [33].

The robustness is proportional to the trial number. However, too many trials will decrease the efficiency. In spite of that, SAC-IA can still get a good result after many iterations and be widely used in the initial alignment process. Since the obtained transformation matrix may deviate from the real one, a refinement step follows, commonly performed through the ICP. Being a RANSAC-based method, SAC-IA is robust to noise and outliers. In particular, unlike the classic RANSAC algorithm, the SAC-IA reduces some unreasonable iterations by judging the qualification of current sampled correspondences according to a distance constrain. However, its efficiency and accuracy are limited in complex cluttered environment.

Fast Global Registration

Existing approaches are generally based on iterative methods that aim to search for correspondences between two point clouds, thanks to which the alignment/recognition process is then carried out. Much of the computational cost is wasted to compare the point cloud with candidates that are subsequently discarded, and in particular these methods are based on a first step of global estimation of the pose (first acquisition) and on a subsequent step of local refinement [11].

The Fast Global Registration (FGR) algorithm overcomes the limitations of these approaches, proving to be as precise as well-initialized local refinement algorithms and faster than them. This approach, presented by Zhou et al. in [34], unlike most existing methods, does not involve iterative sampling, model fitting, or local refinement. It does not require initialization and can align noisy partially overlapping surfaces.

Given two point sets, $\boldsymbol{P}$ and $\boldsymbol{Q}$, the key idea is to establish correspondences between them, after which an optimization process of an *objective* follows based on these correspondences. Importantly, during the optimization process, these matches are not recalculated.

In [34], the set of correspondences $\boldsymbol{K} = (\boldsymbol{p}, \boldsymbol{q})$ between $\boldsymbol{P}$ and $\boldsymbol{Q}$ is computed using the FPFH descriptor given the great speed with which this feature can be calculated and the accuracy it

guarantees. Specifically, three main steps are performed that aim at determining the set K based on the similarity between FPFH, through the Nearest Neighbor retrieval, and on the "compatibility" between tuples of correspondences, through the comparison of the lengths of segments obtained from pairs of points identified on the two point clouds.

Once the set of correspondences $\boldsymbol{K}$ is obtained, the objective is to minimize the distances between corresponding points, taking care not to consider spurious correspondences, thus optimizing the pose $\boldsymbol{T}$. Figure 2.9 shows an example with 2D point sets.

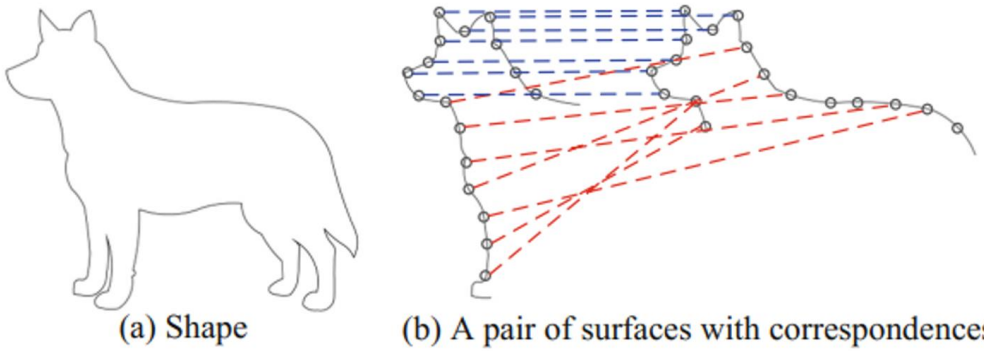

Figure 2.9 - Illustration with 2D point sets. In blue: genuine correspondences; in red: spurious correspondences [34]

To accomplish this task, an objective function E to be minimized is first defined, which contains in the formula a robust penalty ρ, the value of which must be chosen very carefully as it influences the fraction of matches to be excluded in the optimization process. This formula is shown in Equation 2.3.

$$E(\boldsymbol{T}) = \sum_{(\boldsymbol{p},\boldsymbol{q})\in\boldsymbol{K}} \rho(\|\boldsymbol{p} - \boldsymbol{T}\boldsymbol{q}\|) \tag{2.3}$$

As a robust penalty, a scaled Geman-McClure estimator is chosen, whose formula is shown in Equation 2.4 and where the parameter μ controls the range within which residuals have a significant effect on the objective.

$$\rho(x) = \frac{\mu x^2}{\mu + x^2} \tag{2.4}$$

After a reformulation of the objective function, using the Black-Rangarajan duality between robust estimation and line processes [35], and of the matrix $\boldsymbol{T}$ by rewriting it as a vector $\boldsymbol{\xi}$ containing the 3 rotational and 3 translational parameters, an iterative cycle is implemented which, through the Gauss-Newton method, recalculates $\boldsymbol{\xi}$ (and therefore $\boldsymbol{T}$) from time to time until convergence, i.e. until the minimum of the objective function is reached, or until μ falls below a certain threshold.

Given this operating principle of the FGR, this algorithm is very fast. In particular, since the correspondences are not recomputed during the optimization process, unlike SAC-IA, FGR is expected to be faster than the other algorithm. On the other hand, FGR has limited robustness to outliers, also due to the fact that the factor of robustness to outliers and spurious correspondences is strongly dependent on the estimator selected for the objective function. Furthermore, the FGR performances are evaluated under the assumption that the two surfaces to be aligned are partially overlapped.

Database Search: Geometric Hashing

The alignment of the point clouds, as well as through appropriate algorithms, some of which are illustrated above, can also be carried out thanks to the search and matching of the features of the model point cloud previously stored in a database. This in particular is the case of the GH, PAH and CTA algorithms, which, as already seen in the first step (paragraph 2.1.1.1), store data in a hash table, constructed in order to carry out a rapid search for correspondence and consequently align the point clouds.

In the case of GH, once the construction of the hash table containing the information relating to the object model is completed, in the online recognition phase the features are extracted from the range image acquired by the sensor and transformed into the reference system defined by a base, and are matched with the features stored in the database. The new coordinates of the points expressed in the new reference system enter the hash table and each base of the model previously stored in the hash table is given a vote. The model basis that scored the largest number of votes corresponds to the chosen image basis. Then, once the basis in the hash table is identified, the rigid transformation between the reference systems of the model and the image can be determined, thus obtaining the pose. In Figure 2.10 a scheme of the GH recognition phase is shown.

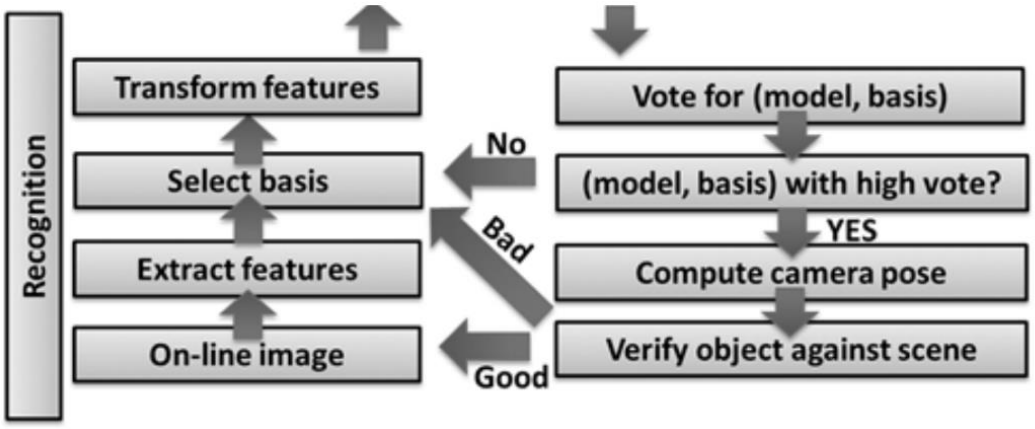

Figure 2.10 - Geometric Hashing recognition step. The diagram represents the continuation of Figure 2.4 [31]

Database Search: Polygonal Aspect Hashing

Instead, in the case of PAH, once the Offline Processing phase in which the hash table is constructed is finished, the Runtime Localization phase follows, i.e. the polygon matching process in which the reference database created in the offline phase and the sparse 3D point cloud obtained from the sensor are used, determining the position vector $\boldsymbol{T}$ and the rotation matrix $\boldsymbol{R}$ of the target model w.r.t. the sensor focal point. This step can be summarized as follows:

- Selection of a polygon of N points from the input point cloud;
- Reduction of the set of possible poses to those where the polygon input scan points line up with their corresponding model surface points within a specified tolerance, thanks to the hash table;
- From each matched polygon pairs, the corresponding relative pose ($\boldsymbol{T}$,$\boldsymbol{R}$) can be computed;
- Evaluation of the quality of the alignment between input point cloud and polygonal model, transforming the points of the point cloud into the model reference frame using ($\boldsymbol{T}$,$\boldsymbol{R}$). The solution is the pose candidate that exhibits the best alignment.

Database Search: Congruent Tetrahedron Align

Finally, in the case of the CTA method, once the tetrahedron is defined in the scanning point cloud and the hash table containing the information from the model point cloud is built, the process of searching for the congruent tetrahedron in the model point cloud is carried out. The output of this search are the four vertices of the tetrahedron in question. Once these vertices and therefore the tetrahedron is found, the pose can be estimated. In general, more than one corresponding tetrahedron can be found in the hash table, but only one is correct. In order to deal with this case and choose the correct one, the ICP algorithm is used to refine the registration result. The accuracy measurement of the algorithm is the ICP convergence score, computed by Equation 2.5:

$$f_{conv} = \frac{1}{n}\sum_{i=1}^{n} \|\boldsymbol{m}_i - (\boldsymbol{R}\boldsymbol{s}_i + \boldsymbol{t})\|^2 \tag{2.5}$$

where $\boldsymbol{s}_i$ and $\boldsymbol{m}_i$ are the vertices of the tetrahedra built respectively on the scanning and on the model point cloud, while $\boldsymbol{R}$ and $\boldsymbol{t}$ are the rotation matrix and the translation vector that define the transformation between the two point clouds. The transformation corresponding to the smallest ICP convergence score is the output of the CTA algorithm and the input of the pose tracking.

2.1.2. Global-Feature-Based Methods

As done for the local-feature-based methods, also for the global-feature-based ones a preliminary comment is made on the pipeline adopted by them and shown in Figure 2.11.

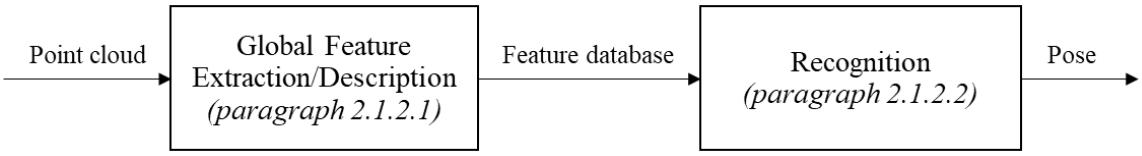

Figure 2.11 - Global-feature-based methods pipeline

The basic idea is to extract features - again not in all cases - from a series of point cloud models with different poses and store them in a database; subsequently, in the recognition phase, each

feature of this database is compared with the feature extracted from the point cloud acquired by the LiDAR, and the quality of this comparison is evaluated through an appropriate metric. The feature that exhibits the best comparison provides the pose parameters.

2.1.2.1. Global Feature Extraction/Description

This paragraph illustrates the feature database construction procedure adopted by some global-feature-based methods.

Online 3 DOF Template Matching

A series of methods that have caught on, especially in the space field, use *the entire point cloud as a global feature* and are the Template Matching (TM) techniques. The basic idea of this method is to compare the point cloud acquired by the sensor with a series of templates generated using a model point cloud in different poses, and evaluate the quality of matching through a correlation function. The pose is obtained from the template that presents the best match. However, strictly speaking, this method would require the generation of templates in the entire 6 DOF pose space, obtaining a huge number of templates and, consequently, taking up a lot of on-board memory and making the computational cost very high. To make this principle applicable to space applications, several ideas have been developed to reduce the necessary number of templates to generate and thus accelerate the subsequent recognition phase.

An example is given by Online 3 DOF TM, proposed by Opromolla et al. in [25], in which the position of the centroid of the acquired point cloud is exploited to estimate the relative position vector of the target with respect to the sensor. This greatly simplifies the problem; in fact, the online creation of the templates is carried out by restraining the pose search space to a 3 DOF database consisting only of the attitude parameters.

It is important to underline that, in this Template Matching algorithm, as well as in the other Online TM techniques that follow, the feature extraction/description step is not completely separated from the recognition step, as the entire database is not created before the recognition phase but, as the templates are generated, the recognition phase is also carried out (in which the quality of the matching between the *i-th* template and the acquired point cloud is evaluated).

Despite the significant improvement compared to the traditional Template Matching technique, given the greatly reduced number of templates to generate, it is still highly time consuming.

Online Fast 3 DOF Template Matching

A variant called Online Fast 3 DOF TM, presented by Opromolla et al. in [10], includes a strategy to speed up the recognition step; therefore, it is covered in paragraph 2.1.2.2.

PCA-based Online Template Matching

Another idea, applied in the PCA-based online TM, developed by Opromolla et al. in [28], is to exploit, together with the position of the centroid of the point cloud, also the Principal Component Analysis (PCA) [36] to estimate the target main axis $\boldsymbol{e}_M$. The main axis $\boldsymbol{e}_M$ is identified by the eigenvector associated with the maximum eigenvalue of the covariance matrix of the measured point cloud. It is important to underline that PCA can only be effectively adopted for elongated targets. Although there is this limitation of applicability, this feature fits perfectly into the space scenario as many spacecrafts and debris have this shape. The determination of the main axis of the target easily allows the calculation of 2 of the 3 Euler angles (roll, α and pitch, β), effectively limiting the application of the TM to *a single DOF database built online*. This clearly allows a strong reduction in the number of templates to be generated, and consequently in the amount of data to be stored on board, significantly lowering the computational cost compared to other TM techniques. The unknown Euler angle (yaw, γ) represents the rotation around the target main axis.

2 DOF Template Matching

Finally, the last variant of the Template Matching techniques that is illustrated is the 2 DOF TM, proposed by Guo et al. in [27]. The key idea of this method is linked to the definition of an attitude sphere where to represent the relative attitudes between LiDAR Coordinate System (LCS) and Target Coordinate System (TCS).This sphere is centered at the origin of TCS, and LCS is located on the sphere surface with Z axis (the line of sight of the LiDAR system) pointing to the center.

The first step of this method is an offline phase and starts from building the silhouette image template dataset offline, where silhouette image means the projection of the 3D point cloud on

the X-Y plane of the LCS. In addition, this method also exploits the centroid of point cloud data for the estimation of the spacecraft position, restricting the sampling within 3-DOF attitude domain, as already seen in the variants reported above. The assumption made with the attitude sphere is very important because, when the silhouette image is generated, rotating the LiDAR system around the line of sight, would not change the shape of the generated silhouette but only alters its orientations. This means that the γ angle can be unwrapped out of the attitude angles, since it can be calculated as the rotation between the scene silhouette image and the template image. Specifically, one silhouette image represents all attitudes that have the same (α,β) but different γ.

The silhouette generation pipeline is shown in Figure 2.12.

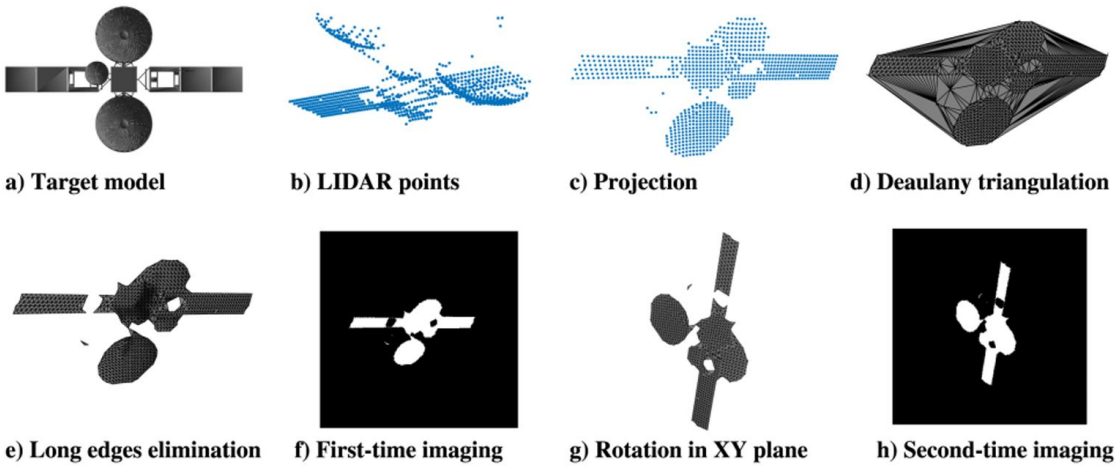

Figure 2.12 - Pipeline of silhouette image generation [27]

Once the point cloud is projected, the Delaunay triangulation algorithm is applied. This step is crucial since it ensures robustness of the shape against point sparseness, thus making the silhouette image resilient to translation. Then, edges longer than a certain threshold are eliminated and the silhouette binary image is formed, in order to store the silhouette shape for matching. An Image Coordinate System (ICS) is defined, where the X axis and the Y axis are set to be aligned with the X axis and Y axis of LCS. Subsequently, each silhouette image is rotated to a status where its two principal directions are aligned with the X axis and Y axis of ICS and finally a second-time silhouette imaging process follows. For binary image generation, the pixel length must be appropriately selected: a small length will result in a feature with higher descriptiveness, but it will also be easier to suffer from noise perturbations.

The last two steps of the silhouette generation pipeline are fundamental to allow the subsequent estimation of γ. So, the template dataset is constructed in a two Euler-angle domain (α,β), i.e., the azimuth angle and elevation angle in the attitude sphere. Unlike the other template matching techniques already seen, in this case the template is built offline, thus saving online processing time.

Viewpoint Feature Histogram

Taking up the concept of descriptors, further ones were developed from PFH and FPFH, first adding information representative of the entire point cloud, obtaining the so-called Viewpoint Feature Histogram (VFH).

The idea of developing the VFH (Rusu et al., [37]) arose from the characteristic of PFH and FPFH of being invariant to an object's scale and its pose. This invariance characteristic is useful if features must be extracted which, regardless of how two objects are oriented, allow their recognition and subsequent alignment using specific algorithms, but clearly alone are not sufficient for pose estimation. The goal was to create a descriptor that could be run in real time to simultaneously recognize an object and calculate its pose. VFH addresses the pose invariance of PFH/FPFH by considering the camera viewpoint as a fourth feature, which is concatenated to the three features of the FPFH. This fourth feature is the distribution of angles between the viewpoint direction (i.e. the position of the centroid of the point cloud) and each points' normal.

The most distinctive difference between PFH/FPFH and VFH is the number of histograms generated per point cloud: indeed, PFH/FPFH created a descriptor for each point in the point cloud, while VFH creates only one descriptor per scene point cloud. Thanks to the addition of this information, VFH may now be used to identify objects in 3D space, being more robust than PFH/FPFH in describing an object, and estimate their relative poses, however stopping at 5 DOF, being invariant to camera roll. Furthermore, other major flaws to VFH are its sensitivity to noise and occlusions, and scale invariance. Indeed, although the height of the VFH histogram implicitly conveys information about the scale of the object, as it is a function of the total number of points in the distance image, in the PCL implementation the histogram is normalized by the total number of points in the scene, which causes all histograms to have the same height range of [0-1]. This inconsistency provides insight into the difficulty these histograms have in conveying information about an object's scale.

Clustered Viewpoint Feature Histogram

Subsequently, the principle adopted by VFH on the entire point cloud was extended to clusters of point clouds, thus obtaining the Clustered Viewpoint Feature Histogram (CVFH).

The CFVH (Aldoma et al., [23]) addresses the deficiencies present in VFH being able to distinguish identical objects but of different scales (the CVFH histogram is not normalized by the total number of points in the scene, so the scale of the object is implicitly characterized by the histogram) and considering the effects of partial occlusions. CVFH remedies the roll invariance of VFH by concatenating a Camera Roll Histogram (CRH) onto the descriptor.

The basic idea is to identify smooth and continuous clusters $\boldsymbol{C}_i$ on the surface $\boldsymbol{S}$ to be described and use only the points in $\boldsymbol{C}_i$ to build a coordinate system while still using all points in $\boldsymbol{S}$ to describe its geometry. The cluster is smooth if the dot product of the normals of two neighboring points is larger than a threshold t_n, and it is continuous if two neighboring points are separated by less than Euclidean distance threshold t_d. Then, a point $\boldsymbol{p}_k$ of normal $\boldsymbol{n}_k$ is added to the cluster $\boldsymbol{C}_i$ if the constraint shown in Equation 2.6 is fulfilled:

$$\exists \boldsymbol{p}_j \in \boldsymbol{C}_i : \left\| \boldsymbol{p}_h - \boldsymbol{p}_j \right\| < t_d \wedge \boldsymbol{n}_h \cdot \boldsymbol{n}_j > t_n \tag{2.6}$$

Depending on the structure of $\boldsymbol{S}$, it might be composed of several $\boldsymbol{C}_i$ from which a different coordinate system is obtained and therefore a different CVFH histogram, each one describing the same surface but encoding it differently. So, the main difference between CVFH and VFH is that, instead of one histogram for the entire scene, there is now a CVFH histogram descriptor for each cluster in the scene. Each $\boldsymbol{C}_i$ is paired with a ($\boldsymbol{c}_i$, $\boldsymbol{n}_i$), respectively representing the centroid and the average of the normals of $\boldsymbol{C}_i$. Each pair ($\boldsymbol{c}_i$, $\boldsymbol{n}_i$) is then deployed as one of the axis of a point-wise reference frame from which three angular distributions (each made out of 45 bins) of the normal $\boldsymbol{n}_j$ can be computed (similar to the first three VFH features) and finally added in the corresponding histogram bin.

CVFH includes as well a fourth and fifth component (45 and 128 bins respectively) into the histogram, the fourth being based on the L1-distribution obtained from $\boldsymbol{c}_i$ and each $\boldsymbol{p}_j \in \boldsymbol{S}$ and the fifth resulting from yet another angular distribution obtained from each $\boldsymbol{n}_j$ and the central view direction (similar to the fifth VFH feature). The total size of a CVFH histogram is 308.

CVFH has been shown to deliver good results in the context of 3D recognition; however, it has the important drawback that, for pose estimation, it needs the addition of the CRH, which results in extra calculations and a larger overall feature histogram.

Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram

Finally, starting from the CVFH, the Oriented Unique Repeatable Clustered Viewpoint Feature Histogram (OUR-CVFH) was developed.

The OUR-CVFH descriptor (Aldoma et al., [24]) solves this main problem of the CVFH, introducing the concept of Semi-Global Unique Reference Frame (SGURF) which better integrates the information previously provided by the CRH, defining multiple repeatable coordinate systems on $\boldsymbol{S}$. There are five components in total to the OUR-CVFH, of which the concatenation creates a single descriptor. To generate the OUR-CVFH the following steps are performed:

- First, points whose curvature is higher than a certain t_c threshold are removed from the surface $\boldsymbol{S}$, yielding $\boldsymbol{S}^f$;
- $\boldsymbol{S}^f$ is now separated into smooth and continuous clusters $\boldsymbol{C}_k$, similarly to what CVFH does. Each cluster is initiated with an arbitrary point in $\boldsymbol{S}^f$ that has not yet been assigned to any cluster;
- Differently to CVFH, the points $\boldsymbol{p}_k \in \boldsymbol{C}_i$ are filtered once more by the angle between $\boldsymbol{n}_k$ and $\boldsymbol{n}_i$ (the average normal of the points in $\boldsymbol{C}_i$);
- Each $\boldsymbol{C}_i$ is associated with a pair ($\boldsymbol{c}_i$, $\boldsymbol{n}_i$) representing its centroid and average normal. For a specific $\boldsymbol{C}_i$, the associated SGURF is found and placed at its centroid.

So far, for a specific surface $\boldsymbol{S}$, N triplets ($\boldsymbol{c}_i$, $\boldsymbol{n}_i$, RF_i), where RF stands for Reference Frame, have been computed. The five components of the OUR-CVFH are defined as follows:

- First, $\boldsymbol{c}_i$ and $\boldsymbol{n}_i$ are used to compute the first three components of CVFH and the viewpoint component (the fifth component), as presented in [23]. The viewpoint component is however encoded using 64 bins instead of the original 128;
- The fourth component of CVFH is removed and instead the surface $\boldsymbol{S}$ is spatially described by means of the computed RF_i. To perform this, $\boldsymbol{S}$ is transformed from the

original coordinate system of $\boldsymbol{S}$ (the camera's reference frame) into the local RF_i. After the transformation, the points in $\boldsymbol{S}$ are divided into the 8 octants defined by the signed axes $(x^-, y^-, z^-) \dots (x^+, y^-, z^-) \dots (x^+, y^+, z^+)$. In order to account for perturbations due to noise or missing parts, interpolation is performed between octants by associating to each point $\boldsymbol{p}_k$ eight weights, one for each octant. The weights are computed by placing three 1D Gaussian functions over each of the axes centered at the RF_i origin with a standard deviation of σ. Because this interpolation scheme describes the likelihood that a particular point lies in a particular octant, this Gaussian weighting describes the likelihood that measurement noise could cause a point to appear in a different octant. Therefore, the spatial distribution information of points, appropriately weighted, grouped into octants is stored in 8 histograms.

The total size of the descriptor is $45 \cdot 3 + 8 \cdot 13 + 64 = 303$ bins. In Figure 2.13, an example of OUR-CVFH histogram is shown.

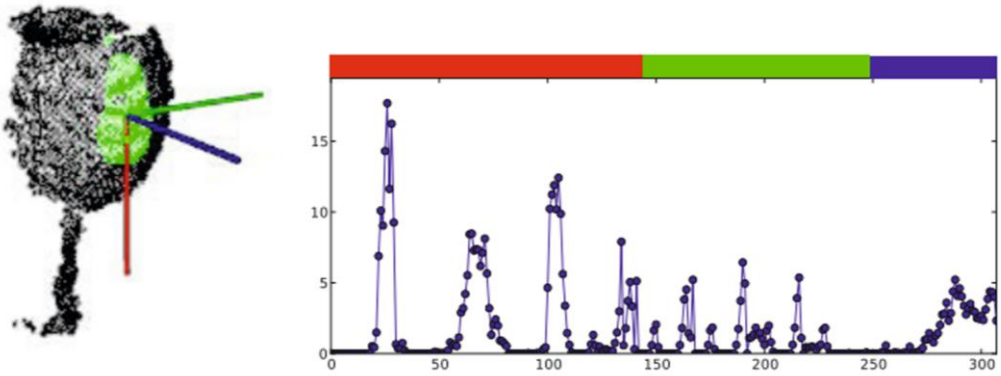

Figure 2.13 - Left: point cloud of a wine glass (black) with associated cluster $\boldsymbol{C}_i$ (green) and the SGURF reference frame. Right: the resulting OUR-CVFH histogram [24]

The first step of the global recognition pipeline using the OUR-CVFH descriptor is a training session, in which a database of features is generated. The recognition step is briefly analyzed in paragraph 2.1.2.2. Training is performed by either using an actual 3D sensor to take range images of a real object at various ranges and attitudes or using a 3D sensor simulator and a 3D mesh model. For space applications the latter option is usually preferable.

The OUR-CVFH descriptor has several advantages, such as the robustness to many types of occlusions, the speed in performing the pose estimate and the accuracy that this method can achieve with sufficient training, without then having to resort to a refinement phase with the ICP. On the other hand, this method generally requires high quality 3D models and evenly distributed high resolution 3D data; its recognition capabilities tends to decrease rapidly as the distance from the camera of the object to be recognized increases; finally, training at multiple ranges presents its own complication since, similarly to the CVFH, the only information on the scale of the object is that implicitly provided by the height of the OUR-CVFH histograms. Indeed, in [24], Aldoma et al. consider training from multiple viewpoints but only at a single constant range.

This architecture has been implemented in recent research for spacecraft navigation, producing favorable results [38, 39].

Basis Point Set

Finally, an interesting and efficient point cloud coding technique is briefly illustrated, the Basis Point Set (BPS), proposed by Prokudin et al. in [17]. The point cloud is encoded as a fixed-length feature vector $\boldsymbol{X} = \{x_1, \ldots, x_n\}$, computed as the minimal distances to a fixed set of points $\boldsymbol{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_k\}^T$. A graphical representation of this technique is shown in Figure 2.14.

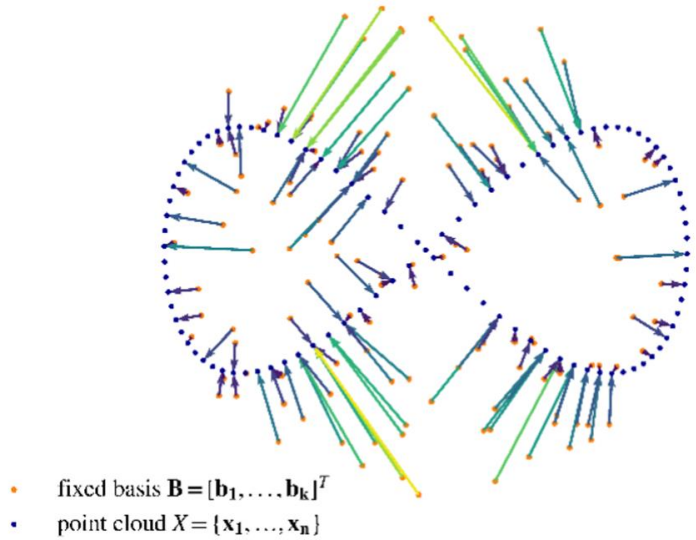

Figure 2.14 - Basis Point Set encoding for point clouds [17]

Therefore, every point cloud is reduced to a relatively small fixed-length vector, whose length can be adjusted to meet computational constraints for specific applications and represents a trade-off between fidelity of the encoding and computational efficiency. Compared to other encodings of point clouds, the proposed representation also has an advantage in being more efficient with the number of values needed to preserve high frequency information of surfaces. Finally, this type of coding allows the BPS method to easily deal with the problem of varying the cardinality of point clouds.

The feature estimation process is divided into three main phases:

- *Normalization*. The encoding algorithm takes in input a set of point clouds $\boldsymbol{X}_i$. Every point cloud can have a different number of points. In this first step, all point clouds are normalized to a fit unit ball;
- *BPS Construction*. To obtain the Basis Point Set vector, k points are randomly selected from a ball of a given radius r, as shown in Equation 2.7:

$$\boldsymbol{B} = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_k]^T, \boldsymbol{b}_j \in \boldsymbol{R}^d, \left\|\boldsymbol{b}_j\right\| \leq r \tag{2.7}$$

  where $d = 3$ for the case of 3D point clouds. This set is arbitrary but fixed for all point clouds in the dataset; r and k are hyper-parameters of the method, and the latter can be used to determine the trade-off between computational complexity and the fidelity of the representation;
- *Feature Computation*. Finally, for each point cloud in the dataset, the feature vector is constructed by calculating the *minimum distance* from every basis point to the nearest point in the point cloud under consideration (Equation 2.8) or by storing not only the distance information, but the entire *delta vector* from each basis point to the nearest point in the original point cloud (Equation 2.9):

$$x_i^B = [\min_{\boldsymbol{x}_{ij} \in \boldsymbol{X}_i} d(\boldsymbol{b}_1, \boldsymbol{x}_{ij}), \dots, \min_{\boldsymbol{x}_{ij} \in \boldsymbol{X}_i} d(\boldsymbol{b}_k, \boldsymbol{x}_{ij})]^T, x_i^B \in \boldsymbol{R}^k \tag{2.8}$$

$$\boldsymbol{X}_i^B = \{(\arg\min_{\boldsymbol{x}_{ij} \in \boldsymbol{X}_i} d(\boldsymbol{b}_q, \boldsymbol{x}_{ij}) - \boldsymbol{b}_q)\} \in \boldsymbol{R}^{k \times d} \tag{2.9}$$

In [17] the BPS is introduced as a method for estimating global point cloud features that can be used as input for *learning algorithms*; therefore, since a recognition phase has not been treated using classical methods, this part will not be present in this literature analysis.

2.1.2.2. Recognition

This paragraph illustrates the matching process between the features stored in the database and the features of the measured point cloud.

Online 3 DOF Template Matching

As already mentioned in paragraph 2.1.2.1, in Template Matching techniques, once the database is built (or, after the generation of each individual template, in the case of Online TM approaches), a phase follows in which the point cloud templates and the point cloud acquired by the sensor are compared and, through a correlation function, the quality of this comparison is evaluated.

In the case of Online 3 DOF TM, after template generation, the centroid overlapping and correlation function evaluation phases follow, which allow the best correlation and therefore the pose of the target to be identified. The correlation parameter selected in [25] is the mean square distance of corresponding points between the template itself and the acquired point cloud. The best correlation is the one that minimizes the correlation function shown in Equation 2.10:

$$\boldsymbol{C}(q, \boldsymbol{T}) = \frac{1}{N_p} \sum_{i=1}^{N_p} \left| \boldsymbol{P}^i_{SENSOR} - \boldsymbol{P}^i_{TEMPLATE}(q, \boldsymbol{T}) \right|^2 \tag{2.10}$$

where $\boldsymbol{T}$ is the relative position vector, q is the unit quaternion describing the relative attitude, N_p is the dimension of the LiDAR point cloud, $\boldsymbol{P}^i_{SENSOR}$ is the *i-th* point of the LiDAR point cloud and $\boldsymbol{P}^i_{TEMPLATE}$ is the *i-th* point of the template point cloud.

A diagram of the overall Online 3 DOF TM architecture is shown in Figure 2.15.

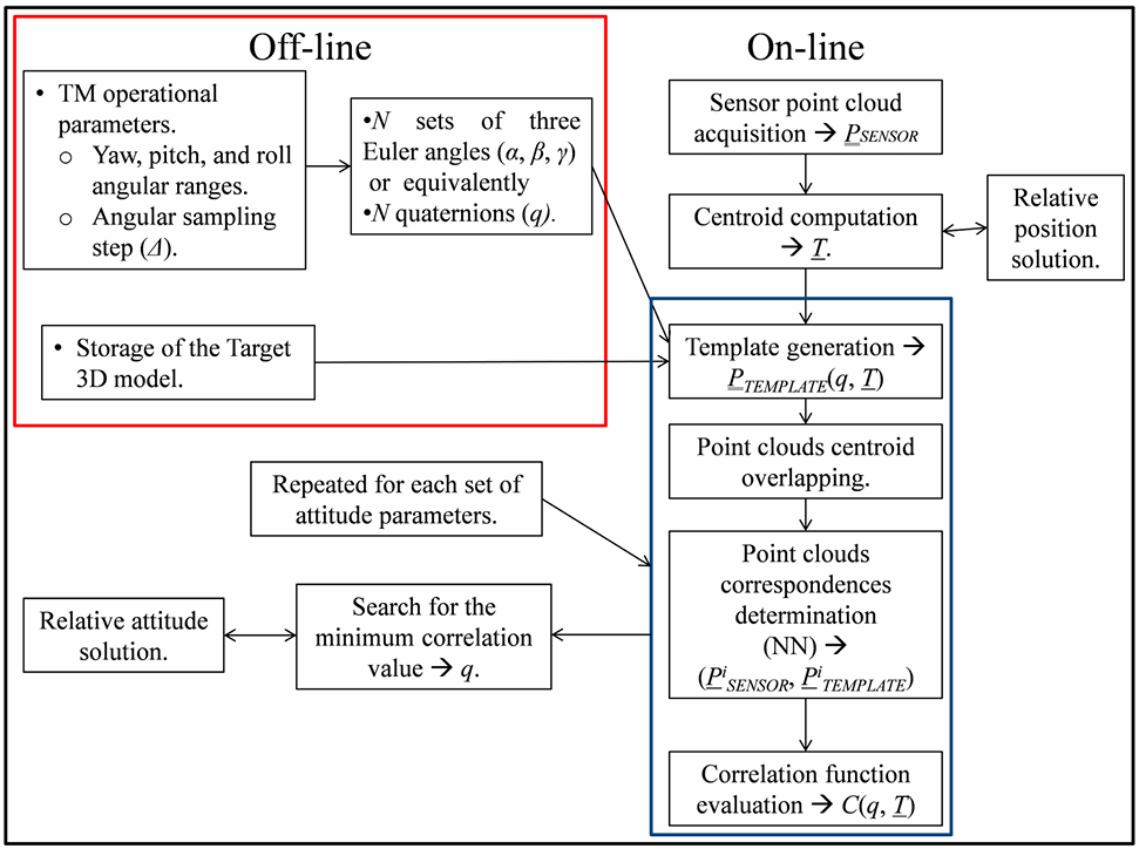

Figure 2.15 - Flow diagram of the Online TM algorithm [10]

However, in the presence of self-occlusion or truncated point cloud conditions, it becomes difficult to guarantee adequate results; therefore, to counter these problems, Nocerino et al. in [26] have proposed improvements to the 3 DOF TM architecture, as well as in the PCA-based variant. Specifically, a *centroid correction phase* is performed once the best correlated template is found, since the centroid may exhibit a notable displacement from the true center of the target, especially along the sensor boresight direction, due to the fact that the measured point cloud exclusively captures the visible portion of the target surface within the sensor's Field of View (FOV). The position estimation is updated using Equation 2.11:

$$\boldsymbol{T}_{new} = \boldsymbol{T} - \boldsymbol{\delta} \tag{2.11}$$

where $\boldsymbol{\delta}$ is the difference between the centroids of the acquired datasets and of the best correlated template.

Online Fast 3 DOF Template Matching

As anticipated in paragraph 2.1.2.1, the Online 3 DOF TM Recognition phase is strongly optimized in the Online Fast 3 DOF TM variant, operating a further reduction in the number of templates in a step preceding the matching between the point cloud template and the point cloud acquired by the sensor. More precisely, before moving on to the matching of the point clouds, various templates are already discarded through the estimation of the error in the average distance between the points of the point cloud and the sensor boresight axis, calculated for both point clouds (sensor and template); if this error exceeds a certain threshold (determined through numerical simulations), then the template is discarded.

In the related paper it is observed that, although the Online Fast TM presents less accurate results than the classic 3 DOF TM, this procedure leads to a significant reduction in the computational cost, and the advantage of this new method lies in the fact that, by choosing a small enough step angle with which to sample the Euler angles and build the templates, the difference in accuracy with the classical method is negligible, despite a significant reduction in the computational cost. Instead, as the angular step increases, it can be seen that the difference in accuracy increases, and therefore, despite the higher computational cost, the classical method becomes preferable.

PCA-based Online Template Matching

Similarly, also for the PCA-based Online TM a correlation function is used to measure the quality of the matching between the templates in order to find, from the best match, the unknown value of γ. The correlation function is defined as the mean squared distance of corresponding template/sensor points, as shown in Equation 2.12:

$$\boldsymbol{C}(\gamma) = \frac{1}{N_p}\sum_{i=1}^{N_p}\left|\left(\boldsymbol{P}^i - \boldsymbol{P}_C\right) - \left(\boldsymbol{P}_T^i(\gamma) - \boldsymbol{P}_{CT}(\gamma)\right)\right|^2 \tag{2.12}$$

where $\boldsymbol{P}_T^i(\gamma)$ and $\boldsymbol{P}_{CT}(\gamma)$ represent the *i-th* point and the centroid of the template point cloud associated with each value of γ. The substantial difference compared to the 3 DOF variants is the problem of the ambiguity of the pose solution due to the use of PCA. Indeed, from this procedure, 2 pose solutions are obtained, as the PCA allows only the direction of the main axis

to be found, but not its direction, thus obtaining 2 solutions of α and β, and consequently having to run the TM part 2 times, obtaining 2 values of γ. This ambiguity is solved by the acquisition-to-tracking transition step, which consists in applying the ICP algorithm twice, exploiting the Nearest Neighbor (NN) and the Normal Shooting (NS) approaches [40].

As in the 3 DOF variant, improvements have also been made in this variant [26]: in addition to the correction of the centroid, already seen for the 3 DOF architecture, a similar approach can be adopted to also correct the estimate of the main axis of the target in the PCA-based TM case. To carry out this correction, the PCA is applied not only to the point cloud acquired by the target, obtaining a main axis that is not very precise due to the partial visibility of the point cloud, but, following the superposition of the centroids, it is also applied to each template, in order to find, in addition to the main axis of the template, the angle between the two main axes, so as to be able to apply a rotation to each template. Once the best template is found, the corresponding rotation matrix is applied to the main axis direction of the acquired dataset to update its estimate.

2 DOF Template Matching

Instead, in the 2 DOF TM, once the template database is built, a silhouette image is constructed using the point cloud acquired by the LiDAR, after which the binary image matching process follows. To measure the quality of the matching between binary images, the *Jaccard coefficient* is chosen, which is defined in Equation 2.13:

$$S = \frac{\boldsymbol{A} \cap \boldsymbol{B}}{\boldsymbol{A} \cup \boldsymbol{B}} \tag{2.13}$$

where $\boldsymbol{A}$ and $\boldsymbol{B}$ are two binary image vectors. All binary images, stored as vectors, are organized into a matrix $\boldsymbol{F}_M = (\boldsymbol{f}_1^T, \boldsymbol{f}_2^T, \ldots, \boldsymbol{f}_m^T)^T$ where m is the number of templates. At this point, given a query scene image $\boldsymbol{f}_s$, through simple multiplications between matrices, it is possible to calculate, for each comparison between the templates $\boldsymbol{f}_i$ and $\boldsymbol{f}_s$, the Jaccard coefficient and select as the nearest template the one corresponding to the highest coefficient. Finally, once the nearest template $\boldsymbol{f}_m$ is found, the pose estimation can theoretically be carried out, having found the pair of Euler angles (α_m, β_m) through the template and being able to calculate γ as the rotation between the scene silhouette image and the template image. However, for symmetric

targets, there are cases of ambiguity in pose estimation; there may be more than two different attitudes that will result in the same template feature. This problem is addressed by adopting a *principal-direction-based strategy*, exploiting small asymmetrical parts of the target, and finally the ICP algorithm is used to refine the pose.

The results shown in [27] demonstrate that 2 DOF TM outperforms CTA, 1 DOF TM and Fast 3 DOF TM in success rate and time consumption but, on the other hand, its main limitation is that the target is assumed to be completely in the LIDAR FOV.

Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram

Very similarly to the methodology adopted by the TM algorithms in the Recognition phase, the actual pose estimation phase using the OUR-CVFH histogram is also based on a matching process between the descriptors extracted from the *measured scene set* and those extracted from the *training set* created in the first step. The matching process is briefly described below:

- Every descriptor belonging to the measured scene set is matched with every descriptor belonging to the training set using a distance metric. From this process, N candidates are found that produce the smallest distance metric;
- For each of the N best matches, ICP refines the pose by aligning the model points with the scene points. This process returns N separate 6 DOF relative pose estimates;
- Finally, the best candidate is selected. This step gives preference to matching clusters of similar size by considering the number of inliers and outliers of the point clouds. With the point clouds in the same reference frame, a point in the model cloud is considered an inlier if the distance to a point in the scene cloud is within a threshold distance t_i, otherwise it is an outlier. Then, a cost metric is calculated, which is defined in Equation 2.14:

$$J = \#inliers - \lambda \, \#outliers \tag{2.14}$$

where λ is used to weight the outlier count. The best candidate is determined as the one maximizing J.

2.1.3. Literature Analysis Conclusion

This Subsection contains a final summary of the performance of the analyzed methods, also in this case carried out by separately considering the feature extraction/description logical block (Table 2.1) and the alignment/recognition block (Table 2.2). Table 2.1 is built with the aim of summarizing: (1) The type of feature (local or global); (2) Whether feature extraction is required; (3) Pros and cons of the methodology adopted; (4) If it is applied in the space field.

Feature	*Type*	*Extr.*	*Pros*	*Cons*	*Space*
Geometric Primitives (GH)	Local	Yes	Pose and scale invariant features	Evenly distributed high resolution 3D data required	No
Polygons (PAH)	Local	No	A polygon with large surface area will provide fewer polygon matches. More polygons can increase robustness to outliers and occlusions	More polygons increase the computational cost. It is essential to find a good compromise	Yes
Tetrahedrons (CTA)	Local	No	The greater the volume, the greater the robustness and reliability of the method	The number of buckets must not be very large because, although the correspondence search becomes more accurate, the computational cost increases	Yes
PFH	Local	Yes	Invariance to position, orientation and point cloud density; the histograms cope well with noisy datasets	High computational cost, not suitable for real-time applications	No
FPFH	Local	Yes	Much lower computational cost, compared to PFH, while maintaining the same advantages and discriminative power	Obviously, some information is lost, compared to PFH	No

Point Cloud Template (TM)	Global	No	The point cloud is sparse, reducing the amount of data to be processed in the recognition phase	Due to the symmetries, errors in the pose may arise due to ambiguity	Yes
Point Cloud w/main axis (PCA TM)	Global	No	Addition of the direction information along which more points are concentrated	The orientation of the principal axis is not established	Yes
Silhouette image (2 DOF TM)	Global	No	Silhouette robustness against point sparseness, resilience to translation. It is stored as a binary image, resulting in simplification of matching	The pixel length must be appropriately selected: a small length will result in a feature with higher descriptiveness, but it will also be easier to suffer from noise perturbations	Yes
VFH	Global	Yes	VFH more robust than PFH / FPFH in describing an object	Sensitivity to noise and occlusions; scale invariance; invariance to camera roll. Only allows 5 DOF pose estimation	No
CVFH	Global	Yes	Ability to distinguish identical objects of different scales and considering the effects of partial occlusions	Requires CRH to resolve roll invariance (larger overall feature histogram); scale sensitivity	No
OUR-CVFH	Global	Yes	Robustness to many types of occlusions	Scale sensitivity	Yes
BPS	Global	No	Point cloud reduced to a relatively small fixed length vector. Fast encoding process while still maintaining accuracy in surface description	Too many points, although they increase the information content, also increase the computational cost, and vice versa if too few are chosen	No

Table 2.1 - Feature Extraction/Description pros and cons

Note how the invariance property is considered a pro for local features, while a con for global ones, precisely due to the difference in the methodology adopted to exploit them.

Table 2.2, similarly, is constructed to summarize: (1) Pros and cons of the applied methodology; (2) If this method is applied in the space field.

Method	*Pros*	*Cons*	*Space*
GH	Ability to handle partially occluded objects. Efficient online recognition, since most of the computational work is done during the offline step	A transformation which is based on correspondences of two base pairs may be sensitive to noise	No
PAH	Efficient with very sparse point clouds; the input point cloud is not expected to be organized in any specific pattern	The point cloud should cover as much of the target as possible	Yes
CTA	Efficient with very sparse point clouds. Accurate and robust to noise	Ambiguity resolved through ICP algorithm. The model point cloud should be resampled as uniform as possible	Yes
SAC-IA	Robustness to noise and outliers	Limited efficiency and accuracy in complex cluttered environment. Higher computational cost than FGR	No
FGR	It does not involve iterative sampling, model fitting, or local refinement, correspondences are not recomputed. Hence, it is very fast. It is accurate and robust to noise	Limited robustness to outliers, highly dependent on the choice of estimator for the objective function. Tested on partially overlapping surfaces	No
Online 3 DOF TM	No offline preprocessing stage required. Efficient with sparse point clouds. Reduced computational cost and amount of on-board stored data w.r.t. traditional 6 DOF TM. Improved centroid estimation, counteracting self-occlusion problems	Compared to subsequent TM variants, it is still highly time consuming	Yes

Online Fast 3 DOF TM	Reduced computational cost w.r.t. 3 DOF TM, coupled with a negligible loss of success rate when the attitude search space is adequately sampled or the point cloud is dense enough. Improved centroid estimation, counteracting self-occlusion problems	By increasing the angular step too much, the trade-off between computational cost and accuracy is no longer convenient	Yes
PCA-based Online TM	Significantly reduced computational cost thanks to PCA. Improved centroid estimation and principal axis estimation, counter acting self-occlusion problems	It is an effective method only if applied to elongated targets. PCA produces ambiguities to be resolved in the acquisition-to-tracking transition step	Yes
2 DOF TM	Ability to work with sparse point clouds. Experimental results demonstrate that it outperforms CTA, 1 DOF TM and Fast 3 DoF TM in success rate and time consumption	For symmetric targets there are cases of ambiguity in pose estimation, ICP required to refine the pose. The target must remain completely in the LiDAR's view	Yes
OUR-CVFH	It is fast in performing the pose estimate and can achieve high accuracy with sufficient training, without then having to resort to a refinement phase with the ICP	It generally requires high quality 3D models and evenly distributed high resolution 3D data; its recognition capabilities tend to decrease as the distance increases	Yes

Table 2.2 - Alignment/Recognition pros and cons

3. Methodologies

The literature analysis has revealed advantages and disadvantages of some of the classic methods of feature extraction/description and alignment/recognition of point clouds.

For this thesis work, great attention is paid to *point-normal structures* and *Hash Tables*, given the interesting strengths emerged from Chapter 2. In particular, point-normal structures are handled both as local and non-local features: in the first case using FPFHs; in the second case, by matching pairs of points with their relative normals, called Point-Pair-Features (PPF) [41, 42, 43], which will be referred to as *surflet pairs*. In this Chapter, all the analyses carried out are presented, which converge in the development of three feature-based algorithms:

- A *Label-based RANSAC* method, which collects offline FPFH statistics for elementary geometrical elements of the target satellite, and then finds online the target points with similar FPFH statistics;
- A *Persistence-Analysis-based (PA-based) RANSAC* method, which uses the so-called Persistence Analysis to identify the points with the most distinctive FPFHs in the point clouds;
- A *PPF-based RANSAC* method, which only exploits the information contained in pairs of points with their local normal vectors to identify correspondences.

Specifically, after having defined the reference geometry used for the analyses (Section 3.1) and after several preliminary analyses conducted on the estimation of the normals and the FPFH (Subsection 3.2.1 and paragraph 3.2.2.1), the studies carried out on the FPFH extraction techniques used for the above-mentioned algorithms are presented, focused on Persistence Analysis (paragraph 3.2.2.2) and on the segmentation of the point cloud in geometric primitives (paragraph 3.2.2.3), key principles of the PA-based RANSAC and Label-based RANSAC methods respectively, while for the PPF-based RANSAC method no feature extraction technique is used. In addition, the analyses conducted on the construction of the Hash Tables are presented (Section 3.3). Finally, the offline phase (Section 3.4) and online phase (Section 3.5) of the above-mentioned algorithms are outlined, to obtain the initial pose guess which is subsequently refined through post-processing (Section 3.6).

3.1. Reference Geometry

The global registration algorithms presented in this thesis work have been developed using as reference geometry the DLR Client Satellite of the OOS-SIM (On-Orbit Servicing Simulator for Capture), the German Aerospace Center Robotics and Mechatronics (DLR-RM) laboratory facility for realistic simulation of on-orbit servicing scenarios [44]. Furthermore, for a clear description of the implemented operations, it is worth defining the TRF of the reference geometry. The frontal and lateral views of the CAD model of this target and the TRF are shown in Figure 3.1.

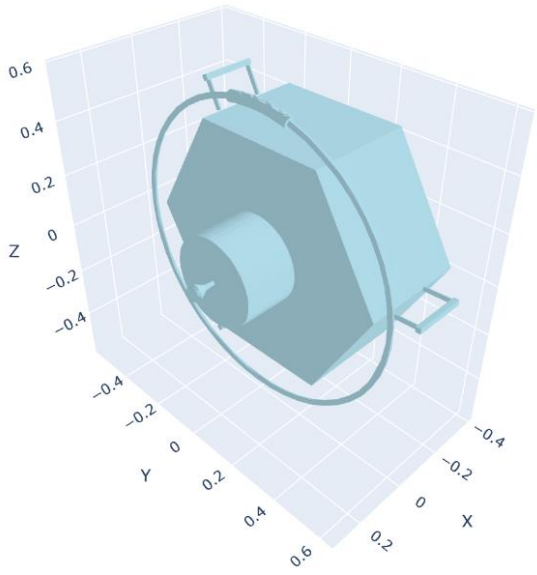

Figure 3.1 - CAD model of the OOS-SIM Client Satellite and TRF representation

As can be observed from Figure 3.1, the considered satellite geometry has a 6-fold symmetry of the structure, broken only by minor elements, with respect to the maximum inertia axis. This (approximate) symmetry adds complexity to the pose estimation problem. Regarding the TRF, instead, it can be observed that the x-axis is the roll axis, while the y and z axes lie on the upper face of the main body.

Starting from the CAD of the reference geometry, a dataset of 1000 synthetic point clouds has been generated by simulating two Velodyne™ VLP-16 scanning LiDARs, rotated by 90° with respect to each other. Random samples are drawn with a pointing constraint according to which

the target is in the Field Of View (FOV) of both sensors, by varying the relative position (between 1 m and 2 m distance) and the relative attitude of the target. The attitude, in particular, is generated in such a way as to uniformly cover SO(3) [45].

The resulting point clouds appear to be partial, sparse and non-uniform, but are unaffected by data artifacts such as noise and outliers.

3.2. Feature Extraction Analysis

This Section describes all the analyses carried out for efficient feature extraction. Since the FPFHs are closely related to how the local normals are estimated, a preliminary analysis on the estimation of point normals is first shown.

3.2.1. Normal Estimation Analysis

All the analyses are performed in Python environment; specifically, the Open3D library is used, an open-source library that allows the visualization and processing of 3D data, including the estimation of normals and FPFH [46]. The function used by Open3D for estimating normals computes them for each point of the point cloud. Given a query point, the function finds the neighborhood of that point and computes the principal axis using covariance analysis. In particular, such neighborhood is defined through two tuning parameters provided as input, r and max_{nn}, i.e., the radius of a sphere, centered in the query point, that includes the neighborhood points and the maximum number of Nearest Neighbors (NN) to be collected, respectively. The analysis presented in this Section focuses on the determination of optimal values of these parameters.

The procedure adopted for normal estimation analysis is the following: given as input a model point cloud, generated from the satellite CAD model, and the LiDAR scans with the ground truth of the poses, knowing the pose the alignment between the two point clouds is performed; then, for each point of the LiDAR point cloud, the NN is identified in the model point cloud by exploiting a *KDTree structure*. A KDTree is a data structure for storing a finite set of points from a k-dimensional space [47, 48].

At this point, for each pair of correspondences, the relative normals are compared and the angular error is estimated; finally, the mean, median and standard deviation of the error vector are computed. Additionally, the normal estimation is improved by exploiting the sensor viewpoint directions, i.e. the rays from the sensor viewpoint to each data point of the LiDAR point cloud, as a reference for an outward direction at each point (although of course mostly not orthogonal to the surface). In practice, the viewpoint correction is implemented through Equation 3.1:

$$\cos^{-1}\left(\boldsymbol{n}_L(i) \cdot \frac{-\boldsymbol{p}_L(i)}{\|\boldsymbol{p}_L(i)\|}\right) > \frac{\pi}{2} \tag{3.1}$$

where $\boldsymbol{n}_L(i)$ and $\boldsymbol{p}_L(i)$ represent respectively the normal vector and the position *(x, y, z coordinates)* of the *i-th* point of the LiDAR point cloud w.r.t. the sensor origin; in Equation 3.1, $\boldsymbol{p}_L(i)$ is considered with the opposite direction. Graphically, $\boldsymbol{n}_L(i)$ and $-\boldsymbol{p}_L(i)$ represent respectively the red and the black arrow shown in Figure 3.2.

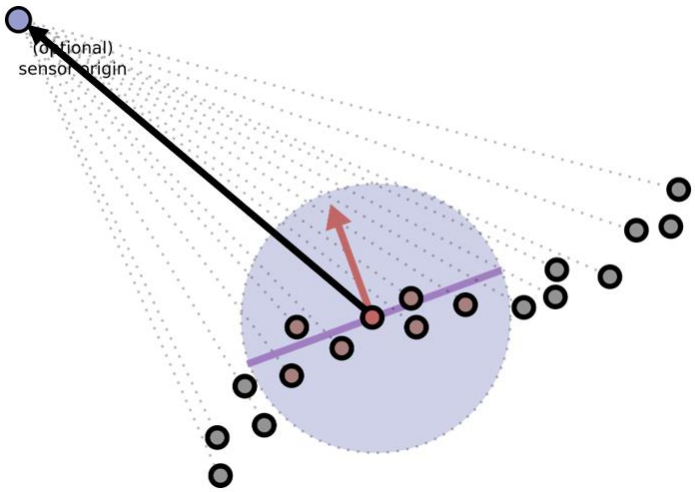

Figure 3.2 - Graphical visualization of normal correction using sensor viewpoint [49]

From Figure 3.2, the meaning of Equation 3.1 can be easily understood: in fact, if the angle between $\boldsymbol{n}_L(i)$ and $-\boldsymbol{p}_L(i)$ is less than 90°, as in the figure, then the normal is well oriented, because it exits the surface and is directed towards the origin of the sensor; if, instead, the angle is greater than 90°, it means that the normal is entering the surface and therefore must be reversed.

Given the very high number of point clouds in the dataset, the normal estimation analysis has been performed on only three scans by analyzing, varying r and max_{nn}, the mean and median of the angular error vector. Figure 3.3 shows the trends of the mean error as r varies, for fixed max_{nn}, and vice versa, for a LiDAR scan of the dataset, named Scan 1.

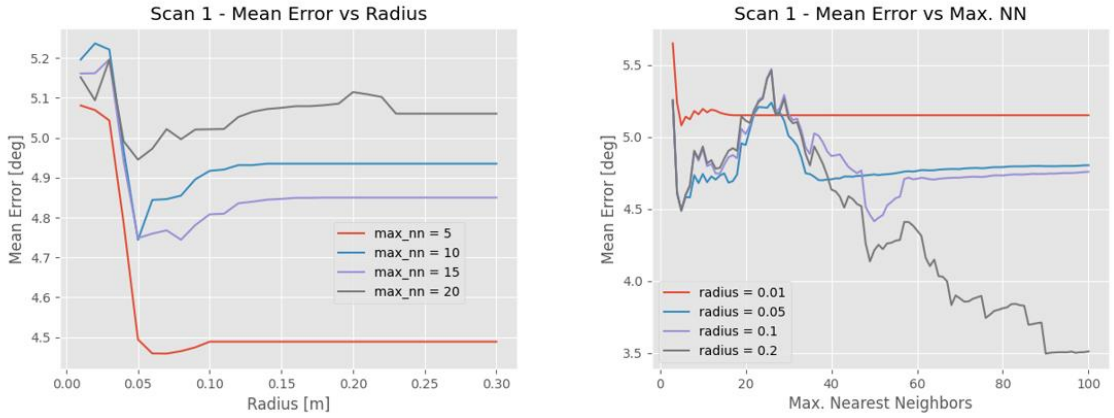

Figure 3.3 - Mean error plots for Scan 1 by varying r and max_{nn}

From the plots shown in Figure 3.3 it can be observed that the mean error always remains very low in the analyzed ranges of r (from 0.01 to 0.3 m) and max_{nn} (from 3 to 100), remaining between 3° and 6°. From the plot on the left, first of all it can first observed that, given a fixed max_{nn}, at a certain point a neighborhood radius r is reached, beyond which the situation no longer changes, since evidently the same neighborhood of points is always taken. Similarly, given a fixed r, once a certain value of max_{nn} is reached, the error becomes constant, and this is clearly more evident for small radii. Regarding the median of the error vector, instead, it remains much lower than the mean (order of magnitude of 10^{-5}), indicating that many normals are well estimated, but with some outliers that raise the mean of the error. Finally, while from the left plot the situation seems to be better for the case $max_{nn} = 5$, from the right plot, instead, it can be seen that, by increasing max_{nn} beyond 30 (approximately), the situation becomes more favorable for bigger r.

These results, however, are related to a rather particular case of point cloud, since it represents a scan with many planar points, as can be seen from the colored point cloud, produced considering, as an example, $r = 0.07\ m$ and $max_{nn} = 5$, shown in Figure 3.4. From the figure it can be seen that the highest angular errors come from the high curvature areas.

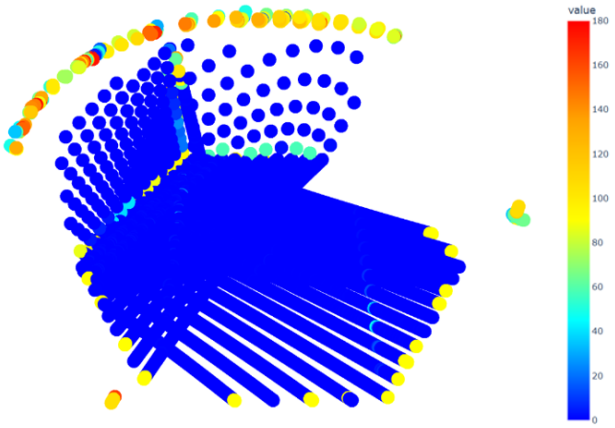

Figure 3.4 - Scan 1 colored point cloud

Regarding instead the subsequent scans (indicated as Scan 2 and Scan 3), the plots shown in Figure 3.5 and Figure 3.6 show trends that are quite different from those observed for Scan 1.

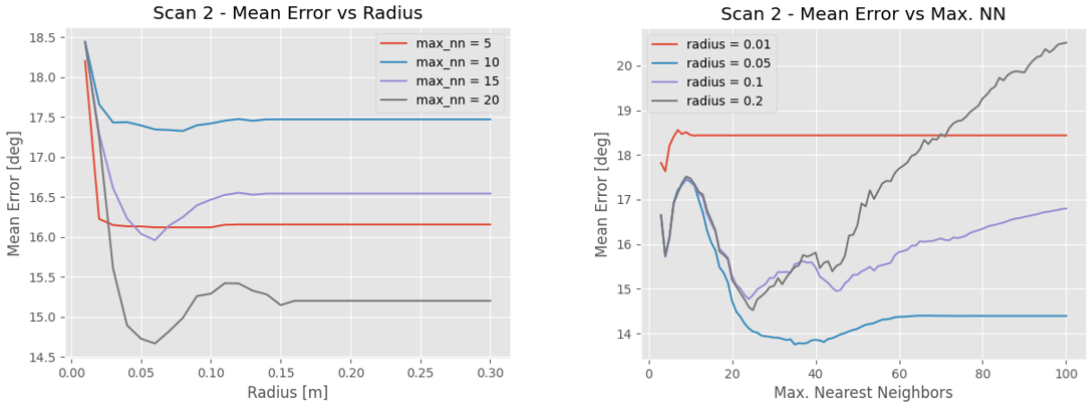

Figure 3.5 - Mean error plots for Scan 2 by varying r and max_{nn}

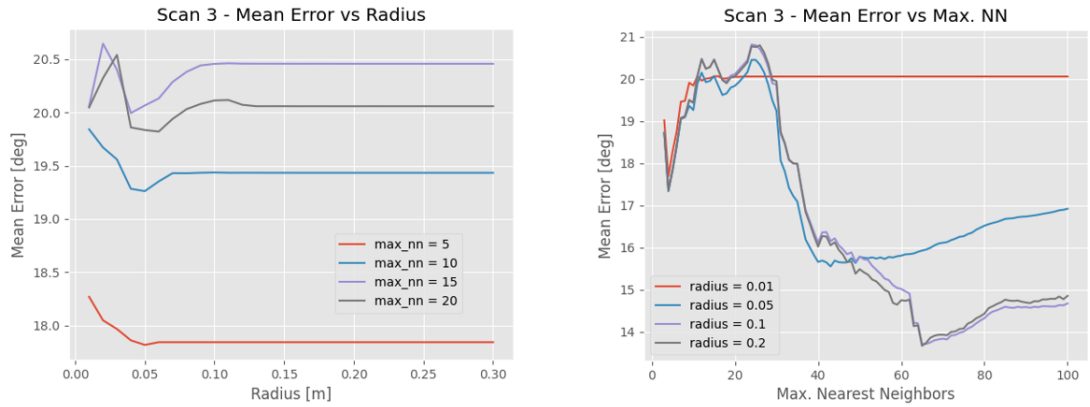

Figure 3.6 - Mean error plots for Scan 3 by varying r and max_{nn}

From these latest results, in fact, it can be seen that the mean error is much larger than in the first case. Once again, from the plots on the left, for a fixed max_{nn}, it can be observed that, from a certain r onwards, the error becomes constant, while from the plots on the right it is evident that, this time, as max_{nn} increases (r fixed), following an improvement in the error there is a worsening, except for the case $r = 0.01\ m$ where, again, the error becomes constant. It can clearly be deduced that the increase in error is based on the fact that, unlike the first scan analyzed, there is now a greater number of non-planar points where the normal is not well estimated.

Finally, the computational time for normal estimation has been measured. Tests have been performed for fixed r values while varying max_{nn}, on the scans indicated in this discussion as Scan 1 and Scan 2. The computational time plots are shown in Figure 3.7.

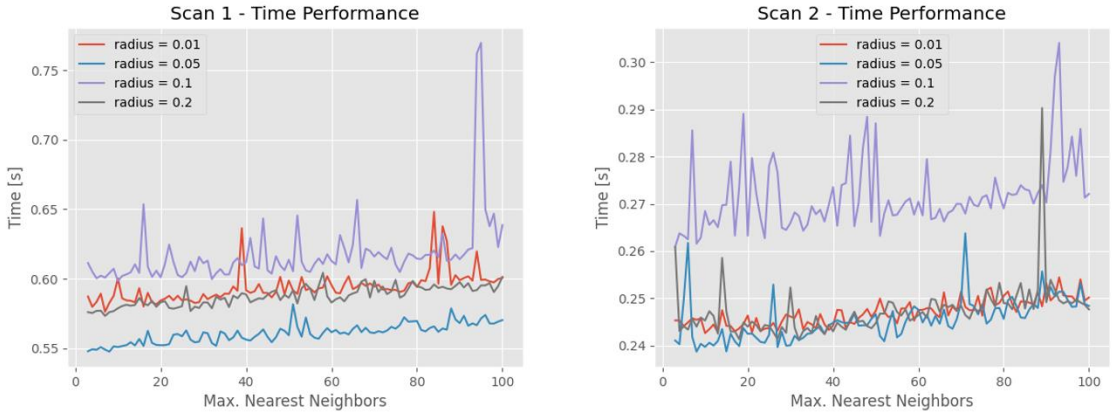

Figure 3.7 - Normal estimation time performance for Scan 1 and Scan 2

From the plots in Figure 3.7 it can be observed that the computational time increases slightly as max_{nn} increases. Furthermore, the reason why the computational time in the case of Scan 1 is much larger than that obtained for Scan 2 is simply related to the number of points of the point cloud, higher in the first case.

Following the analyses performed and the plots obtained, the optimal values of search radius r and maximum number of NN max_{nn} to be used for the estimation of the normals have been identified. The selected values are shown in Table 3.1 and will serve as a reference for normal estimation during the development of the algorithms.

Parameters	*Optimal range identified*	*Final value*
$\boldsymbol{r}$ $[\boldsymbol{m}]$	$0.05 \div 0.1$	0.07
$\boldsymbol{max_{nn}}$	$40 \div 50$	40

Table 3.1 - Parameters selection for normal estimation

3.2.2. FPFH Analysis

The analyses conducted on the estimation of normals are fundamental for the estimation of FPFH; in fact, given a query point, this descriptor is computed by exploiting the coordinates and normals of the points belonging to the neighborhood of the query point itself. In this Subsection, all the analyses that have been carried out on the FPFHs are explained.

The function used by Open3D for FPFH estimation [46] allows, given a point cloud as input, to compute a FPFH for each point of the point cloud. The descriptor is computed as a 33-dimensional vector. Therefore, the output of this function is a $33xN$ matrix, where N is the number of points of the point cloud. As for the estimation of normals, also this function requires the initialization of r and max_{nn} parameters.

3.2.2.1. FPFH Distance Histogram

All the analyses on the FPFHs have been conducted on the point cloud dataset already introduced in Section 3.1; in particular, the first tests are focused on understanding whether the FPFHs are actually able to discriminate a specific region of the satellite.

As input, the LiDAR point clouds with the ground truth of the poses and the model point cloud with the ground truth of the normals are used. Using this information, the normals of the LiDAR point cloud and the FPFHs of both point clouds are determined; then, for each FPFH of the LiDAR point cloud, the NN in the model point cloud is determined, thus creating correspondence pairs; then, the alignment is performed using the ground truth of the pose and, for each correspondence pair, the Euclidean distance between them is computed, thus obtaining a distance vector represented with a histogram.

Figure 3.8 shows, as an example, the histograms obtained for Scan 1 using the previously obtained r and max_{nn} parameters (i.e., $r = 0.07\ m$ and $max_{nn} = 40$) for normal estimation while, for FPFH estimation, $r = 0.01\ m$ (left) and $0.2\ m$ (right) are used, while max_{nn} in both cases is set to 100.

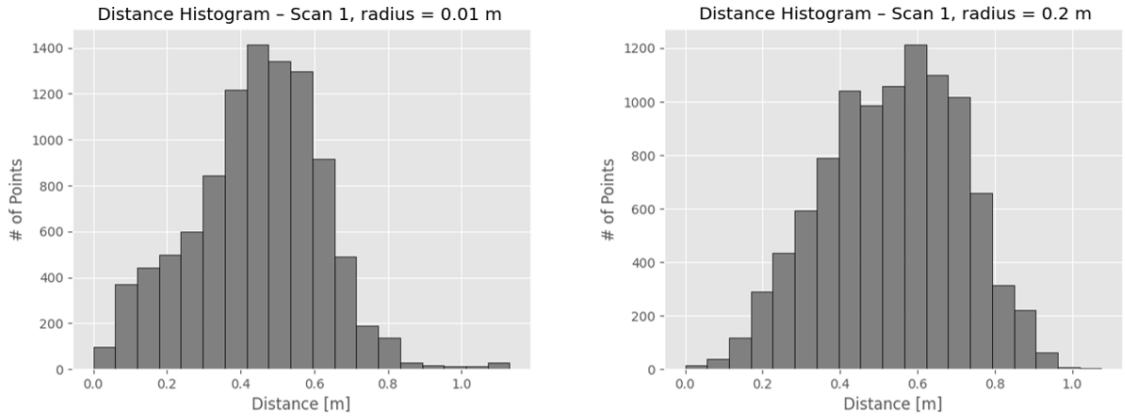

Figure 3.8 - FPFH distance histograms for Scan 1

The histograms shown in Figure 3.8 therefore group the correspondence pairs, obtained through the NN retrieval in the FPFH space, into distance bins representing the Euclidean distances between the pairs of points downstream of the alignment between the model and the measured point cloud. The ideal case would therefore be that in which all the pairs of points are placed in the first bin (zero distance), indicating that the FPFHs have allowed a perfect recognition of the points; but the situation seen in these plots is totally different: the most occupied bins are the central ones, almost forming a sort of Gaussian, and this answers the initial question. Given the symmetric geometry of the target satellite, there are many similar FPFHs even in different areas of the geometry, and therefore the FPFH identified as the most similar (through the NN retrieval) can easily be the FPFH of a point far from the one considered. Therefore, it is as if the association of the NN were random.

Therefore, to make the use of FPFHs on this geometry more effective, it is essential to reduce the number of points to act on; in particular, it is necessary to try to act on the points whose FPFH is *less common*, in order to reduce the risk of confusing it with the FPFH of other points. For this reason, Persistence Analysis has been studied in depth, and is discussed in more detail in the next paragraph.

3.2.2.2. Persistence Analysis

Persistence Analysis is a technique that consists of identifying, within the set of FPFH calculated for each point of the point cloud, the most *distinctive* ones. This allows to make a smaller, more solid set of candidate correspondences. This analysis is divided into two steps:

1. Search for *unique points*. The FPFH is computed for every point; then, the mean of the FPFHs of all the points in the cloud is computed (μ – histogram); so, the distance between the FPFH of each point and the μ – histogram is calculated. The distribution can be approximated by a Gaussian, and the points whose FPFH fall outside the $\mu \pm \beta\sigma$ interval, where μ and σ are respectively the mean and standard deviation of the above distribution, while β is a tuning parameter that controls the width of the interval, are called unique.
2. Search for *persistent points*. The previous step is repeated considering spheres of different radii r_i for the FPFH computation. The unique points as the radius varies are called persistent.

The set of persistent features is indicated in Equation 3.2:

$$\boldsymbol{P}_f = \bigcup_{i=1}^{n-1} [\boldsymbol{P}_{f_i} \cap \boldsymbol{P}_{f_{i+1}}] \tag{3.2}$$

where $\boldsymbol{P}_{f_i}$ are the *unique* features, while $\boldsymbol{P}_f$ is the set of *persistent* features given by the intersection of those features that are unique in both r_i and r_{i+1}. As a distance metric to compute the distance between the FPFH of each point and the μ – histogram, the Kullback-Leibler (KL) Divergence is used [20, 21].

The inputs are once again the LiDAR and model point clouds. After estimating the normal and FPFH for both the point clouds, Persistence Analysis has been carried out, looking for the unique and persistent features, after which, considering only the persistent features sets obtained from both the point clouds, the same procedure previously illustrated has been repeated for the construction of the histogram of distances (therefore, NN retrieval, alignment between the point clouds using the ground truth of the poses and computation of the Euclidean distances between the correspondences).

The results of the analyses carried out considering, as LiDAR point clouds, the scans previously indicated as Scan 1 and 2 are reported below. In particular, in Table 3.2, Figure 3.9 and Figure 3.10 the parameters set and the results obtained are shown respectively.

Point cloud	***Normal estimation parameters***		***FPFH estimation parameters***		***β***
	$r\ [m]$	max_{nn}	$r_i\ [m]$	max_{nn}	
LiDAR	0.07	40	0.05, 0.07, 0.1	100	1
Model	Ground Truth Normals		0.015, 0.02, 0.025	100	1

Table 3.2 - Normal and FPFH estimation parameters for both LiDAR and model point clouds

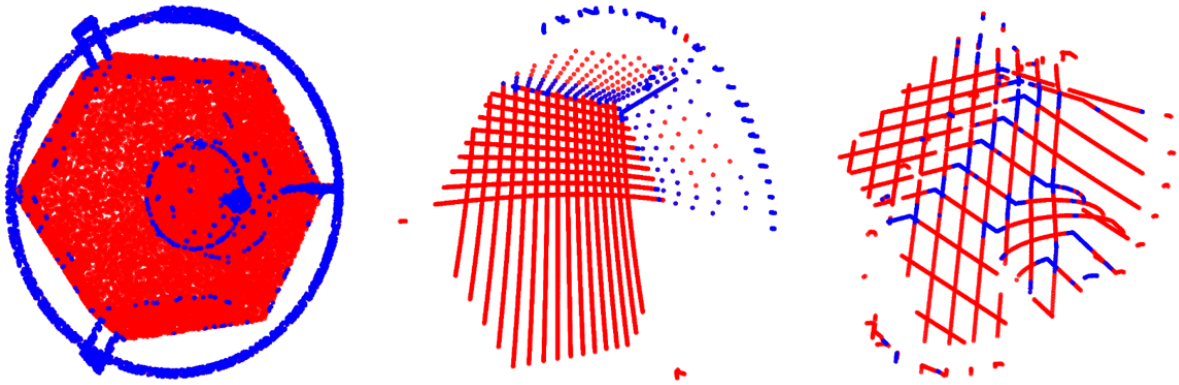

Figure 3.9 - Persistent points for model point cloud (left), Scan 1 (central) and Scan 2 (right)

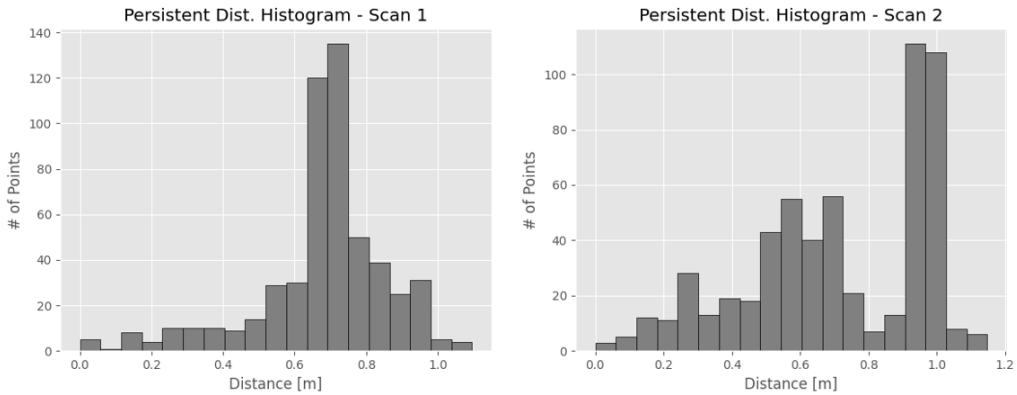

Figure 3.10 - Persistent FPFH distance histograms for Scan 1 and Scan 2

From the results it is clear that Persistence Analysis alone is not enough to improve the Distance Histograms: indeed, from Figure 3.10 a great confusion continues to be observed in identifying

correspondences between NN FPFHs. However, a very interesting result emerges from Figure 3.9: by observing the point clouds, in fact, it is clear that Persistence Analysis actually has an effect in detecting particular classes of points; the most evident advantage is that it seems to exclude quite effectively the points belonging to planar regions, selecting points on the edges, the toroid, the handles and the sphere at the top. Therefore, this technique helps to identify a reduced set of points that can be exploited to achieve an alignment between the model and the LiDAR point cloud, and in fact it is finally adopted in one of the global registration algorithms presented in this Thesis work: the PA-based RANSAC method, whose pipeline is shown in Sections 3.4 (offline phase) and 3.5 (online phase).

Downstream of PA, a further FPFH extraction technique has been investigated, which is based on the segmentation of the point cloud, dividing it into geometric primitives. This analysis is explained in detail in the next paragraph.

3.2.2.3. Geometric Primitives Recognition

The key idea of the analysis presented in this paragraph is to develop a feature extraction technique aimed at recognizing the FPFHs belonging to a specific geometry of the satellite: in other words, the satellite geometry is decomposed into simpler geometries and the FPFH of each of these geometries is analyzed.

To perform this study, the model point cloud is split into geometric primitives (toroid, edges, sphere, handles, cylinder, planes) and a *ground truth* of the mean FPFHs for each geometry is built; then, these reference FPFHs are compared with the mean FPFHs of each geometry extracted by the LiDAR point cloud dataset.

A *training dataset*, defined as the 80% of the 1000 scan input dataset, is used to generate a mean FPFH representative of each of the defined geometries. Naturally, the segmentation of the training scans is performed through the alignment with the model point cloud, using the pose ground truths. These FPFHs are finally compared with the corresponding mean FPFHs obtained from the model point cloud, to evaluate the similarity between the histograms. A good similarity between the histograms of a specific geometry is a symptom of the fact that, taking a

random LiDAR point cloud, the recognition of that geometry through FPFH estimation is possible.

Starting from the satellite CAD model, this is broken down in Blender environment into geometric primitives, from which dense point clouds are generated. The segmented point cloud of the satellite, with geometry labels, is shown in Figure 3.11.

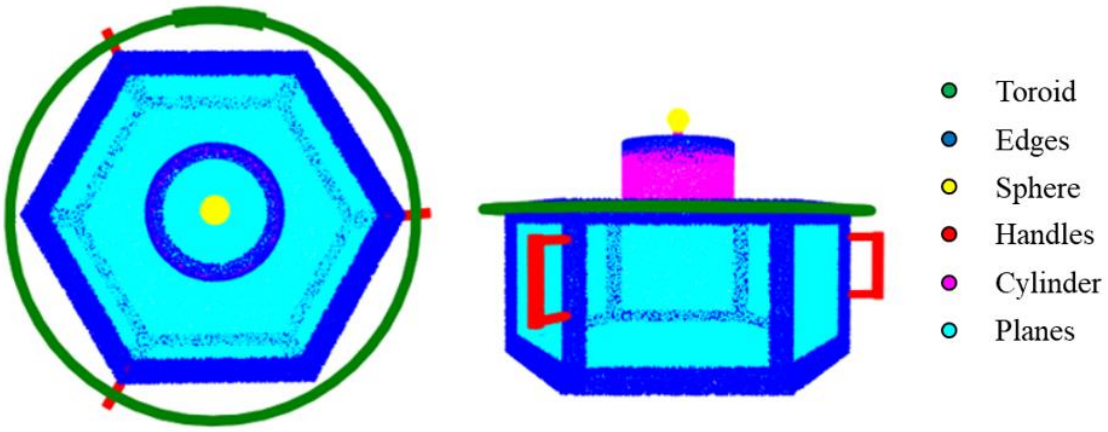

Figure 3.11 - Segmented point cloud

Then, for each point of these point clouds, the NN is searched among the points of the model point cloud, in order to reconstruct the point clouds of the geometric primitives using the same points of the model point cloud. This step is performed to then associate the ground truths of the normals of the model point cloud provided as input to the point clouds of the geometries. Once the ground truths of the normals are assigned to the point clouds, the mean FPFH for each of these geometries is computed.

Regarding the training dataset, instead, each scan is segmented using the point clouds of the geometries previously generated and the pose ground truths and, for each geometry, an FPFH averaged over all the training scans is computed. Figure 3.12, Figure 3.13 and Figure 3.14 show the histograms comparing model point cloud and training dataset.

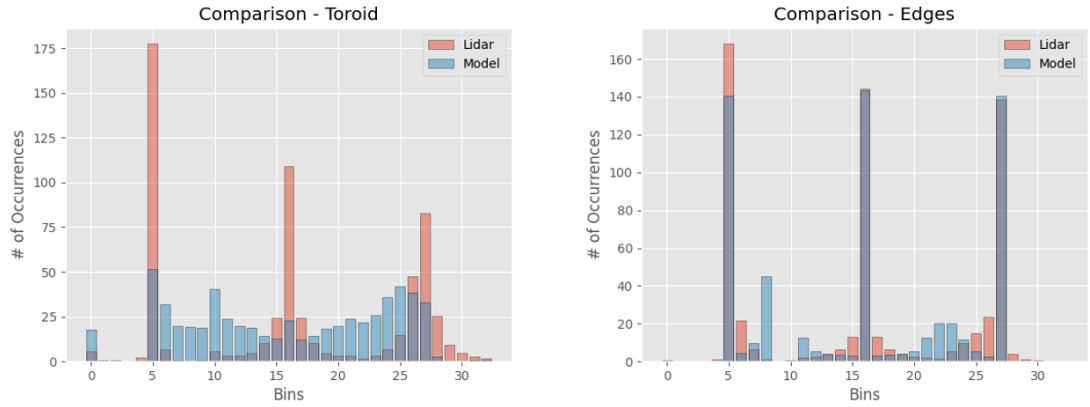

Figure 3.12 - Comparison mean FPFH toroid and edges

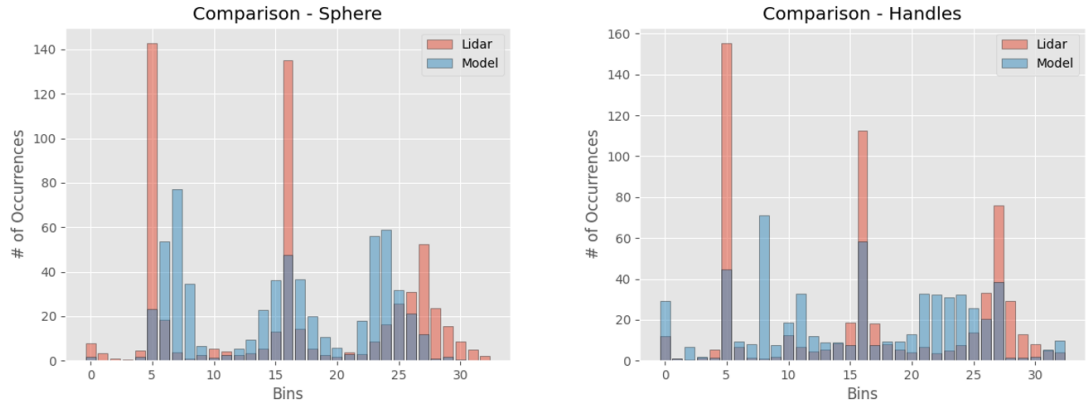

Figure 3.13 - Comparison mean FPFH sphere and handles

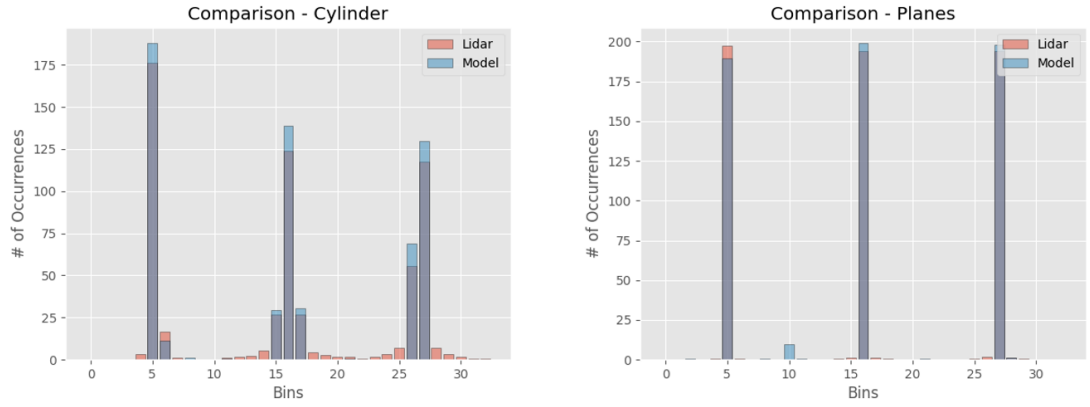

Figure 3.14 - Comparison mean FPFH cylinder and planes

From the results shown it is clear that the mean FPFHs obtained from the training dataset do not reproduce very well the trend of the corresponding mean FPFHs obtained from the model

point cloud, as expected, given the irregular distribution of points in the LiDAR point clouds. However, for some geometries there are interesting similarities: excluding planes, which do not represent the best geometry to exploit for the alignment of the point clouds, edges and cylinder resulted the most interesting geometries to rely on for the pose estimation algorithm.

After identifying the most promising geometries, the corresponding FPFH-based signatures averaged over the training dataset have been taken as a reference to perform an optimization analysis on the extraction of the FPFH from random LiDAR point clouds, selecting a priori the geometry of interest and trying to extract from the point cloud the FPFHs corresponding to that geometry. This analysis has been carried out considering three geometries: in addition to the cylinder and the edges, i.e., the most interesting geometries from the latest analyses shown, the handles have also been included. Therefore, starting from the FPFH-based signatures of these geometries, 100 LiDAR point clouds have been randomly selected and a specific FPFH extraction procedure has been performed, which is explained below.

This procedure consists of 2 phases:

1. *k-NN retrieval*. The FPFHs calculated for each point of the measured point cloud are organized according to a KDTree structure. For each class, the k_{FPFH} NNs to the corresponding FPFH-based signature are found. While this approach would result in $2k_{FPFH}$ points composing the set of candidates $\boldsymbol{K}$, an additional filter is applied to keep only the points with distance from the FPFH-based signature of that class smaller than a threshold (d_{FPFH}).
2. *Reciprocity Test*. All the candidate points from the previous steps are compared to the FPFH-based signatures of all the geometry classes identified for the target satellite, to check that the minimum FPFH-based distance is relative to the identified class. This check avoids selecting candidate points from different geometrical elements, especially in cases in which points belonging to the identified class are not in the sensor field of view.

In the previous explanation of the FPFH extraction procedure, some parameters have been mentioned that are currently unknown, which are the number k_{FPFH} of NNs and the distance threshold d_{FPFH} in the FPFH space: the best values of these parameters are obtained

experimentally through an optimization analysis based on the construction of the *Precision-Recall curves*.

An example is presented below to understand the purpose of this analysis. If the procedure explained above is used to extract for example the points belonging to the cylinder class, the cases shown in Figure 3.15 may occur:

1. *True Positives* T_P. The extracted points actually belong to the cylinder class;
2. *True Negatives* T_N. The non-extracted points actually do not belong to the cylinder class;
3. *False Positives* F_P. The extracted points do not belong to the cylinder class;
4. *False Negatives* F_N. The non-extracted points belong to the cylinder class.

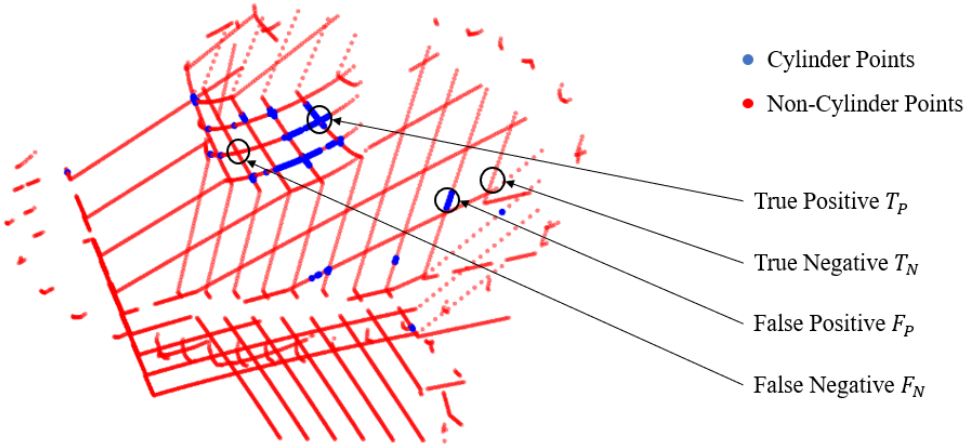

Figure 3.15 - Example of cylinder-points detection and terminology used

The Precision and Recall parameters are based on these definitions. In information retrieval, *precision* is a measure of result relevancy, while *recall* is a measure of how many truly relevant results are returned. They are defined as shown in Equations 3.3 and 3.4.

$$P = \frac{T_P}{T_P + F_P} \tag{3.3}$$

$$R = \frac{T_P}{T_P + F_N} \tag{3.4}$$

Therefore, P indicates a measure of the number of candidates extracted correctly compared to the total number of candidates extracted, while R is a measure of the number of candidates

extracted correctly compared to the actual number of points of interest. So, for each scan a Precision and Recall value are obtained for each geometry. These parameters are averaged over 100 random training scans; furthermore, these parameters have been computed in one case by varying d_{FPFH}, with no constraint on k_{FPFH}, while in another case by varying k_{FPFH}, with no constraint on d_{FPFH}. The settings of these parameters for the two cases are reported in Table 3.3.

Precision-Recall parameters	***Case 1***	***Case 2***
N° scans	100	100
$\boldsymbol{d_{FPFH}}$	[30, 35, 40, …, 400]	No Constraint
$\boldsymbol{k_{FPFH}}$	No Constraint	[10, 15, 20, …, 400]

Table 3.3 - Precision-Recall analyses: Cases 1 and 2 parameter setting

What is expected is that, by relaxing the distance constraint, increasing the threshold, or increasing the number k_{FPFH} of NNs to be identified, the Recall increases, as it increases the percentage of extracted points that actually belong to the geometry of interest; at the same time, however, a decrease in Precision may occur because, being less restrictive in the selection of candidates, in addition to increasing the number of correctly extracted points, it also increases the number of incorrectly extracted points, not actually belonging to that geometry. Therefore, the goal of this analysis is to find the optimal parameters of k_{FPFH} and d_{FPFH} so that a good compromise between Precision and Recall is found. The Precision-Recall curves for cylinder, edges, and handles are shown in Figure 3.16, Figure 3.17 and Figure 3.18.

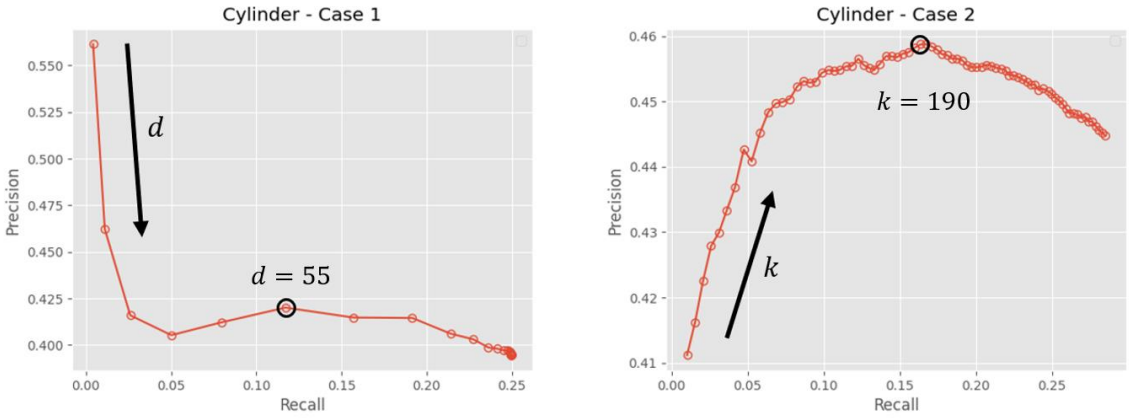

Figure 3.16 - Precision-Recall curves for cylinder geometry: cases 1 and 2

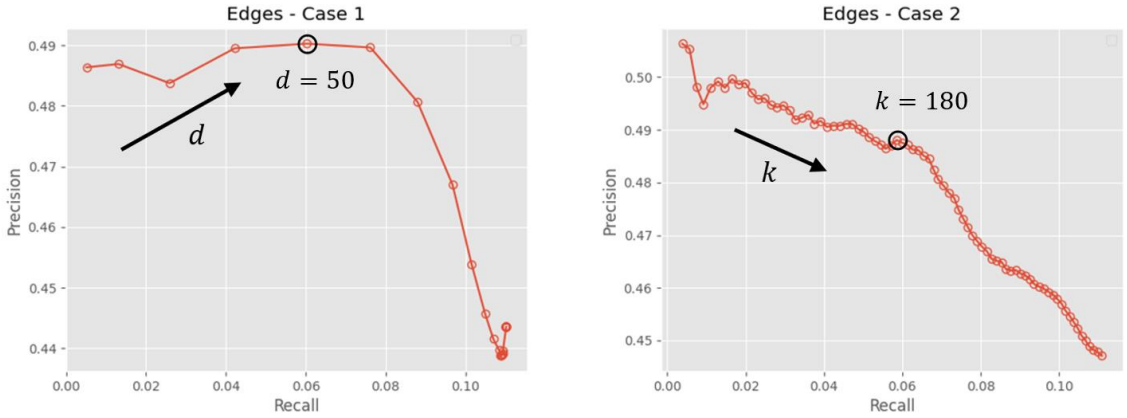

Figure 3.17 - Precision-Recall curves for edges geometry: cases 1 and 2

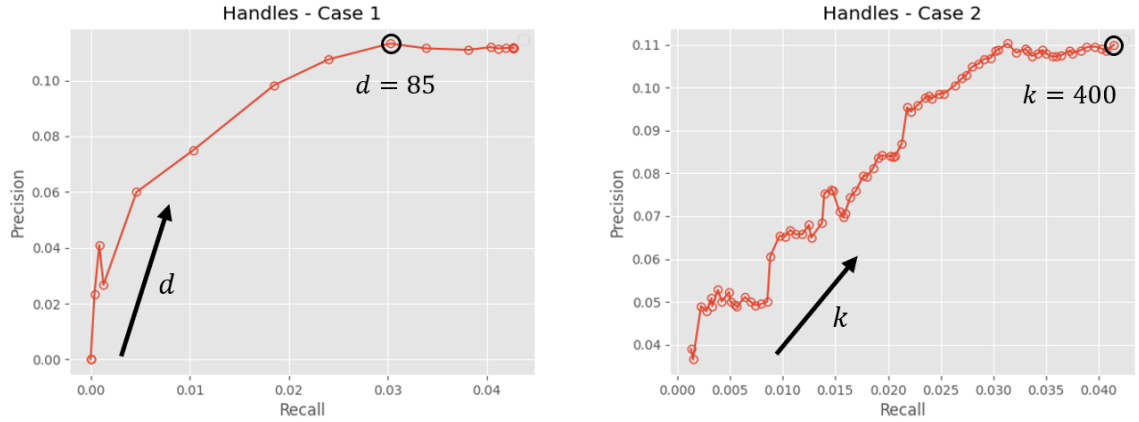

Figure 3.18 - Precision-Recall curves for handles geometry: cases 1 and 2

From these plots some very important results can be drawn. First of all, the observations made upstream of the analyses have been verified: in fact, by increasing d_{FPFH} (Case 1) or k_{FPFH} (Case 2), an increase in Recall at the expense of Precision is observed, especially for the edges; furthermore, the values of Recall and Precision for cylinder and edges are much more promising than those obtained for the handles, where in particular a rather strange trend can be seen: starting from very few detected points, by relaxing the constraints on the identification of the candidates, in addition to increasing the Recall, Precision also increases, even if by very little. The comparison plots between the Precision-Recall curves of the different geometries analyzed are better shown in Figure 3.19.

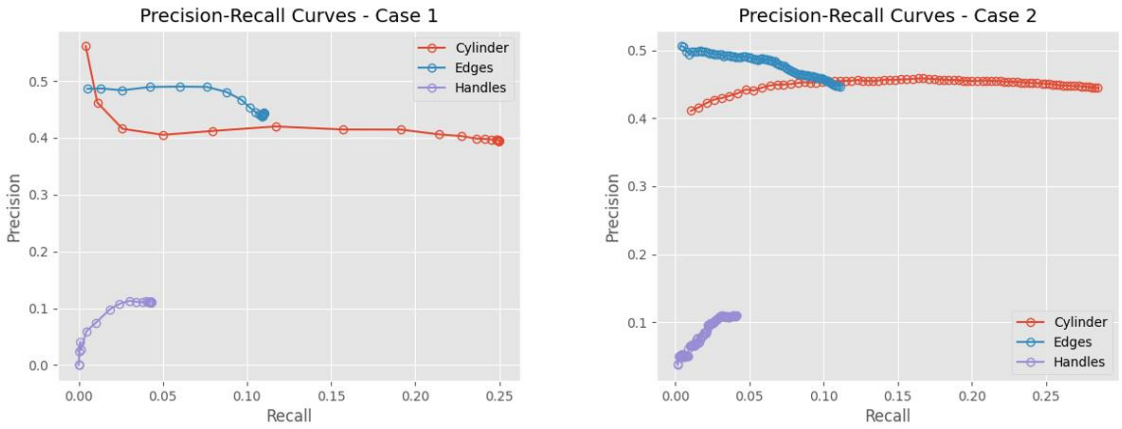

Figure 3.19 - Precision-Recall curves comparison between the geometries analyzed

From Figure 3.19 it is very clear that, unfortunately, handles are not such good geometries to rely on, unlike cylinders and edges.

Therefore, from the Precision and Recall analyses, optimal values of d_{FPFH} and k_{FPFH} emerged to be used to effectively filter the FPFHs belonging to the geometry of interest. These values are shown in Table 3.4, reporting only the cases of cylinder and edges, given the relatively less satisfactory results of the handles.

Filtering parameters	***Cylinder***	***Edges***
$\boldsymbol{d_{FPFH}}$	*55*	50
$\boldsymbol{k_{FPFH}}$	190	180

Table 3.4 - Filtering parameters resulting from Precision-Recall analysis

The use of FPFH-based signatures representative of the above-mentioned geometries is the underlying principle of another of the global registration algorithms presented in this Thesis work: the Label-based RANSAC method, whose pipeline is shown in Sections 3.4 (offline phase) and 3.5 (online phase). The procedure illustrated in this Subsection has been implemented in this global registration algorithm for the extraction of the points belonging to the cylinder and edges classes from the point cloud acquired online, using the tuning parameter values resulting from this study.

3.3. Hash Tables

The underlying principle of the global registration algorithms presented in this thesis work is the following: the pose is estimated through the alignment between two sets of points, a set expressed in the SRF and a set expressed in the TRF. The set of SRF points is extracted from the point cloud measured online, while the set of corresponding TRF points is searched within Hash Tables appropriately built offline. Hash Tables are data structures built for the purpose of storing data in a way that makes their retrieval quick and efficient.

A hash table stores a certain number of *values* by matching them with *keys* that contain information about the related value. Specifically, given a key, it is transformed, through an appropriate *hash function*, into an integer, called *hash index* or *hash code*, which identifies the cell of the hash table, called *bucket*, in which the associated value will be stored. A simplified diagram describing the working principle of a hash table is shown in Figure 3.20.

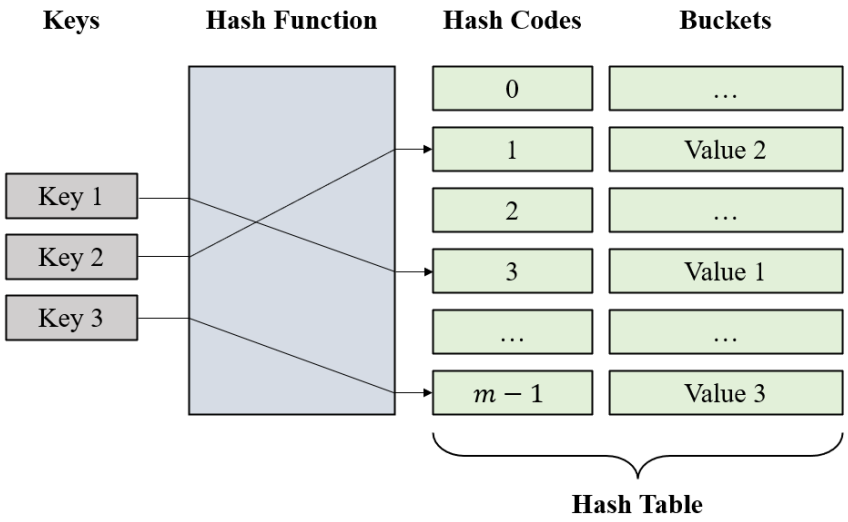

Figure 3.20 - Working principle of a hash table

A hash function converts the data to be indexed into an integer between 0 and $m - 1$; therefore, the hash table is organized into m buckets, each of which is identified by a specific hash code. The choice of the hash function strongly affects the construction of a hash table: an ideal hash function should be an injective function, i.e., a function that transforms different keys into different hash codes; however, being an ideal case, the adopted hash functions do not satisfy this property and, therefore, cases occur in which different keys are transformed into the same

hash code, thus generating collisions. There are several techniques to manage collisions; the most used are:

- *Open addressing*, in which, in case of collisions, the first empty bucket is searched to store the value;
- *Separate chaining*, in which, in case of collisions, the value is inserted into that same bucket, creating a linked list.

In the case of interest, hash tables are built by storing triplets/pairs of points in buckets localized by specific hash codes depending on the geometric properties they exhibit. In particular, a similar approach to separate chaining is used to handle collisions, since in each bucket lists containing sets of points are stored: since the geometric information of the sets of points to be stored is exploited to generate the corresponding keys, the sets stored in the same bucket are characterized by similar geometric properties, and this will be an advantage for determining online the correspondences for alignment. A simplified scheme is shown in Figure 3.21.

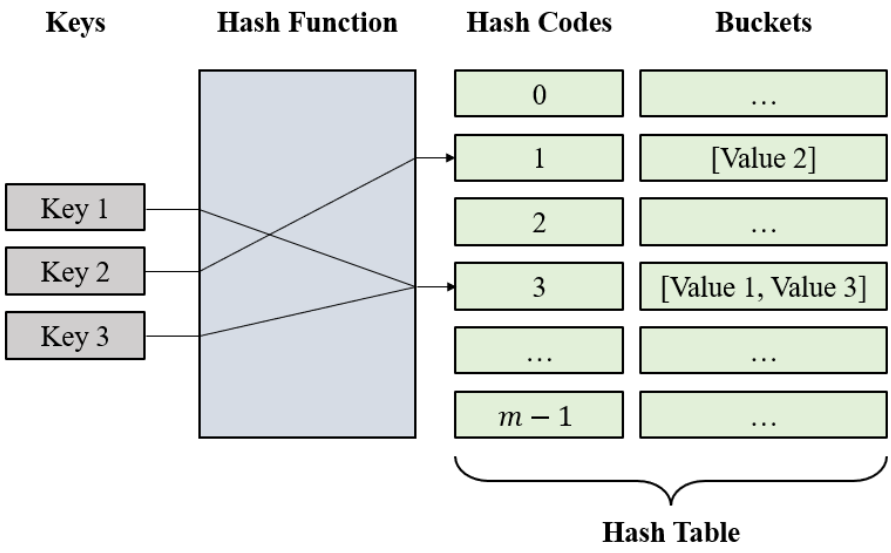

Figure 3.21 - Collision handling

The following subsections describe in detail the hash table construction methodologies used in the algorithms, showing the different variants tested.

3.3.1. Training Hash Tables

This subsection describes the construction methods of the Hash Tables (HTs) used in the final algorithms. These HTs, as the name suggests, are built by exploiting the scans of a training dataset rather than the complete model point cloud (variant presented in Subsection 3.3.2) since, by taking the sets of points to be stored - triplets or surflets, depending on the method considered - from LiDAR scans, all the stored sets are possible sets, i.e., observable by the sensor; instead, by randomly taking point triplets from a complete point cloud, there is the possibility that many stored point sets are not realizable. The following paragraphs describe the various training HTs implemented, built after a preliminary classification of the training scans, as explained in paragraph 3.3.1.1. The values of the HT construction tuning parameters, defined in the next paragraphs, are shown in paragraph 5.2.1.1.

3.3.1.1. Point Cloud Classifier

In Subsection 3.2.2, two main techniques for extracting FPFHs based on particular properties have been analyzed, in particular *Persistence Analysis*, which is based on the selection of the FPFHs furthest from the mean FPFH of the point cloud, the underlying principle of the *PA-based RANSAC method*, and *Point Cloud Segmentation*, with the aim of determining FPFH-based signatures for the identification of points belonging to a certain geometry, the underlying principle of the *Label-based RANSAC method*. As already mentioned at the beginning of Chapter 3, a third algorithm has been developed during this thesis work, which is based on the search for *surflets* with similar properties, named *PPF-based RANSAC method*. However, the adopted features and extraction techniques may present problems depending on the acquired scan: for example, if the acquired LiDAR point cloud is a scan that is made up of a single plane, the selection of the points belonging to the cylinder clearly fails, since that part is not visible in the scan, while the selection of the points belonging to the edges could easily fail, since the extraction algorithm does not recognize "sharp" areas given by the intersection of two planes. Therefore, the technique adopted by the Label-based RANSAC method is not at all suitable for this type of scan, as well as the Persistence Analysis, since the FPFH of the points of the considered scan will all be very similar. For the same reason, the search for surflets is not efficient in this case either. Therefore, in these algorithms a classification of the point clouds of the training dataset is first carried out by a *Point Cloud Classifier* block, distinguishing between

Complex Structure (CS) and Flat Structure (FS) scans. Specifically, while for CS scans, point and local normal information can be used to describe the shape of the target surface, FS scans do not experience local normal variations; therefore, specific HTs are created to separately handle these two types of scans efficiently. Clearly, in a similar way, the point cloud classifier shall be applied to the measured point cloud in the online phase to decide which HT must be used for pose estimation. The distinction between a CS scan and an FS scan is performed by evaluating the variance of the set of normal unit vectors computed over the entire point cloud. A low variance indicates aligned directions of the local normal unit vectors, therefore a geometrically simple point cloud. Specifically, the parameter used to distinguish between these scans is the *sum of the variances* of the normal components τ_{σ_N}, which is compared to a threshold value $\tau_{\sigma_{N,thre}}$: if $\tau_{\sigma_N} < \tau_{\sigma_{N,thre}}$, the point cloud is classified as an FS scan, otherwise as a CS scan. As an example, two cases of CS scan and FS scan are shown in Figure 3.22 and Figure 3.23, respectively.

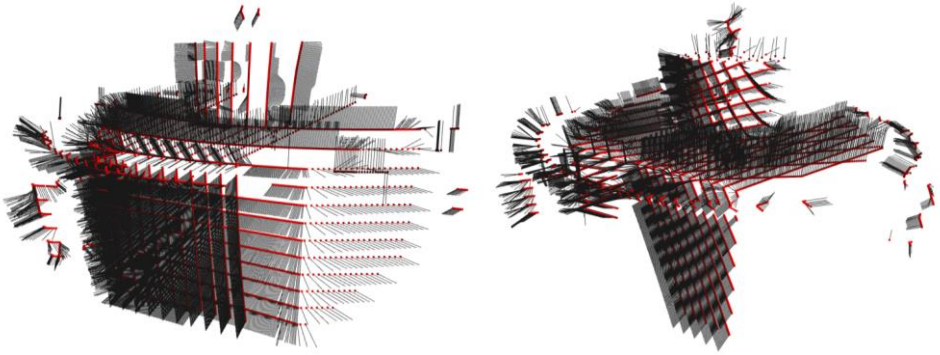

Figure 3.22 - Two examples of CS scan normal distribution

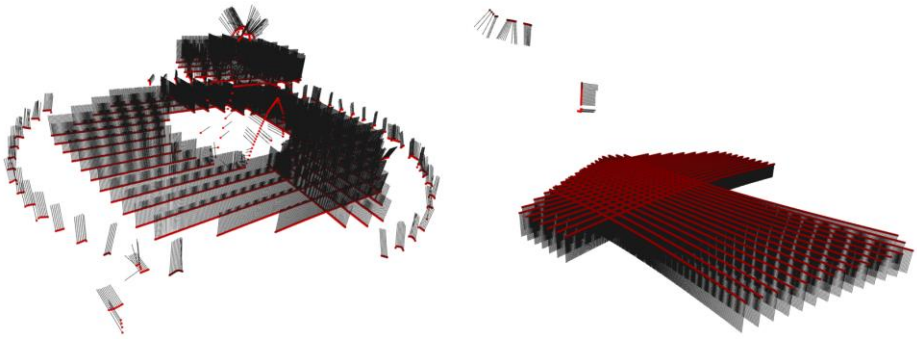

Figure 3.23 - Two examples of FS scan normal distribution

The point clouds shown in Figure 3.22 are CS scans, since cylinder and/or edges can be detected and, in general, the local geometry of the point clouds is quite varied (different FPFHs). The same cannot be said for the point clouds shown in Figure 3.23: the left scan is mostly made up of planar points, and edges are hard to detect, since there are no intersections between two planes; in the right scan the problem is much more evident, since only one plane is visible, the cylinder is not present at all, and the related FPFHs are expected to be all similar. The most important information to distinguish these two classes of point clouds is contained in the normals, whose variance is clearly lower in the case of the FS scans, since most of the normals are aligned with each other; therefore, as a parameter to distinguish the scans, the sum of the variances of the components of the normals calculated at each point of the scans is chosen: if this value falls below a certain threshold $\tau_{\sigma_{N,thre}}$, selected after experimental tests, the scan is classified as a FS scan, otherwise it is classified as a CS scan. Table 3.5 shows, as an example, the values of these parameters for the point clouds shown in Figure 3.22 and Figure 3.23.

τ_{σ_N} ***threshold***	***Left point cloud***	***Right point cloud***
CS scan (Figure 3.22)	0.39	0.51
FS scan (Figure 3.23)	0.04	0.004

Table 3.5 - Sum of variances of normal components of point clouds shown as an example

As already mentioned, given the need for a classification between the point clouds, in order to adopt a certain solution depending on whether the scan considered is CS or FS, specific HTs have been created for both cases; in particular, as shown in the next paragraphs, a HT for CS scans $\boldsymbol{HT_{CS}}$ has been built for each of the three methods mentioned above (Label-based, PA-based and PPF-based RANSAC), using sets of points (triplets or surflet pairs) extracted from the CS scans of the training dataset, and a HT for FS scans $\boldsymbol{HT_{FS}}$ common to all three algorithms, built using instead triplets of points extracted from the FS scans.

3.3.1.2. Hash Table Construction for CS Scans in Label-Based RANSAC

The $\boldsymbol{HT_{CS}}$ used for the Label-based RANSAC method is a two-level HT that associates two hash codes to the stored point triplets, which are the cylinder and edge points, precisely extracted from the CS scans of the training dataset and converted to TRF using the knowledge

of the ground truth pose parameters. These hash codes are defined by the corresponding hash keys as follows:

1. The first hash code is a triplet of integer numbers, whose value is set using as key, $\boldsymbol{g}_{key}$, the label of points; specifically, the values 0 and 1 respectively correspond to edge and cylinder points (e.g., $\boldsymbol{g}_{key} = [cylinder, cylinder, edge] \rightarrow \boldsymbol{g}_{hash} = [1, 1, 0]$).
2. The second hash code is a triplet of integer numbers, $\boldsymbol{d}_{hash}$. The values of $\boldsymbol{d}_{hash}$ are obtained using as keys the inter-point distances of the considered point triplet, $\boldsymbol{d}_{key}$. Specifically, the hash function converting the key into the corresponding code is shown in Equation 3.5, where m is the number of buckets composing the HT and D is the maximum distance between pairs of points (i.e. the target satellite maximum dimension) and, finally, the floor function returns the largest integer less than or equal to the argument, so as to have a triplet of integers as hash code.

$$\boldsymbol{d}_{hash}(i) = floor\left(\frac{m * \boldsymbol{d}_{key}(i)}{D}\right), \qquad i = 1,2,3 \tag{3.5}$$

The value assigned to m strongly affects the number of triplets stored in a single bucket, since it determines the resolution with which point distances are sampled. Therefore, it plays a very important role in the online pose estimation process: a high m results in a finer classification of point triplets, given the high sampling resolution of the inter-point distances, thus getting a low number of triplets in a single bucket, but with a good chance that these are good matches for alignment, and vice versa in the case of low m. Therefore, choosing an appropriate m is crucial to have a good balance between computational efficiency and accuracy.

Once the candidate points to be stored have been obtained, the following procedure for filling the HT is adopted, which consists in a sequence of operations performed for a predetermined number of cycles (n_{cycles}). At each cycle, a triplet of points is randomly extracted and the three inter-point distances are computed: if these three distances are all larger than a threshold (d_{thre}), i.e., the condition in Equation 3.6 is met, the triplet is kept, otherwise it is discarded. Indeed, this approach is essential to avoid storing triplets of points that are excessively close to each other, which could make the subsequent pose determination inefficient.

$$\boldsymbol{d}(i) > d_{thre}, \qquad i = 1,2,3 \tag{3.6}$$

For all the randomly generated triplets that pass this check, the tuple of points is stored in the HT, in terms of the two above-defined hash codes, three times in order to account for any possible order of points in the triplet. This operation is used to avoid the case in which, after extracting the triplet in the online phase, its correspondent is not found because it is saved in the HT considering a different order.

3.3.1.3. Hash Table Construction for CS Scans in PA-Based RANSAC

The $\boldsymbol{HT_{CS}}$ used for the PA-based RANSAC method has a single level that associates a hash code to each of the stored point triplets. The key used is the triplet of distances between point pairs of the triplet, exactly as for the second level of the Label-based RANSAC $\boldsymbol{HT_{CS}}$, and the hash function used is again given by Equation 3.5.

As in the previous case, the CS scans of the training dataset, expressed in TRF, are first extracted. Then, Persistence Analysis is applied to the resulting dataset to find the set of persistent points to be used for HT construction. The tuning parameters adopted to apply the Persistence Analysis are the same as those shown in Table 3.2 for the LiDAR point cloud case.

The procedure for filling the HT is very similar to the one used for the Label-based RANSAC method, with the only difference that now the point triplets are associated with one single hash code $\boldsymbol{d}_{hash}$.

3.3.1.4. Hash Table Construction for CS Scans in PPF-Based RANSAC

Before describing the $\boldsymbol{HT_{CS}}$ construction procedure for PPF-based RANSAC method, it is necessary to introduce an additional reference system, used to obtain the hash keys to be associated with the surflets, called Local Reference Frame (LRF). The unit vectors corresponding to the LRF axes are defined as in Equation 3.7:

$$\hat{\boldsymbol{u}} = \frac{\boldsymbol{n}_1}{\|\boldsymbol{n}_1\|}, \qquad \hat{\boldsymbol{v}} = \frac{\boldsymbol{n}_1 \times \boldsymbol{n}_2}{\|\boldsymbol{n}_1 \times \boldsymbol{n}_2\|}, \qquad \hat{\boldsymbol{w}} = \hat{\boldsymbol{u}} \times \hat{\boldsymbol{v}} \tag{3.7}$$

where $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ represent the normals in the first and second point of the surflet, respectively. The LRF is assumed to be centered at the first point of the surflet. Instead, the SRF is arbitrarily chosen. A simplified representation of the surflet and the defined LRF are shown in Figure 3.24.

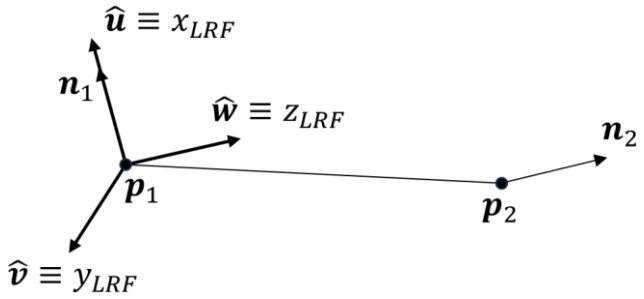

Figure 3.24 - Graphical representation of a surflet and the defined LRF

The $\boldsymbol{HT_{CS}}$ built for the PPF-based RANSAC method has a single level that associates a hash code to each of the stored surflets. So, unlike the two $\boldsymbol{HT_{CS}}$ built for Label-based RANSAC and PA-based RANSAC algorithms, this one does not rely on point triples, but on surflets associated with a 4D key containing three distance information and one angular information:

1. The three distance informations are the three coordinates of the relative position vector between $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, expressed in LRF ($\boldsymbol{d}_{LRF}$);
2. The angular information is the angle α between the two normals.

Since the relative distance vector in SRF and the rotation matrix from SRF to LRF are known, the relative distance vector is simply transformed into the LRF. The formulas shown in Equation 3.8 and Equation 3.9 are used:

$$\boldsymbol{R}_{SRF \to LRF} = [\hat{\boldsymbol{u}} \quad \hat{\boldsymbol{v}} \quad \hat{\boldsymbol{w}}]^T \tag{3.8}$$

$$\boldsymbol{d}_{LRF} = \boldsymbol{R}_{SRF \to LRF}\, \boldsymbol{d}_{SRF} \tag{3.9}$$

Once the key $\boldsymbol{s}_{key} = [\boldsymbol{d}_{LRF}; \alpha]$ is defined, the 4D hash code is generated using the hash function reported in Equation 3.10 and Equation 3.11:

$$\boldsymbol{s}_{hash}(i) = floor\left(\frac{m*\boldsymbol{d}_{LRF}(i)}{D}\right), \qquad i = 1,2,3 \tag{3.10}$$

$$\boldsymbol{s}_{hash}(i) = floor\left(\frac{m_a*\alpha}{A}\right), \qquad i = 4 \tag{3.11}$$

where m is the number of buckets for the coordinates, D is the maximum distance between point pairs and $\boldsymbol{d}_{LRF}$ is the tuple of the three coordinates previously defined, while m_a is the number of buckets for the angles, α is the angle between the respective normals and A is the upper bound of the allowed angles. Surflets are generated from the CS scans of the training dataset, expressed in TRF. To build this $\boldsymbol{HT_{CS}}$, the following sequence of operations is performed for n_{cycles}: one of these scans is randomly selected and a surflet is randomly taken. At this point, a condition is imposed on the angle α between the surflet normals, which must not be close to either 0° or 180°. The main reason behind this condition is the fact that, in the extreme cases where α was 0° or 180°, the normals would be aligned, and an LRF defined as above could not be constructed: in fact, the unit vector $\hat{v}$ is computed as the cross product between the two normals of the surflet, which would be zero if the normals were parallel; moreover, this generally represents an unfavourable condition for recognition: an α around 90° would be preferable, for example, indicating that the two points in the surflet belong to two orthogonal planes and not to the same plane. In paragraph 5.2.1.1, the boundaries α_{min} and α_{max} within which the angle is to be contained are defined. If the angle condition is not satisfied, the cycle starts again; if it is satisfied, another condition is set on the Euclidean distance d between the two points of the surflet, as shown in Equation 3.12:

$$d > d_{thre} \tag{3.12}$$

If this condition is also satisfied, the three coordinates of the distance vector $\boldsymbol{d}_{LRF}$ are defined. Once the key $\boldsymbol{s}_{key}$ is obtained, the hash code $\boldsymbol{s}_{hash}$ is generated through Equation 3.10 and Equation 3.11 and, finally, the surflet is stored in the bucket localized by the computed hash code, two times in order to account for the order of points and normals in the surflet.

It is very important to note that, for the PPF-based RANSAC case, no feature extraction technique is performed, unlike what has been seen for the Label-based RANSAC algorithm and for the PA-based RANSAC one.

3.3.1.5. Hash Table Construction for FS Scans

As already mentioned, in addition to the $\boldsymbol{HT_{CS}}$, built for each of the presented methods, a further HT has been built, $\boldsymbol{HT_{FS}}$, common to all three methods, using exclusively the FS scans of the training dataset. It is built with the following procedure. All the FS scans are expressed in TRF using the corresponding ground truth pose and merged into a single set. Therefore, the same approach described for the generation of Label-based and PA-based RANSAC $\boldsymbol{HT_{CS}}$ is adopted, thus storing triplets of points. The only difference is that a triplet is only identified by the second hash code defined according to Equation 3.5. Given the aforementioned problems in handling surflets with parallel normals, mentioned in paragraph 3.3.1.4, even for the PPF-based RANSAC method the FS scan case is managed using point triplets rather than surflet pairs.

3.3.2. Model Hash Tables

In addition to the $\boldsymbol{HT_{CS}}$ training HTs, further variants have been developed that, instead of using training scans, use the complete model point cloud. Such HTs have been generated only for the Label-based and PA-based RANSAC methods and they also store triplets according to the same procedure, the same keys and the same hash functions, with the only following difference:

1. For the Label-based RANSAC method, triplets are randomly selected from the set of points belonging to the cylinder and the edges of the entire model point cloud;
2. For the PA-based RANSAC method, instead, in paragraph 3.2.2.2. it has been shown that Persistence Analysis tends to extract points belonging to the most particular geometries, such as the toroid, the handles, the edges, excluding planar points quite effectively. Therefore, triplets are randomly extracted from the point cloud model, excluding planes.

Simulations have been performed using both $\boldsymbol{HT_{CS}}$ construction approaches (training and model HTs) for Label and PA-based RANSAC algorithms. In paragraph 5.2.2.1, a comparison between the two HT construction methodologies is reported. Again, information about tuning parameters is given in paragraph 5.2.1.1.

An alternative to fixing the number of cycles n_{cycles} for the construction of the training and model HTs could have been to fix the number of sets of points (triplets/surflets) to be stored and repeat the cycle until the preset number is reached, in such a way as to have HTs whose size is not linked to the randomness with which the triplets/surflets are taken, since not all of them are stored, as seen in the previous sections, but only those that exceed the set distance thresholds.

3.4. Overview of the Offline Phase of the Algorithms

In this Section, an overview of the Offline phase of the developed algorithms is shown. For pre-processing operations, the inputs used are the model point cloud, the training dataset, defined as the 80% of the 1000 scans dataset, as mentioned in paragraph 3.2.2.3, and, in the case of Label-based RANSAC, the target primitives point clouds.

3.4.1. Label-Based RANSAC

Figure 3.25 shows a block diagram detailing the offline pre-processing phase of the Label-based RANSAC method.

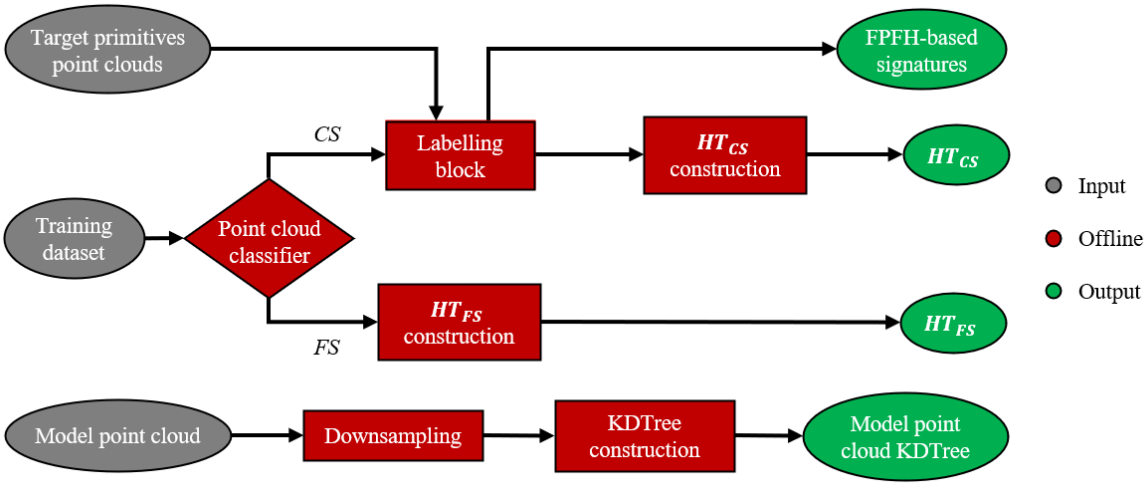

Figure 3.25 - Label-based RANSAC offline block diagram

From the figure it can be observed that:

1. Given as input the point clouds of the geometric primitives *(target primitives point clouds)*, obtained, as explained in paragraph 3.2.2.3, by subdividing the satellite CAD model in Blender environment and generating dense point clouds from them, these enter a *Labelling block*, whose outputs are both the FPFH-based signatures, i.e. the 6 FPFH, each representing a geometry, averaged over the entire training dataset, and the "exact" cylinder and edge points from the scans of the training dataset, to build the $\boldsymbol{HT_{CS}}$;
2. Given as input the training dataset, this is used to build $\boldsymbol{HT_{CS}}$ (paragraph 3.3.1.2) and $\boldsymbol{HT_{FS}}$ (paragraph 3.3.1.5);
3. Given as input the model point cloud, it is downsampled, with a voxel size of $0.025\ m$, and is used for the construction of a KDTree of the point cloud, which will be fundamental in the online phase for the evaluation of the alignment quality. Downsampling is applied because it allows a reduction of the computational time in performing such operation. Of course, for this operation, also the point cloud measured online is downsampled with the same voxel size.

3.4.2. PA-Based RANSAC

Figure 3.26 shows a block diagram detailing the offline pre-processing phase of the PA-based RANSAC method.

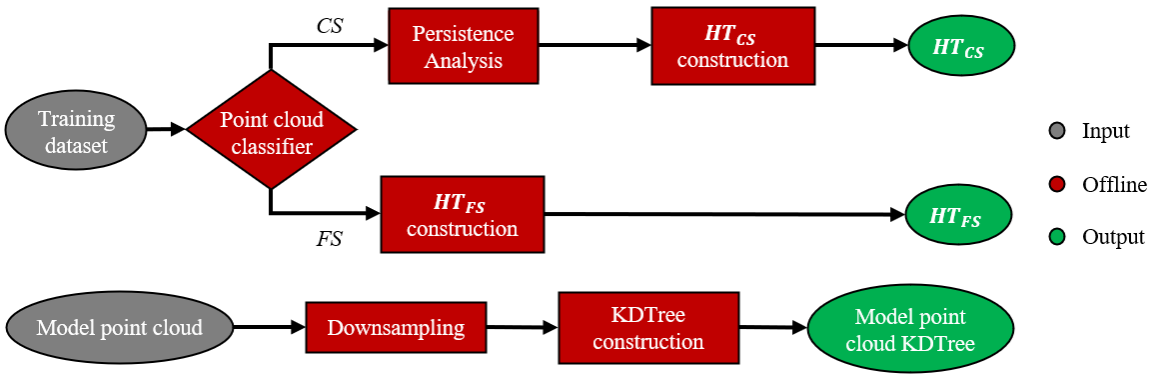

Figure 3.26 - PA-based RANSAC offline block diagram

From the figure it can be observed that the offline phase for the PA-based RANSAC method is simpler than the Label-based RANSAC one, being the outputs only $\boldsymbol{HT_{CS}}$ (paragraph 3.3.1.3), $\boldsymbol{HT_{FS}}$ (paragraph 3.3.1.5) and the KDTree of the model point cloud.

3.4.3. PPF-Based RANSAC

Figure 3.27 shows a block diagram detailing the offline pre-processing phase of the PPF-based RANSAC method.

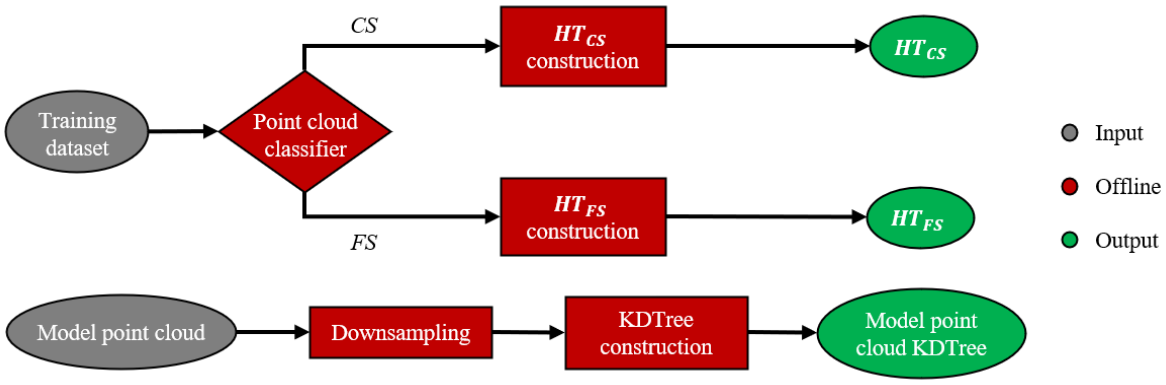

Figure 3.27 - PPF-based RANSAC offline block diagram

From the figure it can be observed that the PPF-based RANSAC method offline phase is the simplest among the proposed HT-based algorithms, since it does not require any feature extraction steps; the outputs are $\boldsymbol{HT_{CS}}$ (paragraph 3.3.1.4), $\boldsymbol{HT_{FS}}$ (paragraph 3.3.1.5) and the KDTree of the model point cloud.

3.5. Initial Pose Determination

In this Section, the initial pose guess determination phase for Label-based, PA-based and PPF-based RANSAC methods is described in detail, exploiting the HTs built offline and the analyzed feature extraction techniques (in the cases of the Label-based and PA-based RANSAC methods).

3.5.1. Label-Based RANSAC

Figure 3.28 shows the Label-based RANSAC online block diagram. Once the point cloud is measured online, it is classified as a CS or FS scan using the same approach adopted in the offline phase (paragraph 3.3.1.1).

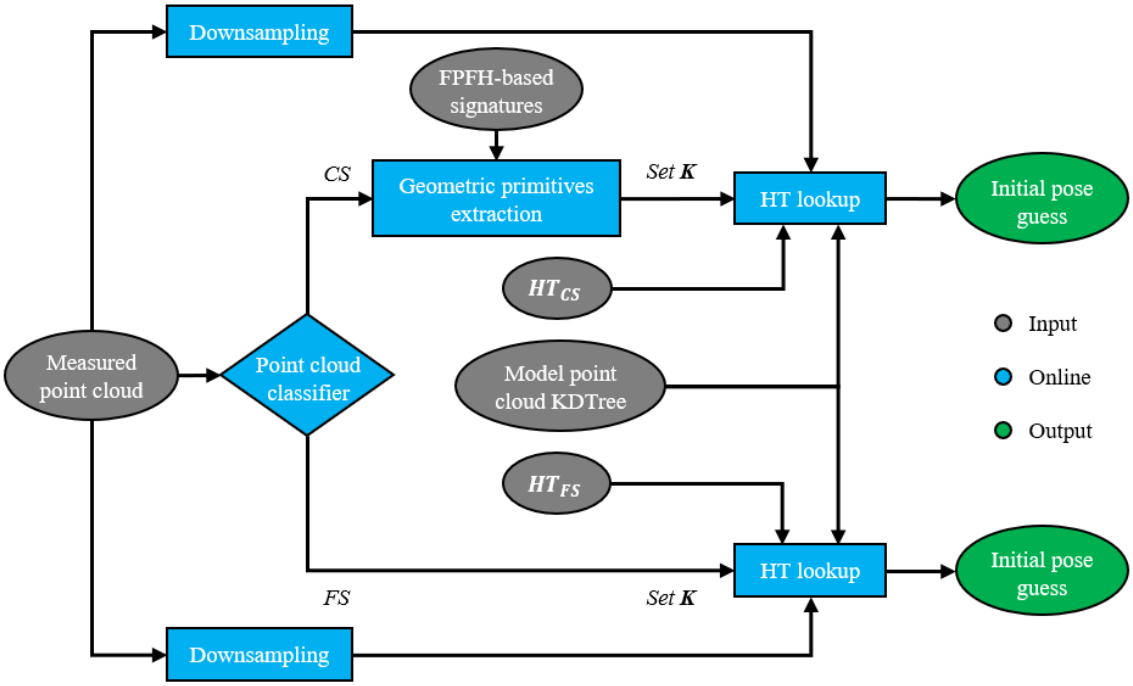

Figure 3.28 - Label-based RANSAC online block diagram

While all the points belonging to an FS scan are part of the set of candidates (***K***) sent in input to the HT lookup block, an additional processing step is required to obtain ***K*** from a CS scan, being only the subset of cylinder and edge points. To identify the points corresponding to these classes, their FPFHs must be compared with the FPFH-based signature obtained for each class. This comparison is done through the double step of *k-NN retrieval* and *Reciprocity Test* described in the analysis of paragraph 3.2.2.3, with the tuning parameters *k* and *d* exactly equal to the optimal ones resulting from the Precision-Recall curves (Table 3.4). Once the set ***K*** is obtained, it enters the HT lookup block, explained in Subsection 3.5.4, whose output is the initial pose guess of the pose.

3.5.2. PA-Based RANSAC

Figure 3.29 shows the PA-based RANSAC online block diagram. From the figure it can be seen that the online phase for the PA-based RANSAC method is very similar to the Label-based RANSAC one; the only difference is that, in case the measured point cloud is classified as CS scan, PA is applied to extract the set ***K*** to be used for the HT lookup step (Subsection 3.5.4).

The tuning parameters adopted to apply the PA are the same as those shown in Table 3.2 for the LiDAR point cloud case.

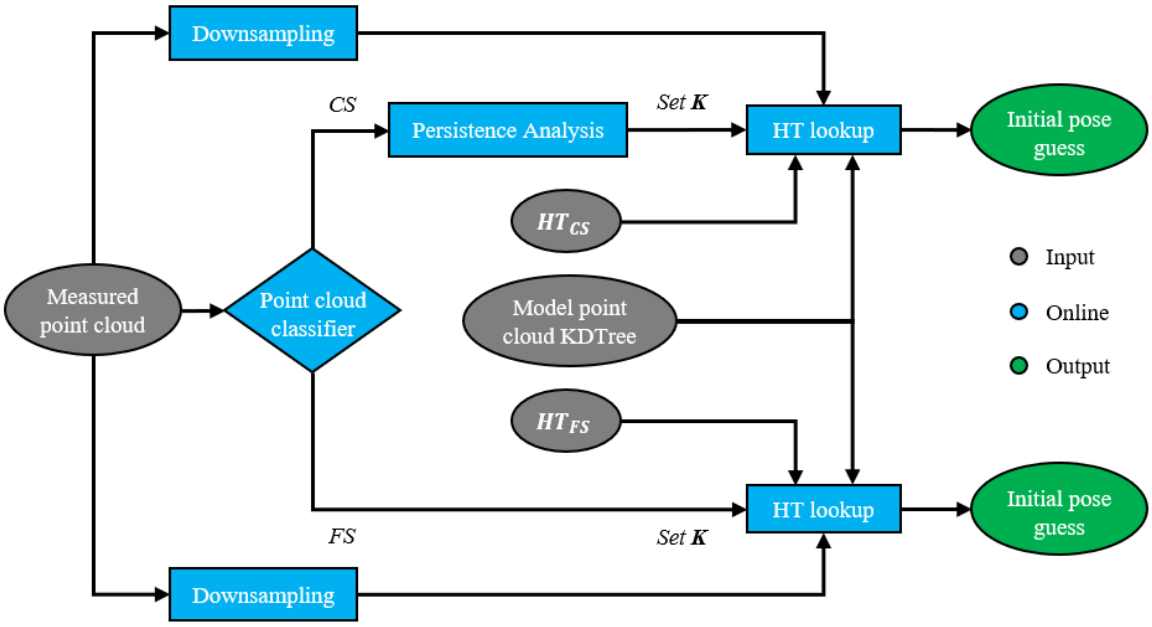

Figure 3.29 - PA-based RANSAC online block diagram

3.5.3. PPF-Based RANSAC

Figure 3.30 shows the PPF-based RANSAC online block diagram.

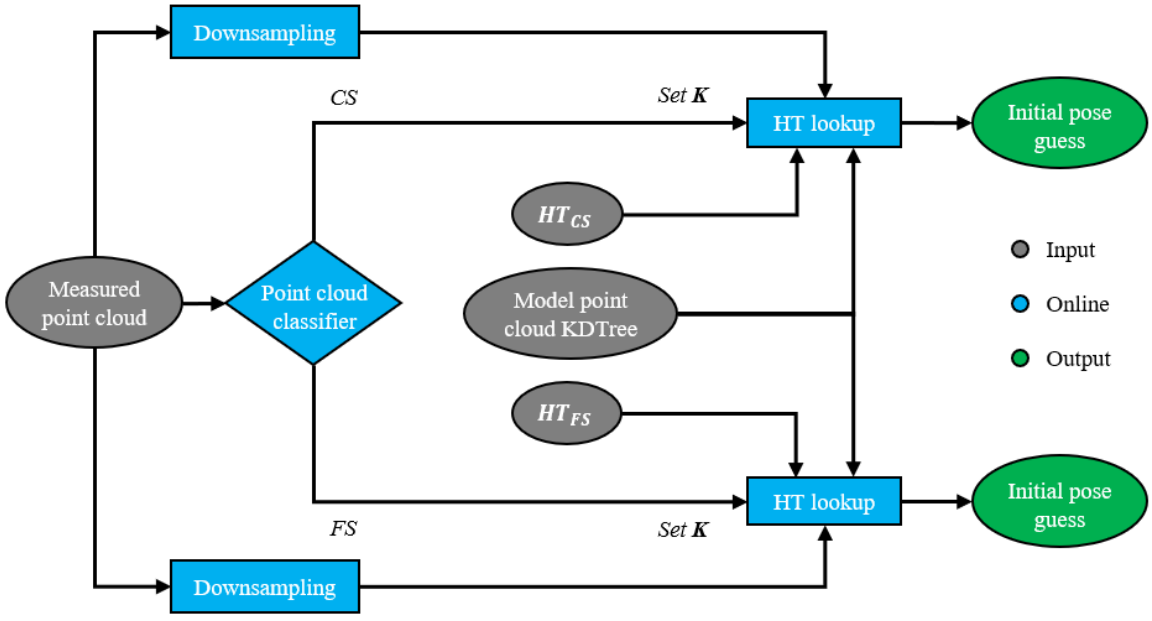

Figure 3.30 - PPF-based RANSAC online block diagram

From the figure it can be seen that, consistently with the offline phase, if the point cloud is classified as a CS scan, there is no candidate extraction phase, but the entire point cloud constitutes the set $\boldsymbol{K}$ that enters the HT lookup phase (Subsection 3.5.4).

3.5.4. Hash Table Lookup

This Subsection analyses in more detail what happens in the HT lookup block encountered in Figure 3.28, Figure 3.29, Figure 3.30.

Once the set $\boldsymbol{K}$ of SRF candidate points is obtained from the measured point cloud, an iterative procedure is started to search for correspondences between measured and model points thus being able to get a pose solution aligning the two point clouds. Specifically, at each iteration, a random triplet/surflet of points (depending on the method considered) is extracted and, after a check on the imposed filtering conditions, i.e. the condition on distances in the case of the Label and PA-based RANSAC methods, $\boldsymbol{d}(i) > 0.025\,m,\ i = 1,2,3$, or the conditions on both distances and angle between normals in the case of the PPF-based RANSAC one, $d > d_{thre}$ and $\alpha_{min} < \alpha < \alpha_{max}$, the hash codes described in Section 3.3.1 are computed and used to access the corresponding HT bucket. In general, each bucket may contain *N* different triplets of points in TRF, where each triplet represents a candidate correct association to the triplet of candidate points from the measured point cloud. For each of the *N* potential associations, a pose guess is computed using the most appropriate alignment algorithm depending on the method considered (paragraph 3.5.4.1). Then, the quality of this pose guess is computed by evaluating the percentage of inliers $in_{\%}$ downstream of the alignment. Two alignment quality assessment approaches have been investigated, presented in paragraph 3.5.4.2. The iterative procedure stops as soon as a threshold representing the minimum desired percentage of inliers ($in_{\%,thre}$) is reached. If it is not reached, the other *N* potential associations are tested and, if it is still not reached, the cycle restarts with the random selection of a triplet/surflet from $\boldsymbol{K}$.

Figure 3.31 shows a block diagram describing the HT lookup phase for the presented methods.

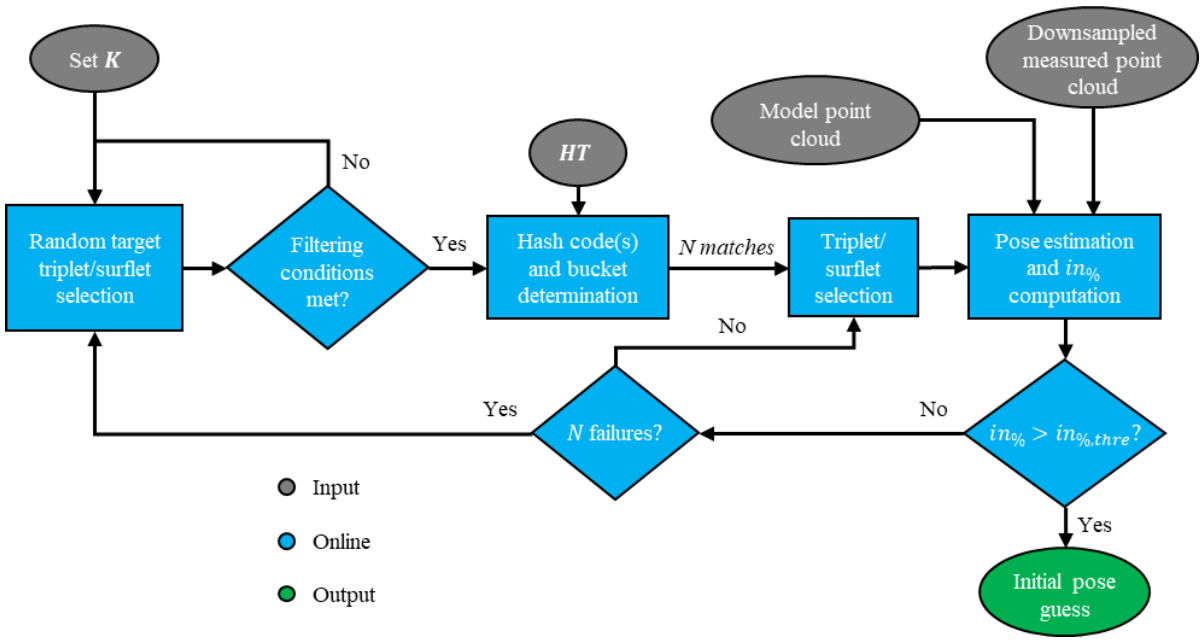

Figure 3.31 – HT lookup. This diagram is representative of both the CS and FS case

3.5.4.1. Alignment Algorithms

The alignment algorithms used in this thesis work are essentially two, depending on whether it uses point triplets or surflets to determine the pose. In particular, the implemented algorithms are *rigid registration algorithms*, therefore they do not take into account variations in scale for example.

Triplets Alignment

The alignment algorithm implemented for the Label-based and PA-based RANSAC methods exploits point triplets to compute the pose matrix from SRF to TRF. The rigid registration problem between two sets of points can be formalized as follows.

Given two sets of 3D points:

- A set of points expressed in the SRF $\{\boldsymbol{p}_i\},\ i = 1,2,\dots,n$
- A set of points expressed in the TRF $\{\boldsymbol{q}_i\},\ i = 1,2,\dots,n$

The goal is to find a rotation matrix $\boldsymbol{R}$ and a translation vector $\boldsymbol{t}$ such that the problem in Equation 3.13 is solved:

$$\boldsymbol{q_i} \approx \boldsymbol{R}\boldsymbol{p_i} + \boldsymbol{t} \tag{3.13}$$

This is a *least-squares problem*, in fact $\boldsymbol{R}$ and $\boldsymbol{t}$ must be such as to minimize the quadratic error between the transformed points of the model point cloud and those of the LiDAR point cloud, as reported in Equation 3.14:

$$\Sigma^2(\boldsymbol{R}, \boldsymbol{t}) = \sum_{i=1}^{n} \|\boldsymbol{q}_i - (\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t})\|^2 \tag{3.14}$$

The algorithm used for the alignment of the triplets of points makes use of the Singular Value Decomposition (SVD) [50]. The steps followed are given below.

First, the centroids of both sets of points are computed, as shown in Equation 3.15.

$$\boldsymbol{p} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{p}_i\,, \qquad \boldsymbol{q} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{q}_i \tag{3.15}$$

Then, the centroids of the related sets are subtracted from each point, thus obtaining two new sets of points centered with respect to their centroids, as reported in Equation 3.16.

$$\boldsymbol{p}_i' = \boldsymbol{p}_i - \boldsymbol{p}, \qquad \boldsymbol{q}_i' = \boldsymbol{q}_i - \boldsymbol{q} \tag{3.16}$$

In this way, rotation and translation are decoupled. In fact, the original least-squares problem is now reduced to rotation only, and can be formulated as reported in Equation 3.17.

$$\Sigma^2(\boldsymbol{R}) = \sum_{i=1}^{n} \|\boldsymbol{q}_i' - \boldsymbol{R}\boldsymbol{p}_i'\|^2 \tag{3.17}$$

Therefore, the problem is broken into two parts:

1. Find $\boldsymbol{R}$ to minimize Σ^2 in Equation 3.17, a step that will be performed through the SVD;
2. Find $\boldsymbol{t}$ through Equation 3.18:

$$\boldsymbol{t} = \boldsymbol{q} - \boldsymbol{R}\boldsymbol{p} \tag{3.18}$$

Once the two new sets of points have been calculated, the covariance matrix is constructed by using Equation 3.19:

$$\boldsymbol{H} = \sum_{i=1}^{n} \boldsymbol{p}_i' \boldsymbol{q}_i'^{T} \tag{3.19}$$

On the obtained matrix, the SVD is performed, decomposing $\boldsymbol{H}$ into three matrices through Equation 3.20:

$$\boldsymbol{H} = \boldsymbol{USV^T} \tag{3.20}$$

Finally, the optimal rotation matrix $\boldsymbol{R}$ that aligns the two sets of points is given by Equation 3.21:

$$\boldsymbol{R} = \boldsymbol{VU^T} \tag{3.21}$$

However, it is necessary to verify that the determinant of $\boldsymbol{R}$ is positive, specifically, that $det(\boldsymbol{R}) = 1$. If $det(\boldsymbol{R}) < 0$, it means that the operation includes a reflection, which is corrected by changing the sign of the last column of the matrix $\boldsymbol{V}$.

Once the rotation matrix $\boldsymbol{R}$ has been calculated, the translation vector $\boldsymbol{t}$ is easily calculated using Equation 3.18. For this problem to have a solution, it is sufficient that the sets of points consist of three non-collinear points.

Surflets Alignment

The alignment algorithm used for the PPF-based RANSAC method is different to the one adopted by Label-based and PA-based RANSAC methods, since it aligns surflets rather than triplets. Specifically, the surflet expressed in SRF and the corresponding one expressed in TRF are aligned using an ad-hoc algorithm that separately solves the rotation and translation problems.

First, it computes the rotation quaternion $\boldsymbol{r}$ from SRF to TRF as a product of two quaternions $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$, as shown in Equation 3.22:

$$\boldsymbol{r} = \boldsymbol{r_1} \otimes \boldsymbol{r_2} \tag{3.22}$$

The two quaternions represent the following rotations:

1. The first quaternion $\boldsymbol{r_1}$ aligns the difference vectors of points, aiming to rigidly rotate one surflet on the other without considering the information on the normals;
2. The second quaternion $\boldsymbol{r_2}$ aligns the projections of the normals on a plane orthogonal to the difference vector.

After computing the rotation quaternion, the translation vector $\boldsymbol{t}$ is computed through the difference between the centroid of the surflet $\boldsymbol{q}$, expressed in TRF, and the centroid of the surflet $\boldsymbol{p}$, expressed in SRF and rotated through the computed quaternion $\boldsymbol{r}$, as shown in Equation 3.23:

$$\boldsymbol{t = q - rpr^*} \tag{3.23}$$

where $\boldsymbol{r^*}$ is the conjugate of $\boldsymbol{r}$.

The basis of this procedure is the assumption that, in the case where the points are not too close, the direction of the point difference vector is more reliable than the direction of the estimated normals; therefore, the difference vectors are perfectly aligned, while the normals contribute equally to an average rotation angle about the axis that is given by the aligned difference vectors.

3.5.4.2. Alignment Evaluation

The quality of the pose hypotheses determined through the alignment algorithm is evaluated in RANSAC style by counting the inliers. To make this search faster, without necessarily having to check all the possible combinations of point triplets, a percentage of inliers to be reached is set as a stop condition. Two main approaches have been developed for assessing alignment quality: the Nearest Neighbor approach and the Binary Matching with 3D voxel grid.

Nearest Neighbor approach

The *Nearest Neighbor (NN) approach* is based on finding, for each point of the measured point cloud transformed in TRF based on the pose guess, the NN between the points of the model point cloud, and calculating the Euclidean distance between the pairs of correspondences. If the distance value is lower than a certain threshold $d_{in,thre}$, the point of the measured point cloud would be counted as an inlier. These inliers are searched leveraging the KDTree of the model

point cloud to efficiently find the NNs between the two point clouds. In this way, the percentage of inliers $in_{\%}$ can be computed as shown in Equation 3.24:

$$in_{\%} = \frac{\#inliers}{\#downsampled\ data\ points} * 100 \tag{3.24}$$

In order to limit the computational time of this step, both the model and the measured point cloud are downsampled with the same voxel size v_{size}. As already mentioned at the beginning of Section 3.5.4, if this percentage reaches a predetermined value $in_{\%,thre}$, then the alignment is considered good and therefore the HT lookup cycle stops. Specifically, two thresholds of inlier percentages are defined in this discussion, indicated as $in_{\%,thre,CS}$ and $in_{\%,thre,FS}$ depending on whether the scan considered is respectively a CS or a FS scan. The values of these tuning parameters are shown in paragraph 5.2.1.1.

Binary Matching with 3D Voxel Grid

The *Binary Matching (BM) approach*, instead, is based on the counting of inliers found by checking whether the points of the transformed measured point cloud fall within the voxels of a grid representing the model point cloud. More precisely, the model point cloud is replaced by a 3D grid of voxels, built offline and encoded through a HT, called in this discussion $\boldsymbol{HT}_{voxel}$; $\boldsymbol{HT}_{voxel}$ basically represents a 3D dictionary in which, for each bucket (corresponding to the single voxel) that contains at least one point of the model point cloud, the digit "1" is stored.

The hash code associated with the bucket containing the information "1" is obtained using as a key the triplet of coordinates of the generic point of the model point cloud, using the hash function shown in Equation 3.25:

$$\boldsymbol{v}_{hash}(i) = floor\left(\frac{m * \boldsymbol{d}(i)}{D}\right), \qquad i = 1,2,3 \tag{3.25}$$

The hash function used is the same one already seen for the construction of the HTs explained in Section 3.3, where m represents the number of buckets, $\boldsymbol{d}$ represents the coordinate vector of the generic point of the model point cloud, while D has been left unchanged with respect to the definition given in Section 3.3.

Clearly, if the number of buckets m chosen is small, and therefore the voxels of the grid are large enough to contain more than one point, if a second point is associated with a bucket already containing the information "1", no operation is performed and the next point is considered.

The higher the number of buckets m, the denser the voxel grid (i.e., the smaller the size of each individual voxel), and this would therefore be equivalent to requiring a greater precision in the alignment to obtain a high percentage of inliers, at the expense of increased computational time.

Given these premises, the evaluation of the alignment quality using this approach is described below: in the online phase, for each point of the transformed measured point cloud, the corresponding hash code is computed, through the hash function in Equation 3.25, and it is checked whether the corresponding bucket contains the value "1" or not; if so, the inlier counter is increased by 1. Then, in the same way as the NN approach, it is possible to calculate the percentage of inliers $in_{\%}$ by dividing the total inliers obtained by the number of points of the downsampled measured point cloud (Equation 3.24) and, if $in_{\%} > in_{\%,thre}$, the alignment is considered good and the HT lookup cycle stops.

Simulations have been performed using both the alignment evaluation approaches (NN and BM) for Label and PA-based RANSAC algorithms. In paragraph 5.2.2.2, a comparison between the two variants is reported, but limited to the estimation of the initial pose guess, since the evaluation of the flip checks has only been implemented using the NN approach.

3.6. Post-Processing

This Section describes the refinement procedure of the initial pose guess, through an application of ICP and a series of checks aimed at correcting any incorrect poses that have passed the HT lookup phase. A diagram of the pose refinement procedure is shown in Figure 3.32.

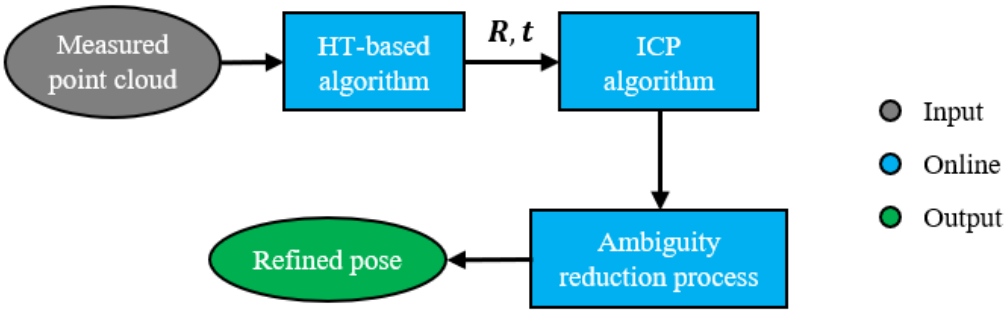

Figure 3.32 – Post-processing

3.6.1. Iterative Closest Point

Once the pose between measured and model point clouds has been determined, it is refined through the Iterative Closest Point (ICP). There are two types of ICP algorithms implemented in Open3D:

1. *Point to Point (P2P) ICP*. It aims to minimize the Euclidean distance between corresponding points in the measured and model point clouds; it establishes correspondences based on the closest point pairs;
2. *Point to Plane (P2L) ICP*. It minimizes the distance between points and planes in the model point cloud; it establishes correspondences between points in the measured cloud and the nearest planes in the model cloud.

For the algorithms presented in this work, the P2L ICP is implemented.

Given as input to the function the transformation matrix resulting from the HT-based algorithm, in this phase, through the ICP steps, the transformation matrix is recursively calculated for a predetermined maximum number of iterations or until convergence is reached, thus updating the pose from time to time. Equation 3.26 shows the pose update from iteration *i* to iteration *i+1* via ICP:

$$\boldsymbol{T}_{i+1} = \boldsymbol{T}_{ICP}^{i,i+1}\,\boldsymbol{T}_i \tag{3.26}$$

After updating the pose matrix, the Root Mean Square Error, RMSE (used to impose a stop condition in case of convergence) and the cost function (which will be used to develop an

Autonomous Failure Detection algorithm, explained in Section 5.3) are evaluated. In particular, the cost function is calculated through Equation 3.27:

$$C(\boldsymbol{T}) = \sum_{(\boldsymbol{p},\boldsymbol{q})\in\boldsymbol{K}} \left(\boldsymbol{n}_{\boldsymbol{q}}^{T} \cdot (\boldsymbol{T}\boldsymbol{p} - \boldsymbol{q})\right)^{\mathbf{2}} \tag{3.27}$$

where:

- $\boldsymbol{p}$ represents the point of the correspondence pair $(\boldsymbol{p}, \boldsymbol{q})$ belonging to the measured point cloud.
- $\boldsymbol{q}$ represents the point of the correspondence pair $(\boldsymbol{p}, \boldsymbol{q})$ belonging to the model point cloud.
- $\boldsymbol{T}$ represents the SRF to TRF transformation matrix. Therefore, $\boldsymbol{Tp}$ represents the transformed point of the measured point cloud, which must align to the corresponding point in the model point cloud $\boldsymbol{q}$.
- $\boldsymbol{n_q}$ represents the normal at the point $\boldsymbol{q}$.
- $\boldsymbol{K}$ represents the set of correspondences that comes out of the current ICP iteration.

The difference with respect to the RMSE is simply that, in the definition of the RMSE, the sum of the squared differences is divided by N and put under root. As mentioned above, the RMSE value is used to impose a further condition, shown in Equation 3.28:

$$RMSE_i - RMSE_{i-1} < icp_{conv} \tag{3.28}$$

If this condition is satisfied, i.e., the difference between the RMSE of two consecutive iterations is lower than a certain threshold icp_{conv}, the algorithm has converged and so it stops before reaching the maximum number of iterations imposed.

3.6.2. Ambiguity Reduction Process

This Subsection is dedicated to the explanation of additional checks that have been implemented downstream of the ICP to strengthen the algorithms developed in the presence of incorrect poses that have passed the HT lookup phase: given the geometry of the satellite under study, it is possible that the inlier percentage set to exit the HT lookup block is satisfied with

an incorrect pose; therefore, it is necessary to implement checks that allow these cases to be identified and the correct pose to be estimated.

A case of incorrect pose that can exceed the threshold of the percentage of inliers is that of a satellite flipped, about its large planar surface, w.r.t the correct position. Therefore, the $\boldsymbol{T}_{ICP}$ transformation, that is the pose solution produced by the ICP-based pose refinement step expressed as a 4x4 roto-translation matrix from SRF to TRF, could correspond either to the desired transformation from SRF to TRF ($\boldsymbol{T}_{SRF \to TRF}$) or to a transformation from SRF to a Flipped Target Reference Frame (FTRF). Figure 3.33 shows the aforementioned problem.

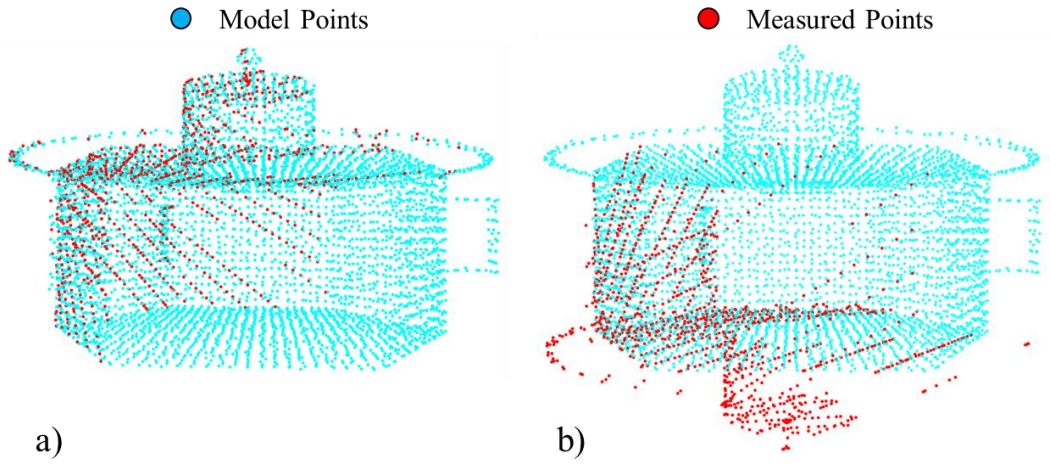

Figure 3.33 - a) Case of correctly estimated pose: $\boldsymbol{T}_{ICP} = \boldsymbol{T}_{SRF \to TRF}$. b) Case of incorrect pose that has passed the HT lookup phase: $\boldsymbol{T}_{ICP} = \boldsymbol{T}_{SRF \to FTRF}$

Therefore, two possible pose solutions can be investigated, which are shown in Equation 3.29 and Equation 3.30:

$$\boldsymbol{T}_{SRF \to TRF_1} = \boldsymbol{T}_{ICP} \tag{3.29}$$

$$\boldsymbol{T}_{SRF \to TRF_2} = \boldsymbol{T}_{FTRF \to TRF} \boldsymbol{T}_{ICP} \tag{3.30}$$

where $\boldsymbol{T}_{FTRF \to TRF}$ is a matrix dependent on the target reference geometry. Given these two possible solutions, the percentage of inliers is calculated for both cases, and the matrix corresponding to the case with the highest percentage of inliers is selected as the final one.

The considered flip cases are shown in Figure 3.34.

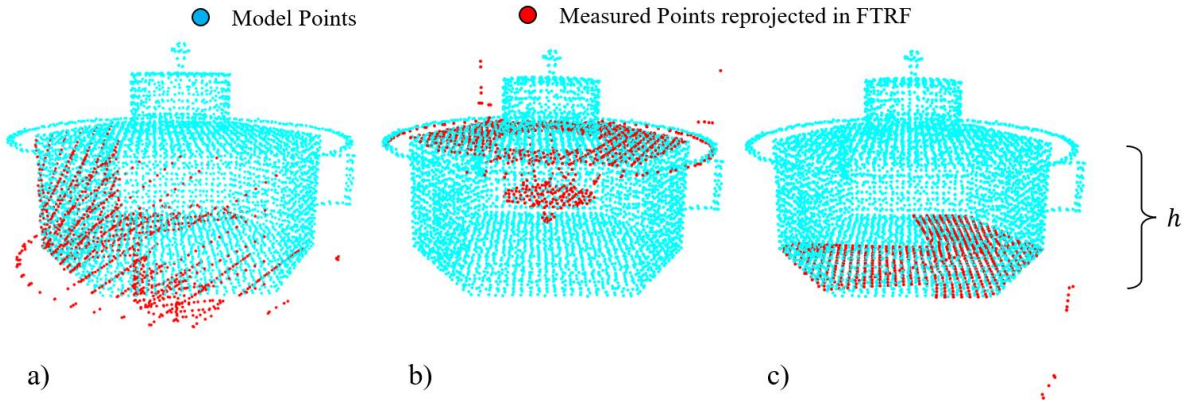

Figure 3.34 - Flips implemented. a) Flip check performed for all scans. b) FS Flip check: upper plane; c) FS Flip check: lower plane

Given the Client Satellite geometry, two different types of flip checks are performed: one applied to both CS and FS scans (a), and an additional check for FS scans only (b, c), which is performed before (a). The latter flip check type addresses the case in which the corresponding planes match, but with opposite normal orientation. Therefore, to detect such a case, a comparison is made between the direction of corresponding normal vectors. If the median angle between corresponding normals is greater than 90°, it is highly likely that the FS scan is upside down, and therefore it is necessary to apply, downstream of $\boldsymbol{T}_{ICP}$, a $\boldsymbol{T}_{FTRF \to TRF}$ transformation to flip the measured point cloud. The choice of the median instead of the mean derives from the fact that, in this way, the final angle value is not influenced by outliers. Since the FS scan considered (upper or lower plane) is not known a priori, the transformations related to both cases (b, c) are applied and the best one in terms of number of inliers is selected.

Table 3.6 shows rotations and translations composing the transformation matrix $\boldsymbol{T}_{FTRF \to TRF}$, where h is the height of the hexagonal main body. In particular, regarding rotation, the above mentioned flips are tested for two cases, i.e., the rotation around the pitch axis and around the yaw axis, since, depending on the axis around which the rotation is performed, the detection of the handles can make the difference in recognizing the correct pose. The best one is selected by counting the inliers.

		γ, β, α *rotation sequence [deg]*	x, y, z *translation [m]*
$T_{FTRF \to TRF}$	***Case (a)***	$0, 180, 0$	$-h, 0, 0$
		$180, 0, 0$	$-h, 0, 0$
	Case (b)	$0, 180, 0$	$0, 0, 0$
		$180, 0, 0$	$0, 0, 0$
	Case (c)	$0, 180, 0$	$-2h, 0, 0$
		$180, 0, 0$	$-2h, 0, 0$

Table 3.6 – Flip matrices $T_{FTRF \to TRF}$ defined

The reason why the flip check is performed downstream of the ICP is simply due to the fact that, if the alignment obtained upstream were very rough (therefore, percentage of inliers just above the minimum to exit the HT lookup cycle), even if the flip were correct, there would be a higher probability of failing to distinguish the correct case from the flipped case by comparing the respective percentages of inliers: indeed, even more so in this specific case, the increase in the percentage of inliers downstream of the flip is mainly entrusted to the good alignment of the points off plane (i.e., those points belonging to the toroid, the cylinder and the sphere); therefore, the better the starting alignment, the easier it will be to see this difference in the percentage.

4. Open3D Global Registration Algorithms

The global registration algorithms developed during this thesis work have been compared with algorithms already implemented in Open3D. These algorithms are FPFH-based RANSAC and Fast Global Registration (FGR) [46], briefly described in this Chapter.

4.1. FPFH-Based RANSAC

The implementation procedure of the Open3D FPFH-based RANSAC is described below. Both the model and the measured point cloud are downsampled with voxel size v_{size}, normals are estimated, and then, for each point of the point clouds, the FPFH is calculated.

At this point, the RANSAC algorithm is applied: iteratively, n points are randomly extracted from the model point cloud; then, the set of corresponding points in the measured point cloud is identified by searching for the NN in the FPFH space. For a quick search, Open3D provides algorithms to filter the good matches:

1. *CorrespondenceCheckerBasedOnDistance*. It checks the goodness of a match by measuring the distance between the corresponding points in the model and the measured point cloud downstream of the alignment. This distance is compared with a threshold entered as input; the match is considered valid if the distance between the points is below the threshold. The value of this tuning parameter is shown in paragraph 5.2.1.1.
2. C*orrespondenceCheckerBasedOnEdgeLength*. It checks the quality of the correspondences from another point of view, that is, considering the length of the segments obtained from pairs of points on the model and on the measured point cloud. Specifically, given two points on the model point cloud and the two corresponding points on the measured point cloud, the two segments are constructed and the lengths are calculated ($\|edge_m\|$ and $\|edge_t\|$, respectively). This condition is satisfied if the relations shown in Equation 4.1 are valid:

$$\frac{\|edge_m\|}{\|edge_t\|} > thre, \qquad \frac{\|edge_t\|}{\|edge_m\|} > thre \tag{4.1}$$

where $thre$ is the threshold set in input (value shown in paragraph 5.2.1.1). In fact, it is possible to exploit the similarity between the geometries constructed on both point clouds, given the nature of the rigid transformation.

3. *CorrespondenceCheckerBasedOnNormal*. It checks the consistency between the normals of two corresponding points in the two point clouds. In this case, given two corresponding points, one on the model and one on the measured point cloud, the angle between the normals is measured and compared with a certain threshold. The correspondence is considered valid if the angle between the normals is less than this threshold.

The checks implemented in the classic Open3D FPFH-based RANSAC algorithm to find good matches are the first two.

As highlighted in Chapter 3, a very similar concept has been applied in the development of the three global registration algorithms presented in this thesis. Specifically, the iterative sampling of correspondences, followed by inlier computation for alignment quality assessment, is a typical characteristic of RANSAC-based approaches and is effectively utilized in the HT-based methods.

The matches that satisfy the above conditions are used to compute the transformation. A *maximum number of iterations* and a *confidence probability* (values shown in paragraph 5.2.1.1) are set as stopping conditions for the algorithm, where the latter indicates, by definition, the desired probability that the RANSAC algorithm provides at least one useful result after running.

More precisely, RANSAC returns a good result if, in at least one iteration, it selects only inliers from the dataset, when the n points are randomly selected to compute the transformation. Using this definition of success probability p, that is, at least one iteration out of k, n inliers are selected, the failure probability is defined as the probability that, in all k iterations, a set of inliers is never selected, but at least one outlier is always selected among the n points. This definition is shown in Equation 4.2:

$$1 - p = (1 - w^n)^k \tag{4.2}$$

where w represents the probability that a single selected point is an inlier, which is nothing but the inlier ratio (a number between 0 and 1 that has exactly the same meaning as the percentage of inliers previously seen).

From Equation 4.2, given as input the value of the desired probability, it is possible to calculate the number of iterations needed through Equation 4.3:

$$k = \frac{log(1 - p)}{log\ (1 - w^n)} \tag{4.3}$$

Clearly, if the selected confidence level requires too high a number of iterations, the other stop condition, i.e. the maximum number of iterations, stops the algorithm, avoiding large computation times.

The larger these two numbers are, the more accurate the result is, but also the more computational time is required for the algorithm to output the solution.

4.2. Fast Global Registration

In the Open3D tutorial, FGR is also presented as an algorithm for global registration, in contrast to the FPFH-based RANSAC approach.

Using the same point cloud pre-processing procedure (downsampling, normal estimation, FPFH estimation), the FGR implementation follows, as already illustrated in Chapter 2 [34]. Again, the FGR algorithm takes as input the two point clouds subjected to downsampling, the two sets of FPFH, and a distance threshold for the identification of correspondences.

5. Experiments

This Chapter presents the performance results produced by the developed global registration algorithms, which are compared with the Open3D FPFH-based RANSAC and FGR algorithms.

5.1. Evaluation Metrics

In this Section, the error metrics for evaluating the performance of the algorithms are defined. Given the quasi-symmetric geometry of the satellite under study, specifically around the roll axis, for the method evaluation, the symmetric variant of the estimated pose which minimizes the error is considered. Let $\boldsymbol{T}$ be the pose ground truth and $\widehat{\boldsymbol{T}}$ be the estimated pose, within the set of equivalent poses by symmetry, with the smallest error.

The Average Distance of model points for Distinguishable (ADD) and for Indistinguishable (ADI) points are computed [51, 52], given a complete uniform model point cloud $\mathcal{M}$, which are defined in Equation 5.1 and Equation 5.2:

$$ADD = \underset{\boldsymbol{x}\in\mathcal{M}}{\text{avg}} \|\boldsymbol{T}\boldsymbol{x} - \widehat{\boldsymbol{T}}\boldsymbol{x}\| \tag{5.1}$$

$$ADI = \underset{\boldsymbol{x}_1\in\mathcal{M}}{\text{avg}} \min_{\boldsymbol{x}_2\in\mathcal{M}} \|\boldsymbol{T}\boldsymbol{x}_1 - \widehat{\boldsymbol{T}}\boldsymbol{x}_2\| \tag{5.2}$$

Furthermore, for each testing sample, the translational and rotational errors are computed as shown in Equation 5.3 and Equation 5.4:

$$t_{err} = \|\boldsymbol{t} - \hat{\boldsymbol{t}}\| \tag{5.3}$$

$$\phi_{err} = 2\arccos|\langle \boldsymbol{q}, \hat{\boldsymbol{q}}^* \rangle| \tag{5.4}$$

where $\boldsymbol{t}$, $\boldsymbol{q}$ are the translational and rotational components of $\boldsymbol{T}$, while $\hat{\boldsymbol{t}}, \hat{\boldsymbol{q}}$ are the translational and rotational components of $\widehat{\boldsymbol{T}}$. The rotational part is expressed in terms of quaternions: specifically, $\boldsymbol{q}$ is the true attitude quaternion, $\hat{\boldsymbol{q}}^*$ is the conjugate of the estimated attitude

quaternion, while $|\langle \boldsymbol{q}, \hat{\boldsymbol{q}}^* \rangle|$ is the scalar part of the quaternion resulting from the quaternion product.

These error metrics are used to compute the Success Rate (SR), which is defined as the percentage of correctly estimated poses, as shown in Equation 5.5:

$$SR = \frac{\#\ correct\ estimates}{\#\ test\ samples} * 100 \tag{5.5}$$

A pose is considered correctly estimated if $\phi_{err} < 5°$ and $t_{err} < 5\ cm$ are simultaneously satisfied.

Finally, the computational time of the online phase is measured from the acquisition of the LiDAR point cloud to post-processing. The simulations are conducted on a machine equipped with an Intel(R) Core(TM) i9-14900K CPU @ 3.20 GHz and 64 GB DDR5 RAM.

5.2. Performance Analysis

The next Subsections show the results of the developed algorithms. The results are obtained by testing the global registration algorithms on a *testing dataset* composed by the 20% of the entire 1000-scan dataset used for analyzes.

In particular, in Subsection 5.2.1, the values of the tuning parameters set and the comparison results between the main algorithms, without and with post-processing, are shown; in Subsection 5.2.2, some comparisons are shown between different variants of the developed HT-based algorithms; finally, in Subsection 5.2.3, additional results regarding further analyses carried out on the Open3D implementations are presented.

5.2.1. Comparison of the Main Algorithms

This Subsection presents the performance comparison results between the main variants of the developed HT-based algorithms and the Open3D FPFH-based RANSAC and FGR, appropriately modified by integrating to the already existing architecture the classification of

point clouds in CS and FS scans and the post-processing block (ICP + flip checks) also used for the HT-based algorithms (Subsection 3.6).

As already seen in Chapter 3, some variants of the HT-based algorithms have been proposed, depending on:

1. The offline $\boldsymbol{HT_{CS}}$ construction methodology (*model HT* or *training HT*, Section 3.3);
2. The alignment quality evaluation (*NN* or *BM* with the 3D voxel grid, paragraph 3.5.4.2).

The performances of the Label-based RANSAC, PA-based RANSAC and PPF-based RANSAC algorithms shown in this Subsection are those obtained using the *training HTs* and the *NN approach* for alignment quality evaluation.

5.2.1.1. Tuning Parameters

Table 5.1 shows the values of the tuning parameters used for the construction of the HTs and defined in Subsection 3.3.1.

Training HTs	$\boldsymbol{n_{cycles}}$	***Hash function parameters***				***Filtering parameters***		
		$\boldsymbol{m}$	$\boldsymbol{D}\ [\boldsymbol{m}]$	$\boldsymbol{m_a}$	$\boldsymbol{A}\ [\boldsymbol{deg}]$	$\boldsymbol{d_{thre}}\ [\boldsymbol{m}]$	$\boldsymbol{\alpha_{min}}\ [\boldsymbol{deg}]$	$\boldsymbol{\alpha_{max}}\ [\boldsymbol{deg}]$
Label $\boldsymbol{HT_{CS}}$	5e6	100	1.2	N/A	N/A	0.025	N/A	N/A
PA $\boldsymbol{HT_{CS}}$	5e6	100	1.2	N/A	N/A	0.025	N/A	N/A
PPF $\boldsymbol{HT_{CS}}$	5e6	100	1.2	40	170	0.025	10	170
$\boldsymbol{HT_{FS}}$	1e6	100	1.2	N/A	N/A	0.05	N/A	N/A

Table 5.1 - Tuning parameters selected for training HTs construction

Table 5.2 shows the tuning parameters used for normal, FPFH estimation and online extraction of the set of candidates $\boldsymbol{K}$.

Label-based RANSAC method	***Normals***		***FPFH***		***Cylinder points extraction***		***Edge points extraction***	
	$\boldsymbol{r}\ [\boldsymbol{m}]$	$\boldsymbol{max_{nn}}$	$\boldsymbol{r}\ [\boldsymbol{m}]$	$\boldsymbol{max_{nn}}$	$\boldsymbol{d_{FPFH}}$	$\boldsymbol{k_{FPFH}}$	$\boldsymbol{d_{FPFH}}$	$\boldsymbol{k_{FPFH}}$
	0.07	40	0.07	100	55	190	50	180
PA-based RANSAC method	***Normals***		***FPFH***		***Persistence Analysis parameters***			
	$\boldsymbol{r}\ [\boldsymbol{m}]$	$\boldsymbol{max_{nn}}$	$\boldsymbol{r}\ [\boldsymbol{m}]$	$\boldsymbol{max_{nn}}$	$\boldsymbol{\beta}$	$\boldsymbol{r_i}\ [\boldsymbol{m}]$		$\boldsymbol{max_{nn}}$
	0.07	40	0.07	100	1	0.05, 0.07, 0.1		100
PPF-based RANSAC method	***Normals***		***FPFH***		***No feature extraction step***			
	$\boldsymbol{r}\ [\boldsymbol{m}]$	$\boldsymbol{max_{nn}}$	$\boldsymbol{r}\ [\boldsymbol{m}]$	$\boldsymbol{max_{nn}}$				
	0.07	40	N/A	N/A				

Table 5.2 - HT-based normal, FPFH estimation and feature extraction tuning parameters

where r and max_{nn} for normal estimation are defined in Subsection 3.2.1, d_{FPFH} and k_{FPFH} are defined in paragraph 3.2.2.3, while β, r_i and max_{nn} for PA are defined in paragraph 3.2.2.2. Table 5.3 shows the tuning parameters used for normal and FPFH estimation for FPFH-based RANSAC and FGR.

Method	***Normals***		***FPFH***	
	$\boldsymbol{r}\ [\boldsymbol{m}]$	$\boldsymbol{max_{nn}}$	$\boldsymbol{r}\ [\boldsymbol{m}]$	$\boldsymbol{max_{nn}}$
FPFH-based RANSAC	0.07	40	0.07	100
FGR	0.1	40	0.1	100

Table 5.3 - FPFH-based RANSAC and FGR normal and FPFH estimation tuning parameters

The reason for the difference between FPFH-based RANSAC and FGR in the choice of r for the computation of normals and FPFHs lies in the relatively better performance of FGR in the case of $r = 0.1\ m$, compared to $r = 0.07\ m$.

Regarding point cloud classification, for the HT-based methods $\tau_{\sigma_{N,thre}} = 0.1$ has been set to distinguish CS scans from FS scans while, for FPFH-based RANSAC and FGR algorithms, $\tau_{\sigma_{N,thre}} = 0.25$ has been set since the workflow reported in [46], also reported in Chapter 4, involves the implementation of point cloud downsampling before estimating the normal unit

vectors, which therefore alters the threshold value to be considered. The value of voxel size set for point cloud downsampling is equal to $0.025\ m$ and the threshold $\tau_{\sigma_{N,thre}} = 0.25$ is selected in such a way as to classify as FS scan and CS scan the exact same samples classified using $\tau_{\sigma_{N,thre}} = 0.1$ on the non-downsampled testing dataset. For completeness of the analyses, simulations have also been performed without downsampling of the point clouds (and therefore with the same threshold adopted for the HT-based methods), indeed demonstrating the greater effectiveness of FPFH-based RANSAC and FGR with downsampling. The results are shown in Subsection 5.2.3.

Table 5.4 and Table 5.5 finally present the tuning parameter values for alignment evaluation and ICP, respectively, for the HT-based algorithms, selected through an iterative process to determine the optimal ones.

Method	***Alignment evaluation***			
	$\boldsymbol{v_{size}}$ ***[m]***	$\boldsymbol{d_{in,thre}}$ ***[m]***	$\boldsymbol{in_{\%,thre,CS}}$ ***[%]***	$\boldsymbol{in_{\%,thre,FS}}$ ***[%]***
HT-based	0.025	0.05	80	90

Table 5.4 – Alignment evaluation tuning parameters adopted

Method	***ICP***		
	$\boldsymbol{icp_{thre}}$ ***[m]***	$\boldsymbol{n_{it,max}}$	$\boldsymbol{icp_{conv}}$ ***[m]***
HT-based	0.05	100	10^{-6}

Table 5.5 - ICP tuning parameters adopted

where:

- v_{size} is the voxel size downsampling parameter used to reduce the number of points of the model and the measured point cloud;
- $d_{in,thre}$ is the distance threshold that allows the identification of inliers;
- $in_{\%,thre,CS}$ and $in_{\%,thre,FS}$ represent the percentages of inliers to be satisfied for the pose to be considered good, respectively for CS and FS scans;
- icp_{thre} is the distance threshold set for the execution of the ICP;
- $n_{it,max}$ is the maximum number of ICP iterations;

- icp_{conv} is the convergence threshold, calculated as the difference between the RMSE of two consecutive iterations that, if reached, stops the ICP regardless of the number of iterations reached.

Regarding FPFH-based RANSAC, instead, the number of points n, used to compute the initial pose guess, is set equal to 3; the input parameters of the *CorrespondenceCheckerBasedOnDistance* and *CorrespondenceCheckerBasedOnEdgeLength* functions (Section 4.1) are set equal to $0.05\,m$ and 0.9 respectively and, finally, the convergence criteria of the algorithm, which are the *maximum number of iterations* and the *confidence probability*, are set to 100000 and 0.999, respectively; finally, for FGR, the only input parameter required by the Open3D function is the maximum correspondence distance, set equal to $0.05\ m$. For both algorithms, the same values of icp_{thre}, $n_{it,max}$ and icp_{conv} are used.

5.2.1.2. Comparison Results

Figure 5.1 shows the performance comparison between the five methods above mentioned both without and with the post-processing steps included into the algorithmic pipeline. The top 4 plots show the percentage of samples of the testing dataset whose ADD/ADI, normalized with respect to the maximum size of the satellite, considered equal to $1.2\ m$, falls below a certain threshold. The bottom 4 plots show the performance of the algorithms in terms of rotational and translational error ϕ_{err} and t_{err}, in ADD/ADI style, thus showing the percentage of samples of the testing dataset whose rotational/translational error falls below a certain angle/distance. The larger the area under the curve, the better the performance. For all the 8 plots the [min., max.] limits of the horizontal axis are set to $[0, p]$, with p such that 96% of the samples of the testing dataset are reached with the worst RANSAC-based method.

From the shown plots, first, it can be noted that FGR provides a poor accuracy compared to all the other techniques, proving inadequate for the type of problem studied. The reason for this difference may be the greater generality of the problem addressed, compared to the conditions tested in [34], in which the performance of FGR was evaluated on partially overlapping surfaces.

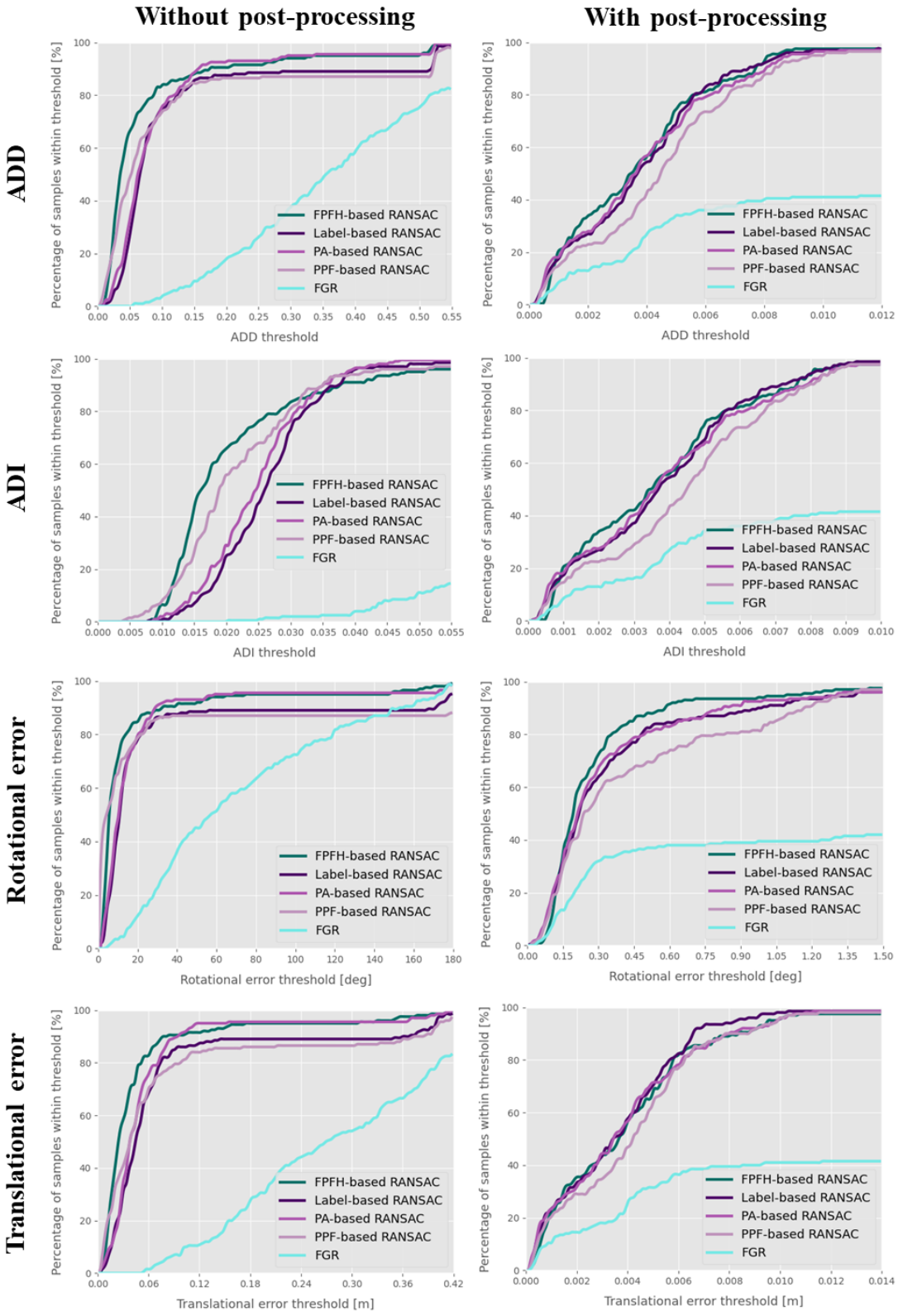

Figure 5.1 - Comparison results between the algorithms. The metrics compared are ADD, ADI (top 4 plots), rotational error and translational error (bottom 4 plots), without post-processing (left column) and with post-processing (right column)

As for the RANSAC-based algorithms, the plots in the left column show a slight overall superiority of FPFH-based RANSAC. Furthermore, in the ADD plot, a jump in the percentage of samples when the ADD threshold gets slightly higher than 0.5 can be observed; this is caused by the flipped poses whose samples are included when this threshold on ADD is exceeded. This behavior is also visible in the plots of the translational and rotational errors without post-processing; in particular, in the one of the rotational errors it is observed right before 180°, confirming the previous observation. On the other hand, the plots in the right column clearly show the significant improvement in the accuracy of the results due to the post-processing: indeed, assuming that for $\phi_{err} < 5°$ and $t_{err} < 5\ cm$ the pose is correctly estimated, the simulations performed resulted in a SR of 98%, 98.5%, 98.5% and 98.5%, for FPFH-based RANSAC, Label-based RANSAC, PA-based RANSAC and PPF-based RANSAC, respectively, thus proving that the developed HT-based methods represent promising alternatives. It is worth highlighting that the addition of flip checks in the post-processing block contributed to the performance improvement of the FPFH-based RANSAC method: in fact, through simulations with and without flip checks, a difference in SR of 4.5% has been observed, therefore 93.5% in the case without flips and, as already mentioned, 98% in the case with flips, at the expense of an increase in the mean computational time of $0.067\ s$. Finally, the computational times for both cases without and with post-processing are reported in Table 5.6.

Method	***Computational time without post-processing [s]***		***Computational time with post-processing [s]***	
	Mean	***Median***	***Mean***	***Median***
FPFH-based RANSAC	0.0965	0.0950	0.2511	0.2060
Label-based RANSAC	0.3062	0.2136	0.4610	0.3309
PA-based RANSAC	0.4441	0.3693	0.5331	0.4357
PPF-based RANSAC	0.1098	0.1084	0.2420	0.1776
FGR	0.0237	0.0230	0.2295	0.1670

Table 5.6 - Time performance

From Table 5.6, it can be observed that the Label-based and PA-based RANSAC algorithms are less performant than FPFH-based RANSAC and PPF-based RANSAC, which instead have comparable computational times. It is important to underline that the presented HT-based

implementations are not optimized, unlike FPFH-based RANSAC, whose implementation comes from the Open3D library. Therefore, this comparison is to be considered as preliminary.

5.2.2. Comparison of Algorithm Variants

In this Subsection, a series of comparisons between variants of the developed algorithms, anticipated in Chapter 3, are shown. These variants present different approaches for offline HT construction methodology (paragraph 5.2.2.1) and alignment quality assessment (paragraph 5.2.2.2) and are tested on Label-based and PA-based RANSAC methods.

5.2.2.1. Training Hash Table vs Model Hash Table

This paragraph shows the comparison analysis between training and model HTs approaches. The tuning parameters for constructing the model HTs are the same as in Table 5.1, considering only the $\boldsymbol{HT_{CS}}$ for Label and PA-based while, to manage the FS scans, the training HT is used.

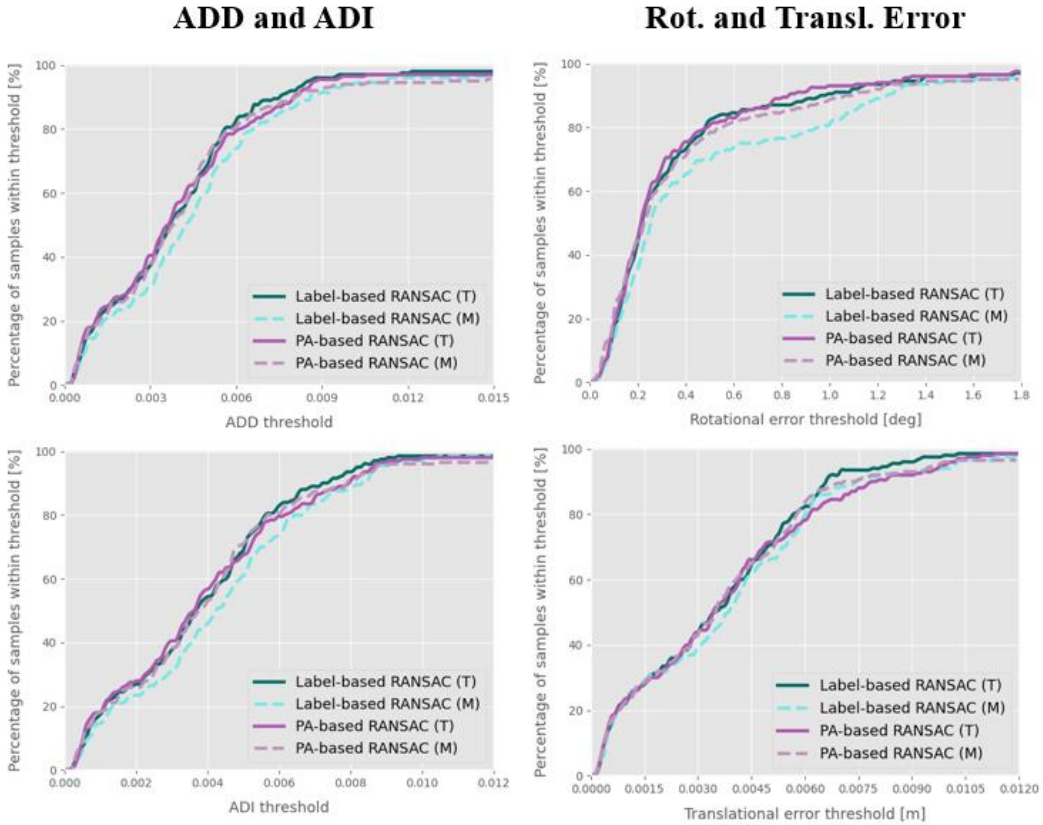

Figure 5.2 - Comparison of Label and PA-based RANSAC using training and model HT approaches, with post-processing. Left: ADD, ADI. Right: rotational and translational errors

Figure 5.2 shows the comparison between the two approaches for Label-based RANSAC and PA-based RANSAC algorithms. *T* indicates the approach that uses the training $\boldsymbol{HT_{CS}}$, while *M* indicates the approach that uses the model $\boldsymbol{HT_{CS}}$. Both these variants are compared using the *NN approach* for alignment quality assessment. The performances of the two variants are only compared with post-processing. The limits of the horizontal axis are set to $[0, p]$, with p such that 95% of the samples of the testing dataset are reached with the worst method, instead of 96%, so that the curves can be distinguished a little more clearly. From plots shown in Figure 5.2, it can be observed that the two variants are very comparable in the case of the PA-based RANSAC method, but with a clear advantage for the training HT variant in the case of the Label-based RANSAC one. The simulations performed resulted in a SR of 98.5%, 98.5%, 98.5% and 96.5%, for Label-based RANSAC (T), Label-based RANSAC (M), PA-based RANSAC (T) and PA-based RANSAC (M), respectively. Runtimes are shown in Table 5.7.

Method		***Computational Time with Post-Processing [s]***	
		Mean	***Median***
Label-based RANSAC	***T***	0.4610	0.3309
	M	0.4393	0.3451
PA-based RANSAC	***T***	0.5331	0.4357
	M	0.7705	0.6275

Table 5.7 – Runtimes of Label-based and PA-based RANSAC, with training and model HTs

Table 5.7 shows comparable computational times, except for the PA-based RANSAC method with the model HT, which is slower.

5.2.2.2. Nearest Neighbor vs Binary Matching

In this paragraph, the comparison analysis between NN and BM approaches is shown. Table 5.8 shows the values of the parameters used to build $\boldsymbol{HT_{voxel}}$, defined in paragraph 3.5.4.2.

$\boldsymbol{HT}$	$\boldsymbol{m}$	$\boldsymbol{D}$ $[\boldsymbol{m}]$
$\boldsymbol{HT_{voxel}}$	12	1.2

Table 5.8 - Parameters selected for $\boldsymbol{HT_{voxel}}$

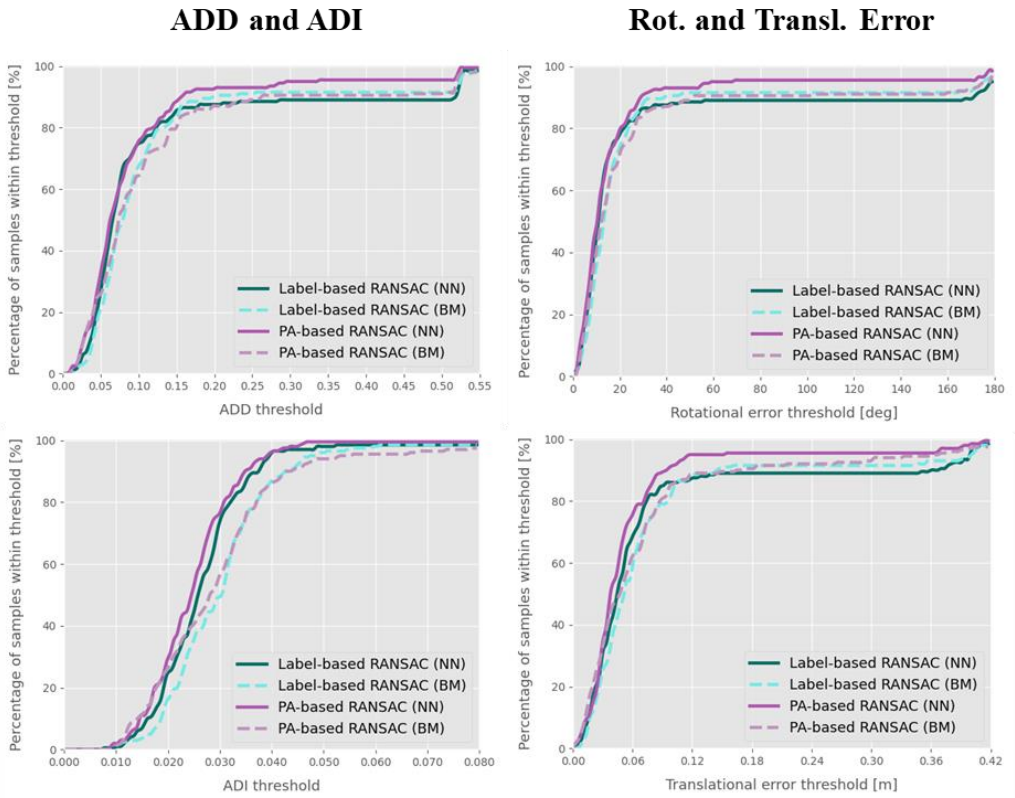

Figure 5.3 - Comparison of Label and PA-based RANSAC using NN and BM approaches, without post-processing. Left: ADD, ADI. Right: rotational and translational errors

Figure 5.3 compares the two approaches for Label-based and PA-based RANSAC algorithms. *NN* refers to the Nearest Neighbors, while *BM* refers to Binary Matching. Both these variants are compared using the *training HTs*. The performance of the two variants is compared without post-processing, as flip check evaluation in post-processing has only been implemented using the NN approach. The limits of the horizontal axis are set to $[0, p]$, with p such that 96% of the samples of the testing dataset are reached with the worst method.

Method		***Computational Time without Post-Processing [s]***	
		Mean	***Median***
Label-based RANSAC	***NN***	0.3062	0.2136
	BM	0.2474	0.1914
PA-based RANSAC	***NN***	0.4441	0.3693
	BM	0.3688	0.3363

Table 5.9 – Runtimes of Label-based and PA-based RANSAC, with NN and BM approaches

Runtimes are shown in Table 5.9. From Figure 5.3 and Table 5.9 it can be observed that the BM approach for alignment quality evaluation represents a very promising alternative, especially for the lower computational time. It could be interesting to investigate the integration of a post-processing phase where the quality of flip checks is evaluated using such an approach instead of the NN-based one.

5.2.3. Performance of Open3D Algorithms without and with Downsampling

As already mentioned in paragraph 5.2.1.1, further simulations have been performed with FPFH-based RANSAC and FGR without downsampling, in order to compare their performances with the downsampled case. In Figure 5.4 the plots of ADD, ADI, rotational and translational error with post-processing are shown.

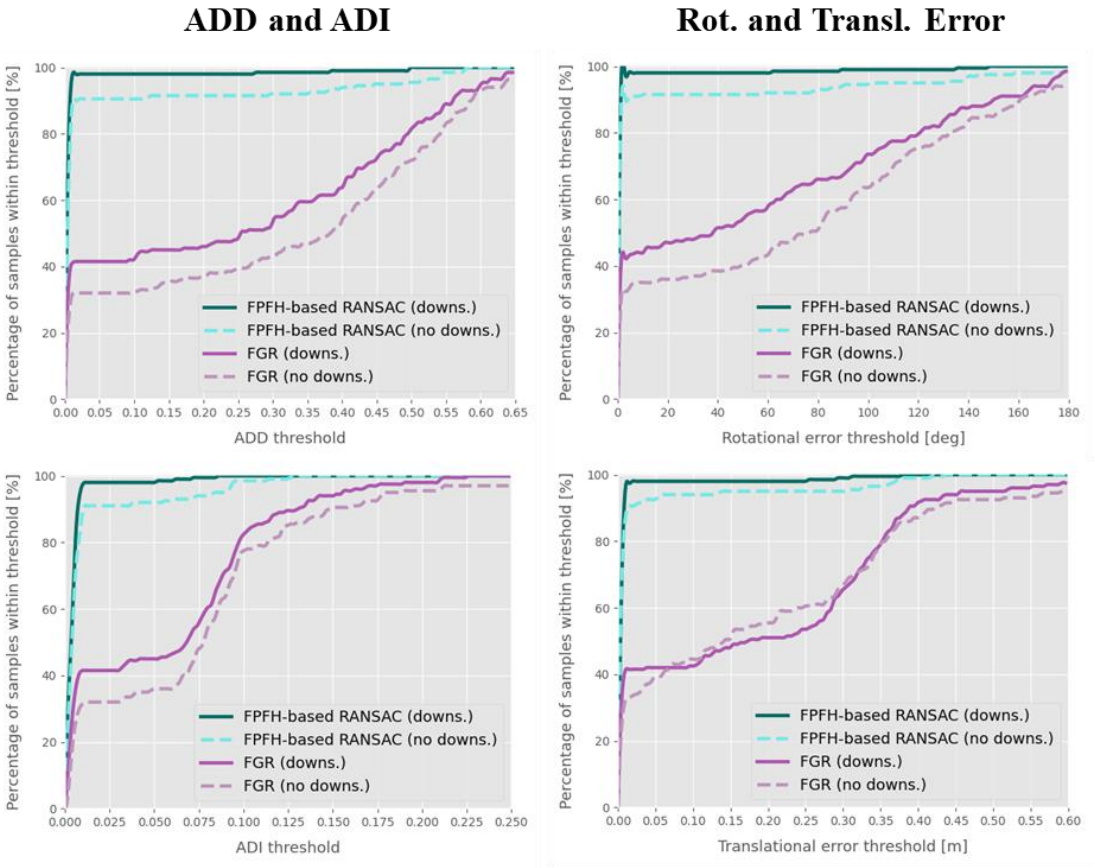

Figure 5.4 – Comparison of Open3D algorithms with downsampling (downs.) and without downsampling (no downs.), with post-processing. Left: ADD, ADI. Right: rotational and translational errors

The limits of the horizontal axis are set to $[0, p]$, with p such that 96% of the samples of the testing dataset are reached with the worst method. Runtimes are shown in Table 5.10.

Method		***Computational Time with Post-Processing [s]***	
		Mean	***Median***
FPFH-based RANSAC	***Downs.***	0.2511	0.2060
	No downs.	1.3757	1.0095
FGR	***Downs.***	0.2295	0.1670
	No downs.	2.4265	1.7824

Table 5.10 – Runtimes of Open3D algorithms, without and with downsampling

From Figure 5.4 and Table 5.10, it is possible to observe the evident superiority, in terms of accuracy and computational time, of the Open3D algorithms with preliminary downsampling applied to the measured and model point clouds.

5.3. Autonomous Failure Detection

The last analysis carried out during this work is the development of an Autonomous Failure Detection (AFD) algorithm which, using:

1. The rotational and translational errors ϕ_{err} and t_{err}, through which the SR is defined;
2. The convergence value of the ICP cost function f_{end}, i.e. the minimum sum of squared distances to closest planes;

is able to autonomously declare the success or failure in the pose estimation. This is possible through the computation of the optimal value of a reference variable f_τ, which is compared with f_{end}; in fact, if the inequality shown in Equation 5.6 is verified:

$$f_{end} < f_\tau \tag{5.6}$$

the algorithm declares the success in the pose estimation, otherwise it declares the failure.

To determine the optimal value of f_τ, a statistical approach is used that makes use of probabilities calculated from the experimental results obtained.

First of all, it is necessary to define the success probability P_s and failure probability P_F of the algorithm which, starting from the results obtained, are defined as shown in Equation 5.7 and Equation 5.8:

$$P_S = \frac{\#\ correct\ poses}{\#\ total\ poses} \tag{5.7}$$

$$P_F = \frac{\#\ incorrect\ poses}{\#\ total\ poses} \tag{5.8}$$

So, considering the following events:

- *A*: the AFD algorithm declares a success, i.e., $f_{end} < f_\tau$;
- *B*: the pose is successfully computed, i.e., $(\phi_{err} < 5°)\ \cap\ (t_{err} < 0.05\ m)$.

four conditional probabilities are defined:

- P_{SS} as the probability that the AFD algorithm declares a success given the success in pose estimation, which is computed through Equation 5.9:

$$P_{SS} = P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{5.9}$$

- P_{SF} as the probability that the AFD algorithm declares a success given the failure in pose estimation, which is computed through Equation 5.10:

$$P_{SF} = P(A|\bar{B}) = \frac{P(A \cap \bar{B})}{P(\bar{B})} \tag{5.10}$$

- P_{FS} as the probability that the AFD algorithm declares a failure given the success in pose estimation, which is computed through Equation 5.11:

$$P_{FS} = P(\bar{A}|B) = \frac{P(\bar{A} \cap B)}{P(B)} \tag{5.11}$$

- P_{FF} as the probability that the AFD algorithm declares a failure given the failure in pose estimation, which is computed through Equation 5.12:

$$P_{FF} = P(\bar{A}|\bar{B}) = \frac{P(\bar{A} \cap \bar{B})}{P(\bar{B})} \tag{5.12}$$

Given this premise, the AFD algorithm is based on calculating these conditional probabilities on a vector of f_τ: it is varied from a very small value, such that every pose is declared a failure, to a very large value, such that every pose will be declared a success. In this way it is possible to study the behaviour of the four conditional probabilities just defined as a function of f_τ, so as to be able to select the value that optimally distinguishes successes from failures.

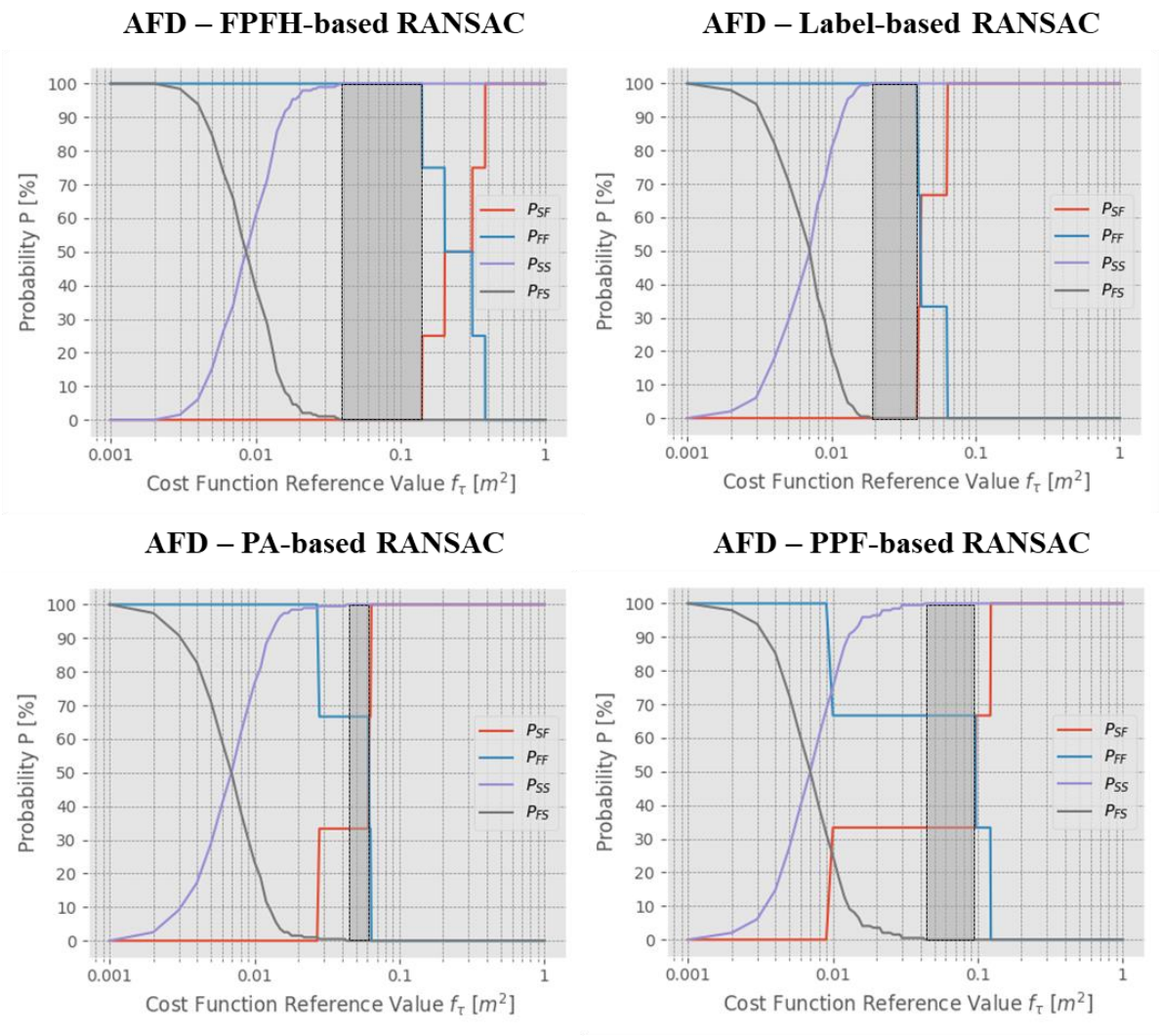

Figure 5.5 – AFD results. Highlighted are the ranges of f_τ such that the probability P_T that the AFD algorithm tells the truth is maximum

Figure 5.5 shows the AFD results for the main variants of FPFH-based RANSAC, Label-based, PA-based and PPF-based RANSAC, respectively (Subsection 5.2.1). The potential values of f_τ to be selected can be deduced identifying at which f_τ there is the maximum probability P_T that the AFD algorithm is telling the truth. This probability is computed as shown in Equation 5.13:

$$P_T = P_{SS}\,P(B) + P_{FF}\,P(\bar{B}) = P((A \cap B) \cup (\bar{A} \cap \bar{B})) \tag{5.13}$$

The stepped behavior in the P_{SF} and P_{FF} plots is due to the low number of failures in the testing dataset: indeed, on a set of 200 scans, 98.5% of SR corresponds to 3 failures, and this explains the corresponding number of steps in the related plots. Because of this, it is not possible to identify an exact optimal value of f_τ, but it is possible to indicate a range of optimal values that can be selected. These ranges for the four plots are shown in Table 5.11.

Method	***Optimal f_τ values***	
	Min $[m^2]$	***Max*** $[m^2]$
FPFH-based RANSAC	0.038	0.14
Label-based RANSAC	0.019	0.039
PA-based RANSAC	0.043	0.061
PPF-based RANSAC	0.043	0.096

Table 5.11 – Optimal f_τ values resulting from AFD analysis

6. Conclusions and Future Works

In this thesis work, three feature-based algorithmic solutions for LiDAR-based pose acquisition of known non-cooperative spacecraft have been presented, which exploit point-normal structures as local features (FPFH) or as non-local primitives (PPF) and a RANSAC-based approach to find point correspondences between a model and a data point cloud, supported by the use of Hash Tables appropriately built offline for the rapid online retrieval of potential good point correspondences.

After a brief introduction on the space scenarios relevant to this challenge and an overview of the LiDAR sensors central to this discussion, a literature review has been presented, focusing on the current state-of-the-art in classical global registration methods (i.e. non-learning-based) to evaluate their advantages and disadvantages and identify the most promising techniques to focus further research on. Then, several analyses have been performed, in Python environment, focused on the effective extraction of FPFHs, exploiting the segmentation of the reference geometry in geometric primitives and the Persistence Analysis, key principles of the Label-based RANSAC and PA-based RANSAC methods, respectively, and on the construction of Hash Tables, used for Label-based, PA-based and PPF-based RANSAC. These analyses supported the development of the above mentioned pose estimation algorithms, refined through Iterative Closest Point (ICP) and additional checks to correct "ambiguous" poses. The performances of the developed algorithms have then been compared with well-known feature-based approaches implemented in the Python Open3D Library: FPFH-based RANSAC and Fast Global Registration (FGR). Performance assessment has been carried out using a dataset of synthetic point clouds. Finally, an autonomous failure detection strategy has been described to enhance the robustness of the proposed architectures.

From the obtained results, reported in Chapter 5, the following conclusions can be drawn. All the developed variants are promising methods for solving the task in the context of orbital servicing missions. A slight advantage in accuracy and speed has been observed for FPFH-based RANSAC. The speed measurements, however, are just preliminary as not all methods currently have an optimized implementation.

Regarding FGR, instead, which approaches pose estimation as a global optimization without relying on strict point correspondences, has proven to be significantly inferior and unsuitable for the task.

Moreover, significant accuracy improvements have been demonstrated by postprocessing the results of global pose estimation using ICP-based refinement and checking for flip errors caused by pose ambiguities in the LiDAR data under certain conditions.

As for the variants of the developed algorithms, in Label-based and PA-based RANSAC a higher overall efficiency in building hash tables using scans of the training dataset instead of the whole model point cloud has been found, and for this reason this variant has not been implemented also in PPF-based RANSAC, while the alignment evaluation approach using the 3D voxel grid turned out to be a very promising alternative to the approach that computes inliers via nearest neighbor retrieval, and that is worth to be further investigated, implementing it also in PPF-based RANSAC.

It is important to note that, although all three developed HT-based methods can be generalized to satellites with different geometries, the current Label-based RANSAC method relies exclusively on the geometric primitives present in the DLR Client Satellite, i.e. toroids, edges, spheres, handles, cylinders and planes, using FPFH-based signatures of these geometries, computed using the training dataset. A potential and promising generalization of the Label-based RANSAC method could involve an algorithm capable of creating and identifying clusters of regions characterized by similar FPFHs. This approach would represent a more functional and advantageous extension of the current method, as it eliminates the need for initial pre-processing to compute averaged FPFHs for recognition. More importantly, this would enable the method to be rapidly extended to targets with arbitrary geometries, significantly enhancing its versatility.

Building on this concept, future research should aim to consolidate and extend the results obtained, incorporating improvements in data realism or testing on real, noisy LiDAR data, accounting for varying sensor characteristics, and including additional satellite models.

References

[1] Kessler, D. J., Johnson, N. L., Liou, J. C., and Matney, M., "The Kessler Syndrome: Implications to Future Space operations," *Advances in the Astronautical Sciences,* Vol. 137, 2010.

[2] Flores-Abad, A., Ma, O., Pham, K., and Ulrich, S., "A review of space robotics technologies for on-orbit servicing", *Progress in Aerospace Sciences,* Vol. 68, 2014, pp. 1-26.

[3] Bonnal, C., Ruault, J. M., and Desjean, M. C., "Active debris removal: Recent progress and current trends", *Acta Astronautica,* Vol. 85, 2013, pp. 51-60.

[4] Opromolla, R., Fasano, G., Rufino, G., and Grassi, M., "A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations," *Progress in Aerospace Sciences,* Vol. 93, 2017, pp. 53-72.

[5] D'Amico, S., Benn, M., and Jørgensen, J. L., "Pose estimation of an uncooperative spacecraft from actual space imagery," *International Journal of Space Science and Engineering,* Vol. 2, No. 2, 2014, pp. 171-189.

[6] Yu, F., He, Z., Qiao, B., and Yu, X., "Stereo-Vision-Based Relative Pose Estimation for the Rendezvous and Docking of Noncooperative Satellites," *Mathematical Problems in Engineering,* Vol. 2014, No. 1, 2014, pp. 1-12.

[7] Pyrak, M., and Anderson, J., "Performance of Northrop Grumman's Mission Extension Vehicle (MEV) RPO imagers at GEO," *Autonomous Systems: Sensors, Processing and Security for Ground, Air, Sea and Space Vehicles and Infrastructure 2022,* Vol. 12115, 2022, pp. 64-82.

[8] English, C., Zhu, S., Smith, C., Ruel, S., and Christie, I., "TriDAR: A hybrid sensor for exploiting the complimentary nature of triangulation and LIDAR technologies," *Proceedings of the 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space,* Vol. 1, 2005.

[9] Christian, J., and Cryan, S., "A Survey of LIDAR Technology and its Use in Spacecraft Relative Navigation," *AIAA Guidance, Navigation, and Control (GNC) Conference,* 2013.

[10] Opromolla, R., Fasano, G., Rufino, G., and Grassi, M, "A Model-Based 3D Template Matching Technique for Pose Acquisition of an Uncooperative Space Object," *Sensors*, Vol. 15, No. 3, 2015, pp. 6360-6382.

[11] Li, L., Wang, R., and Zhang, X., "A Tutorial Review on Point Cloud Registrations: Principle, Classification, Comparison, and Technology Challenges," *Mathematical Problems in Engineering*, Vol. 2021, 2021.

[12] Monji-Azad, S., Hesser, J., and Löw, N., "A review of non-rigid transformations and learning-based 3D point cloud registration methods," *ISPRS Journal of Photogrammetry and Remote Sensing*, Vol. 196, 2023, pp. 58-72.

[13] Yang, J., Li, H., Campbell, D., and Jia, Y., "Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 38, No. 11, 2016, pp. 2241-2254.

[14] Myronenko, A., and Song, X., "Point Set Registration: Coherent Point Drift," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 32, No. 12, 2010, pp. 2262-2275.

[15] Biber, P., and Strasser, W., "The normal distributions transform: a new approach to laser scan matching," P*roceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, Vol. 3, 2003, pp. 2743-2748.

[16] Lamdan, Y., and Wolfson, H. J., "Geometric Hashing: A General And Efficient Model-based Recognition Scheme," *[1988 Proceedings] Second International Conference on Computer Vision*, 1988, pp. 238-249.

[17] Prokudin, S., Lassner, C., and Romero, J., "Efficient Learning on Point Clouds with Basis Point Sets," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 4331–4340.

[18] Ruel, S., Luu, T., Anctil, M., and Gagnon, S., "Target Localization from 3D data for On-Orbit Autonomous Rendezvous & Docking," *2008 IEEE Aerospace Conference*, 2008, pp. 1-11.

[19] Yin, F., Chou, W., Wu, Y., Yang, G., and Xu, S., “Sparse Unorganized Point Cloud Based Relative Pose Estimation for Uncooperative Space Target,” *Sensors*, Vol. 18, No. 4, 1009, 2018.

[20] Wahl, E., Hillenbrand, U., and Hirzinger, G., “Surflet-pair-relation histograms: a statistical 3D-shape representation for rapid classification,” *Fourth International Conference on 3-D Digital Imaging and Modeling*, 2003, pp. 474-481.

[21] Rusu, R. B., Márton, Z., Blodow, N., and Beetz, M., “Persistent Point Feature Histograms for 3D Point Clouds,” *Proceedings of the 10th International Conference on Intelligent Autonomous Systems (IAS-10)*, 2008, pp. 119-128.

[22] Rusu, R. B., Blodow, N., and Beetz, M, “Fast Point Feature Histograms (FPFH) for 3D registration,” *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 3212-3217.

[23] Aldoma, A., Vincze, M., Blodow, N., Gossow, D., Gedikli, S., Rusu, R. B., and Bradski, G., “CAD-model recognition and 6DOF pose estimation using 3D cues,” *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 2011, pp. 585-592.

[24] Aldoma, A., Tombari, F., Rusu, R. B., and Vincze, M., “OUR-CVFH -- Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation,” *Pattern Recognition*, 2012, pp. 113-122.

[25] Opromolla, R., Fasano, G., Rufino, G., and Grassi, M, “Uncooperative pose estimation with a LIDAR-based system,” *Acta Astronautica*, Vol. 110, 2015, pp. 287-297.

[26] Nocerino, A., Saggiomo, F., Piatti, S., Fasano, G., Grassi, M., Opromolla, R., and Schmitt, C., “Numerical and Experimental Validation of LIDAR-based Template Matching Algorithms for Non- Cooperative Spacecraft Pose Initialization,” 2023, pp. 1-20.

[27] Guo, W., Hu, W., Liu, C., and Lu, T., “Pose Initialization of Uncooperative Spacecraft by Template Matching with Sparse Point Cloud,” *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 9, 2021, pp. 1707-1720.

[28] Opromolla, R., Fasano, G., Rufino, G., and Grassi, M, "Pose Estimation for Spacecraft Relative Navigation Using Model-Based Algorithms," *IEEE Transactions on Aerospace and Electronic Systems,* Vol. 53, No. 1, 2017, pp. 431-447.

[29] Barequet, G., "8 - Geometric Hashing and Its Applications," *Database and Data Communication Network Systems,* 2002, pp. 277-287.

[30] Verreault, S., Laurendeau, D., and Bergevin, R., "Pose determination for an object in a 3-D image using geometric hashing and the interpretation tree," *Proceedings of Canadian Conference on Electrical and Computer Engineering,* Vol. 2, 1993, pp. 755-758.

[31] Li-Chee-Ming, J., and Armenakis, C., "Feasibility Study for Pose Estimation of Small UAS in Known 3D Environment Using Geometric Hashing," *Photogrammetric Engineering and Remote Sensing,* Vol. 80, 2014, pp. 1117-1128.

[32] Fischler, M. A., and Bolles, R. C., "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," *Readings in Computer Vision,* 1987, pp. 726-740.

[33] Wei, L., Hongtai, C., and Zhang, X., "Efficient 3D Object Recognition from Cluttered Point Cloud," *Sensors,* Vol. 21, No. 17, 5850, 2021.

[34] Zhou, Q. Y., Park, J., and Koltun, V., "Fast Global Registration," *Computer Vision - ECCV 2016,* 2016, pp. 766–782.

[35] Black, M. J., Rangarajan, A., "On the unification of line processes, outlier rejection, and robust statistics with applications in early vision," *International Journal of Computer Vision,* Vol. 19, No. 1, 1996, pp. 57-91.

[36] Shlens, J., "A Tutorial on Principal Component Analysis," *ArXiv,* Vol. abs/1404.1100, 2014.

[37] Rusu, R. B., Bradski, G., Thibaux, R., and Hsu, J., "Fast 3D recognition and pose using the Viewpoint Feature Histogram," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems,* 2010, pp. 2155-2162.

[38] Rhodes, A., Kim, E., Christian, J. A., and Evans, T., “LIDAR-based Relative Navigation of Non-Cooperative Objects Using Point Cloud Descriptors,” *AIAA/AAS Astrodynamics Specialist Conference*, 2016.

[39] Rhodes, A., Christian, J. A., and Evans, T., “A Concise Guide to Feature Histograms with Applications to LIDAR-Based Spacecraft Relative Navigation,” *The Journal of the Astronautical Sciences*, Vol. 64, 2017, pp. 414–445.

[40] Rusinkiewicz, S., and Levoy, M., “Efficient variants of the ICP algorithm,” *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, 2001, pp. 145-152.

[41] Birdal, T., and Ilic, S., “Point Pair Features Based Object Detection and Pose Estimation Revisited,” *2015 International Conference on 3D Vision*, 2015, pp. 527-535.

[42] Hinterstoisser, S., Lepetit, V., Rajkumar, N., and Konolige, K., “Going Further with Point Pair Features,” *Computer Vision – ECCV 2016*, 2016, pp. 834–848.

[43] Vidal, J., Lin, C. Y., and Martí, R., “6D Pose Estimation using an Improved Method based on Point Pair Features,” *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*, 2018, pp. 405-409.

[44] Artigas, J., De Stefano, M., Rackl, W., Lampariello, R., Brunner, B., Bertleff, W., Burger, R., Porges, O., Giordano, A., Borst, C., and Albu-Schaeffer, A., “The OOS-SIM: An on-ground simulation facility for on-orbit servicing robotic operations,” *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 2854-2860.

[45] Shoemake, K., “III.6 - UNIFORM RANDOM ROTATIONS,” *Graphics Gems III (IBM Version)*, 1992, pp. 124-132.

[46] Zhou, Q. Y., Park, J., and Koltun, V., "Open3D: A Modern Library for 3D Data Processing", *ArXiv preprint arXiv:1801.09847*, 2018.

[47] Moore, A., W., “An Intoductory Tutorial on Kd-trees Extract from Andrew Moore's Phd Thesis: Eecient Memory-based L Earning for Robot Control,” 1991

[48] Ran, L., Wanggen, W., Yiyuan, Z., Libing, L., and Ximin, Z., “Normal estimation algorithm for point cloud using KD-Tree,” *IET International Conference on Smart and Sustainable City 2013 (ICSSC 2013)*, 2013, pp. 334-337.

[49] Estimating Normals for Point Clouds - Point Cloud Utils.

[50] Arun, K. S., Huang, T. S., and Blostein, S. D., "Least-Squares Fitting of Two 3-D Point Sets", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 5, 1987, pp. 698-700.

[51] Piccinin, M., and Hillenbrand, U., "Deep Learning-Based Pose Regression for Satellites: Handling Orientation Ambiguities in LiDAR Data", *Journal of image and graphics - in print*.

[52] Hodaň, T., Matas, J., and Obdržálek, Š., "On Evaluation of 6D Object Pose Estimation," *Computer Vision – ECCV 2016 Workshops*, 2016, pp. 606-619.

Acknowledgements

With these pages my master's degree comes to an end, an extraordinary adventure that I have lived surrounded by many people who have supported me and without whom none of this would have been possible, and therefore I would like to deeply thank them because they are part of this goal.

On the Federico II side, I thank my supervisors, Professors Michele Grassi and Roberto Opromolla, and my co-supervisor, Ing. Alessia Nocerino, for the great support provided to the thesis work during my time at DLR, which, despite the distance, was absolutely indispensable.

Of course, I thank my co-supervisors of the DLR, Dr. Margherita Piccinin and Dr. Ulrich Hillenbrand, for having always supported and assisted me throughout the thesis period in the institute, analyzing and facing together all the difficulties encountered along the way with enlightening advice and, therefore, guiding the work with enormous availability and professionalism. I am grateful to have lived this experience with you because it has enriched me a lot, from a professional and personal point of view, and represents a piece of my life that I will remember with great happiness, for the satisfactions and teachings it left me.

I dedicate a deep and heartfelt thank you to my university classmates, for making these years unforgettable. I thank Antonio Sodano and Giuseppe Puleo, friends I now seem to have known for a lifetime, for the strength of the bond that has united us over the years. Thank you so much for spending every year together. I thank Davide Albanese, Anton Matacera, Francesco D'Antuono, Vittorio di Napoli, Gianluca Montuori, Simona Morra, Antonio Napolitano, Vincenzo Lioncino, Antonio Russano, Luigi D'Arco, Alida Sanzari and Flavia Migliaccio. Studying together was fun, but at the same time something highly instructive, since I had the opportunity to learn a lot from each of you. Although I have known some a little less than others, I have still had the opportunity to discover fantastic people who are now special to me; You were all inspiring, super capable, determined, ambitious, always ready to lend a hand, I really couldn't hope for a better company than yours. Thank you guys from the bottom of my heart, thank you for being a piece of my life. I really hope that our paths will continue to intertwine in the future.

Leaving the university sphere, I dedicate a special thanks to my "historical" friends and companions of adventures and outings: Giuseppe, a friend from middle school and fellow engineer, with whom I often engage in interesting university conversations, Christian, a friend for several years, an extremely sharp person, with whom I talk about everything and towards whom I hold the record of the longest voice message ever sent so far, Domenico and Lorenzo, friends literally for a lifetime, since elementary school, so there is little to say, we practically grew up together, and finally Lena, the most recent friend, a person of great kindness and sensitivity. I thank you guys deeply, for helping me, practically all 5 years, to disconnect my brain when necessary, not to think about my mountain of commitments, anxieties, for always listening to my outbursts, for video calls during my stay in Germany to make me feel your company. You are special companions and know that, even if perhaps unconsciously, you have had a fundamental impact on me over the years, and for this I really thank you from the bottom of my heart.

I thank Carolyn, whom I met by surprise in Germany during my thesis period! I have never properly thanked you for the moral support you gave me during my stay in Germany, I deeply appreciated your support and the outings together that helped me to disconnect a little from thoughts at work.

I deeply thank all my relatives, my grandparents, my uncles and my cousins, for never missing the opportunity to ask me about my university exams, about my path, and congratulate me on the goals achieved. All the warmth you have given me during this period has been a great help in spurring me on.

I thank Luca Mercurio who, in addition to being a great addition to our family, has contributed to making the long afternoons of study more bearable with his contagious joy.

But last to thank I left the most important, my parents and my sister Francesca, my family. I don't know where to start to thank you for everything you have done for me, simple words on a thesis page are not enough to express all my gratitude to you. You have been next to me in every moment, in the best moments, of happiness for the results achieved in the exams, living together those moments of satisfaction, but above all in the worst moments, of crisis and stress, always spending many words of support and showing all the possible and imaginable closeness, living together those moments of suffering. In Germany I felt that I had always had you next to

me, with many calls, during every moment of the day, during breaks from work, during my free moments and even during home activities! As simple and trivial as it may seem to you, I assure you that for me it was not at all, because it helped me immensely to spend that period without significantly missing home.

I dedicate these last lines to you. Simply Thank you. Thank you mom and dad, Thank you Francesca, for being my point of reference, for being part of my soul.

Thanks for everything.

Ringraziamenti

Con queste pagine giunge al termine il mio percorso di laurea magistrale, un'avventura straordinaria che ho vissuto circondato da molte persone che mi hanno sostenuto e senza i quali nulla di tutto questo sarebbe stato possibile, e che dunque desidero profondamente ringraziare perché parte di questo traguardo.

Ringrazio, dal lato della Federico II, i miei relatori, i Professori Michele Grassi e Roberto Opromolla, e la mia correlatrice, l'Ing. Alessia Nocerino, per il grande sostegno fornito al lavoro di tesi durante il mio periodo al DLR, il quale, nonostante la distanza, è stato assolutamente indispensabile.

Un grazie va naturalmente ai miei correlatori del DLR, la Dr. Margherita Piccinin e il Dr. Ulrich Hillenbrand, per avermi sempre supportato e assistito durante tutto il periodo di tesi in istituto, analizzando e affrontando insieme tutte le difficoltà incontrate lungo il lavoro con consigli illuminanti e, dunque, guidando il lavoro con enorme disponibilità e professionalità. Sono grato di aver vissuto questa esperienza con voi perché mi ha arricchito moltissimo, dal punto di vista professionale e personale, e rappresenta un tassello della mia vita che ricorderò con molta felicità, per le soddisfazioni e gli insegnamenti che mi ha lasciato.

Dedico un profondo e sentito grazie ai miei compagni di università, per aver reso questi anni indimenticabili. Ringrazio Antonio Sodano e Giuseppe Puleo, amici che ormai mi sembra di conoscere da una vita, per la forza del legame che ci ha unito in questi anni. Grazie di cuore per aver trascorso ogni anno insieme. Ringrazio Davide Albanese, Anton Matacera, Francesco D'Antuono, Vittorio di Napoli, Gianluca Montuori, Simona Morra, Antonio Napolitano, Vincenzo Lioncino, Antonio Russano, Luigi D'Arco, Alida Sanzari e Flavia Migliaccio. Studiare insieme è stato uno spasso, ma allo stesso tempo qualcosa di fortemente istruttivo, dato che da ciascuno di voi ho avuto modo di imparare moltissimo. Sebbene abbia conosciuto alcuni un po' meno di altri, ho comunque avuto modo di scoprire delle persone fantastiche e oramai per me speciali; siete stati tutti d'ispirazione, super capaci, determinati, ambiziosi, sempre pronti a dare una mano, davvero non potevo sperare in una compagnia migliore della vostra. Grazie di cuore a tutti voi ragazzi, grazie di essere stati un pezzo della mia vita. Spero tanto che le nostre strade continuino a intrecciarsi in futuro.

Uscendo dalla sfera universitaria, dedico un grazie speciale i miei amici "storici" e compagni di avventure e uscite: Giuseppe, amico dalle medie e collega ingegnere, con cui spesso ingaggio interessanti conversazioni universitarie, Christian, amico da diversi anni, persona estremamente acuta, con cui parlo di tutto e nei confronti di cui detengo il record del messaggio vocale più lungo mai inviato finora, Domenico e Lorenzo, amici letteralmente da una vita, dalle elementari, quindi c'è poco da dire, siamo praticamente cresciuti insieme, e infine Lena, l'amica più recente, una persona di grandissima gentilezza e sensibilità. Vi ringrazio profondamente ragazzi, per avermi aiutato, praticamente tutti e 5 gli anni, a staccare il cervello quando necessario, per non pensare alla mia montagna di impegni, alle ansie, per aver ascoltato sempre i miei sfoghi, per le videochiamate durante la mia permanenza in Germania per farmi sentire la vostra compagnia. Siete dei compagni speciali e sappiate che, anche se magari inconsapevolmente, avete avuto un impatto fondamentale su di me in questi anni, e per questo vi ringrazio davvero di cuore.

Ringrazio Carolyn, incontrata a sorpresa in Germania durante il mio periodo di tesi! Non ti ho mai ringraziato a dovere per il sostegno morale che mi hai dato durante la mia permanenza in Germania, ho profondamente apprezzato il tuo supporto e le uscite insieme che mi hanno aiutato a staccare un po' dai pensieri sul lavoro.

Ringrazio profondamente tutti i miei parenti, i miei nonni, i miei zii e i miei cugini, per non aver mai perso l'occasione di chiedermi degli esami universitari, del mio percorso, e farmi i complimenti per i traguardi raggiunti. Tutto il calore che mi avete dato in questo periodo è stato di grandissimo aiuto per spronarmi.

Ringrazio Luca Mercurio che, oltre a essere una grande aggiunta alla nostra famiglia, ha contribuito a rendere più sopportabili i lunghi pomeriggi di studio con la sua contagiosa allegria.

Ma per ultimi da ringraziare ho lasciato i più importanti, i miei genitori e mia sorella Francesca, la mia famiglia. Non saprei da dove partire per ringraziarvi per tutto quello che avete fatto per me, delle semplici parole in una pagina di tesi non sono sufficienti per esprimere tutta la mia gratitudine nei vostri confronti. Siete stati accanto a me in ogni momento, nei momenti migliori, di felicità per i risultati raggiunti agli esami, vivendo insieme quei momenti di soddisfazione, ma soprattutto nei momenti peggiori, di crisi e stress, spendendo sempre tante parole di supporto e mostrando tutta la vicinanza possibile e immaginabile, vivendo insieme quei momenti di

sofferenza. In Germania ho sentito di avervi sempre avuti affianco a me, con tantissime chiamate, durante ogni istante della giornata, durante le pause da lavoro, durante i miei momenti liberi e persino durante le attività casalinghe! Per quanto vi possa sembrare una cosa semplice e banale, vi assicuro che per me non lo è stato affatto, perché mi ha immensamente aiutato a trascorrere quel periodo senza sentire in maniera significativa la mancanza di casa.

Queste ultime righe le dedico a voi. Semplicemente Grazie. Grazie mamma e papà, Grazie Francesca, per essere il mio punto di riferimento, per essere parte della mia anima.

Grazie di tutto.