AIAA SciTech Forum
6-10 January 2025, Orlando, FL
AIAA SCITECH 2025 Forum

10.2514/6.2025-1416

Pose Estimation of Non-cooperative Spacecraft from LiDAR: Comparison of Methods based on Point Correspondences

Clemente Tecchia[1] Margherita Piccinin[2]
Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Weßling, 82234, Germany

Alessia Nocerino[3] Roberto Opromolla[4]
University of Naples "Federico II", Naples, 80125, Italy

Ulrich Hillenbrand[5]
Institute of Robotics and Mechatronics, German Aerospace Center (DLR), Weßling, 82234, Germany

This paper deals with LiDAR-based pose estimation of a known, non-cooperative spacecraft in mission scenarios requiring close-proximity robotics operations, like On-Orbit Servicing and Active Debris Removal. Attention is focused on classical methods, which today are still relevant in the space domain, being the only solution for safety-critical real-time applications that is space-qualifiable with recognized standards. In this work, different classical algorithms for global pose estimation are compared. We study variants of methods, both known and new ones, which support RANSAC search by guiding point correspondences between measured and model point clouds, using point-normal structures as local features (FPFH) or as non-local primitives (PPF). We also compare these methods to Fast Global Registration (FGR) which treats global registration as a robust optimization problem without assigning hard correspondences. For all the approaches, after the global pose initialization, the estimate is refined using the point-to-plane Iterative Closest Point (ICP) algorithm. The pose estimation pipeline also includes additional evaluation steps to resolve pose ambiguities. All methods are tested using a dataset of synthetic partial and sparse point clouds obtained with a LiDAR data simulator. The methods based on point correspondences outperform FGR by a large margin. Excluding FGR, they are all promising candidates for the considered application with some slight differences in speed and accuracy.

I. Introduction

Spacecraft pose estimation is the problem of computing the set of parameters that describe the relative position and attitude of an active satellite with respect to another space object. The execution of this task requires advanced algorithmic and technological solutions to enable accurate relative navigation and robotics operations in the frame of

[1] Intern, Department of Perception and Cognition.
[2] Post-Doctoral Researcher, Department of Perception and Cognition.
[3] Post-Doctoral Researcher, Department of Industrial Engineering.
[4] Assistant Professor, Department of Industrial Engineering.
[5] Group Leader, Department of Perception and Cognition.

space missions like On-Orbit Servicing (OOS) [1] and Active Debris Removal (ADR) [2], which foresee autonomous maneuvers of a chaser spacecraft in close-proximity to a designated target. If such a space target is non-cooperative (neither equipped with a dedicated communication link nor with easily recognizable artificial markers), the pose estimation task must rely on processing data acquired by active or passive Electro-Optical (EO) sensors [3]. For the common case that the target has a known geometry, model-based pose estimation is then the best option. Many research efforts are currently dedicated to the design and development of vision-based pose estimation techniques relying on either monocular or stereo cameras [4, 5], justified by the relatively small size, weight, and low power consumption of passive imaging sensors. Nevertheless, active Light Detection and Ranging (LiDAR) systems are being typically used in conjunction with cameras, or as main relative navigation sensors, and demonstrated by the recent success of the Mission Extension Vehicle missions [6]. Indeed, LiDAR systems present advantages in terms of operative range, direct depth observability and robustness against unfavorable illumination conditions. This motivates the focus of this study on the design of robust, accurate, and fast LiDAR-based pose estimation algorithms.

To accurately estimate the spacecraft relative state during close-proximity operations, a LiDAR-based pose estimation system shall be able to execute two main functions: *pose initialization*, in which an initial pose measurement is obtained from processing exclusively a LiDAR point cloud in the absence of a priori information; *pose tracking*, in which the pose parameters are updated starting from an initial guess corresponding to the relative state estimate at a previous time instant. In both cases, the pose estimation task can be handled as a registration problem in which one has to look for the set of rotational and translational parameters providing the best alignment between the measured point cloud (*data points*) and the one associated to the target geometry (*model points*). Thus, pose initialization and tracking methods are typically referred to as global and local registration approaches, respectively. The pose initialization task is more complex and more demanding in terms of computational burden due to the need to search for the best solution within the entire 6 Degree of-Freedom (DOF) pose space; it gets even more challenging when dealing with targets having an almost symmetric shape with only minor symmetry breakers that are not well represented in the LiDAR data, which can cause pose ambiguity issues.

Hence, this paper focuses on global registration methods, and specifically on those that rely on establishing local point-wise correspondences between the data point cloud and a model point cloud.

A. Related Work

Within the classical, not learning-based, approaches, global pose estimation can be based on features that are extracted on a local or a global scale of the given point cloud. Object pose can also be inferred from computing and evaluating multiple alignments through matching elementary data and model point constellations. A brief overview of common methods is provided hereinafter.

Previously developed local feature-based methods include: Geometric Hashing (GH) [7], Polygonal Aspect Hashing [8] and Congruent Tetrahedron Align (CTA) [9], which are based on the use of hash tables built by means of an intense offline pre-processing phase to allow for a quick online search of matches between geometric primitives. A prominent example of local feature descriptor is given by the Point Feature Histogram (PFH) [10, 11] which exploits information contained in pairs of points with their normal vectors within a k-neighborhood to effectively describe the local geometry of a point cloud. This concept has been made suitable for applications in a space scenario by the introduction of the Fast Point Feature Histogram (FPFH) [12], namely a faster version of the PFH descriptor with a very good discriminative power. Local descriptors like the FPFH do not provide directly a pose information (unlike the so-called global descriptors), therefore the feature extraction process shall be followed by alignment techniques such as the Sample Consensus Initial Alignment (SAC-IA) [12] or the Fast Global Registration (FGR) [13] algorithms. The former is an algorithm based on Random Sample Consensus (RANSAC) that iteratively selects at least three points from the model and acquired point clouds based on the similarity between the relative descriptors, computes the transformation matrix and evaluates the quality of the alignment through a specific error metric. FGR instead solves an optimization process by minimizing the distances between model and measured point cloud correspondences.

Regarding global feature-based methods, commonly used approaches include Template Matching (TM) techniques, which calculate the pose through matching between the point cloud acquired online and a database of templates which can either be built online, as in the case of the 3 DOF TM [14] and the 1 DOF TM [15], or offline as in the case of the 2 DOF TM [16]. Another option is to rely on global descriptors such as the Viewpoint Feature Histogram (VFH) [17], the Clustered VFH (CVFH) [18], or the Oriented, Unique and Repeatable CVFH (OUR-CVFH) [19], which are extensions of PFH and FPFH that exploit extrinsic information such as the sensor viewpoint to describe globally the point cloud. An alternative to feature histograms is represented by the Basis Point Set (BPS)

[20], where the point cloud is encoded as a fixed-length feature vector of the minimal distances from each basis point in a fixed set.

An approach using pairs of points with their local normal vectors, commonly referred to as Point Pair Features (PPF), has been used widely outside the space domain [21-23]. The point pairs are sampled non-locally across the surface of the data and the model point clouds. Pose hypotheses are computed through rigid alignment of each data pair with each model pair that has a similar geometric configuration, enforced through indexing into a hash table. The search process is commonly run in a RANSAC style, with selection of the result by maximizing a match score of the model to the data. As an alternative, clustering of pose hypotheses in an adapted parameter space was used [24], where the PPF has been referred to as a surflet pair (the term surflet emphasizing the local linear surface approximation). Likewise, the pose hypotheses can be computed simply from triples of distributed points [24, 25].

Beyond the classical methods, today there is increasing interest toward machine learning-based approaches, especially using deep learning, to directly estimate the satellite pose from a LiDAR point cloud after a training process relying on a realistic LiDAR simulator [26-28]. These methods are mostly adaptations of the ones used for ground applications. Generally, deep learning-based methods can be regarded as state of the art for pose estimation from any visual data. However, their use in the space domain is not yet common practice, mainly due to the lack of space-qualified GPU hardware solutions and of accepted and consolidated software standards for qualification. This hinders especially safety-critical and real-time operations like a robotic capture. Indeed, even in the most recent handbooks for the use of machine learning methods in space software [29], classical methods are still recommended to be used as a redundant software system or to take over safety-critical tasks.

B. Problem Statement and Outline

The advantage of detecting and using local correspondences between data and model for pose estimation is a high level of robustness for model deviations on a large scale, e.g., in point clouds degraded through sensory artefacts. Unlike for representations through global features, some isolated similarities of data to model can be enough to find a good alignment that can reach a high match score. However, it all depends on a good selection of potential point correspondences, as the huge number of possible correspondences would otherwise be prohibitive.

The point matching task is particularly challenging when dealing with sparse or very noisy point clouds; it is under-constrained for space targets characterized by planes or axes of symmetry, or by approximate symmetry, such as rocket bodies or spacecraft launch interfaces (typically approached for docking/berthing purposes), or indeed the satellite model that we consider in this study.

We investigate four different techniques for selecting good point correspondences. For two of them we introduce strong constraints for candidate points in order to increase the rate of correct matches within the reduced point sets. These are (i) a *Label-based* method which collects offline FPFH statistics for local geometric structures on the satellite model, and then finds online the data points with similar FPFH statistics; (ii) a *Persistence-Analysis-based (PA-based)* method, which uses the so-called Persistence Analysis [11] to identify the points with the most distinctive FPFHs in the point clouds. The third technique is rather the opposite in that it does not constrain the candidate points but only enforces minimal geometric similarity: (iii) a *PPF-based* method matching pairs of points and their normal vector with similar configuration from data and model [21-23]. Finally, we investigate (iv) a *FPFH-based* method matching points based on FPFH similarity, using the implementation in the Open3D library (there simply called RANSAC) [30]. The proposed point matching techniques are integrated within a complete pose estimation architecture with a RANSAC-style search followed by the point-to-plane Iterative Closest Point (ICP) algorithm [31] for pose refinement and processing steps to distinguish between ambiguous pose solutions as well as reject unreliable estimates.

We compare these methods for pose estimation using point correspondences to FGR [13], which treats it as a robust global optimization problem without assigning hard point correspondences, where we also use the implementation from Open3D.

Performance assessment is carried out using a dataset of synthetic point clouds obtained from LiDAR simulations of the client satellite of OOS-SIM (On-Orbit Servicing Simulator for Capture) at the German Aerospace Center Robotics and Mechatronics (DLR-RM) laboratory [32]. This dataset is split into a training part and a testing part. The former is used for offline processing, i.e., for setting up components of each method; the latter is used for online processing, i.e., for evaluation. Note that the level of training and computational complexity of the trained methods is much below the scale of modern learning-based approaches, deep learning in particular.

The paper is organized as follows. Section II describes in detail the investigated pose estimation architectures and their point matching methodologies. Section III presents and discusses the achieved results. Finally, Section IV concludes with a summary and outlook to future research directions.

II. Methodologies

In this Section we describe the methods used in this paper for estimating the pose of an uncooperative known space target.

In first place, we present a set of four methods, both known (PPF-based, FPFH-based) and our own compositions (Label-based, PA-based), which rely on point correspondences. These methods use a global pose estimation principle based on the RANSAC algorithm, due to its well-known advantages of robustly handling noisy data and outliers commonly encountered in real-world scenarios. RANSAC operates by iteratively selecting random subsets of the input data, hypothesizing a model based on these subsets, and evaluating the model's agreement with the entire dataset. This process involves distinguishing inliers, which conform to the hypothesized model within a predefined tolerance, from outliers, which do not. By maximizing the number of inliers across multiple iterations, RANSAC identifies the model that best represents the underlying data structure while rejecting outliers effectively [33].

In second place, we briefly introduce in this Section also the FGR [13] method, which treats the registration as a robust global optimization and to which we compare the RANSAC-based methods.

Finally, we describe the post-processing steps which follow and are the same for all the five pose initialization methods, consisting in the pose refinement via ICP and in the rejection of unreliable estimates due to pose ambiguities induced by poor conditioning in the sensory data.

To describe the alignment process, we refer to two point sets, the data points, which are measured by a LiDAR sensor and expressed in the Sensor Reference Frame (SRF), and the model points, expressed in the Target Reference Frame (TRF), which is a body-fixed frame of the target satellite.

It is also worth mentioning that the Label-based, PA-based and PPF-based RANSAC make use of *Hash Tables* (HT) for enhancing the point correspondences search. HTs are data structures built offline for the purpose of storing data in a way that makes their retrieval quick and efficient. Therefore, the general scheme for these three algorithms (shown in Fig. 1) is based on two main steps listed below.

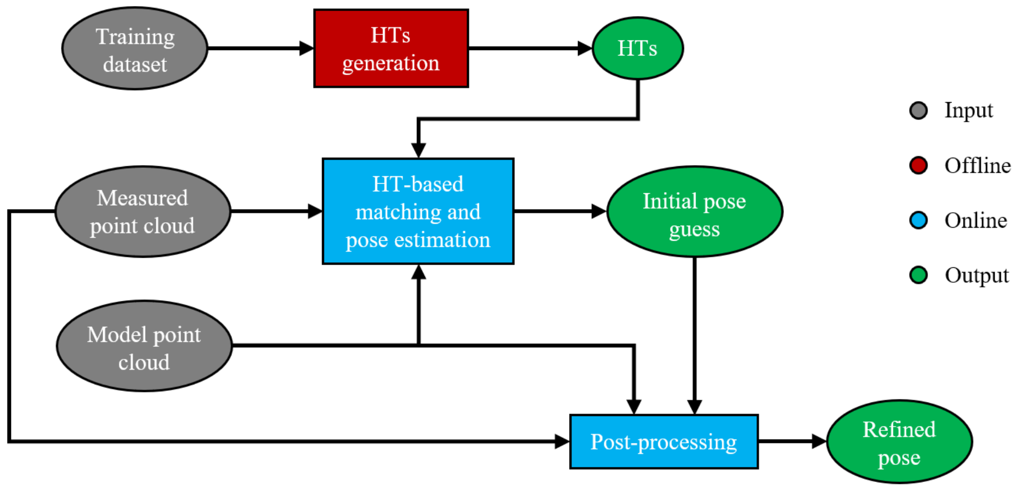

Fig. 1 Pose Estimation Architecture Block Diagram.

- *Offline Hash Tables generation*. A training dataset composed by simulated scans and ground truth poses is used to generate HTs, which store triplets of points or surflet pairs (the term we mostly use for PPF) - depending on the method considered - in buckets, localized by specific hash codes depending on the properties they exhibit. It is worth highlighting that HTs are not built directly from a complete and uniform point cloud of the target, as this choice may lead to practically unfeasible triplets (or pairs). Instead, we make use of the training dataset scans to include in the HT only the point triplets or respectively surflet pairs coherent with the partial views of a LiDAR sensor.
- *Online Hash-Table-based matching and pose estimation*. The measured point cloud is processed and the HTs are searched for correspondences, in an iterative process, until a good match is found. The pose first guess is computed by aligning the corresponding points and relying on a pre-computed KDTree of the model point cloud for the alignment quality evaluation.

A. Global pose estimation methods

In this subsection, the five pose initialization strategies analyzed are described.

1. Label-based RANSAC

The Label-based RANSAC method relies on the decomposition of the geometry under study into labeled geometric primitives (e.g., toroid, plane, edge, cylinder or sphere). The key idea of this approach is to compare the FPFH-based signatures, computed from the training dataset for each defined geometrical class, with the FPFHs of the measured point cloud, computed online. Indeed, this restrains the correspondence determination problem.

Fig. 2 shows a block diagram detailing the offline pre-processing phase of the Label-based RANSAC method.

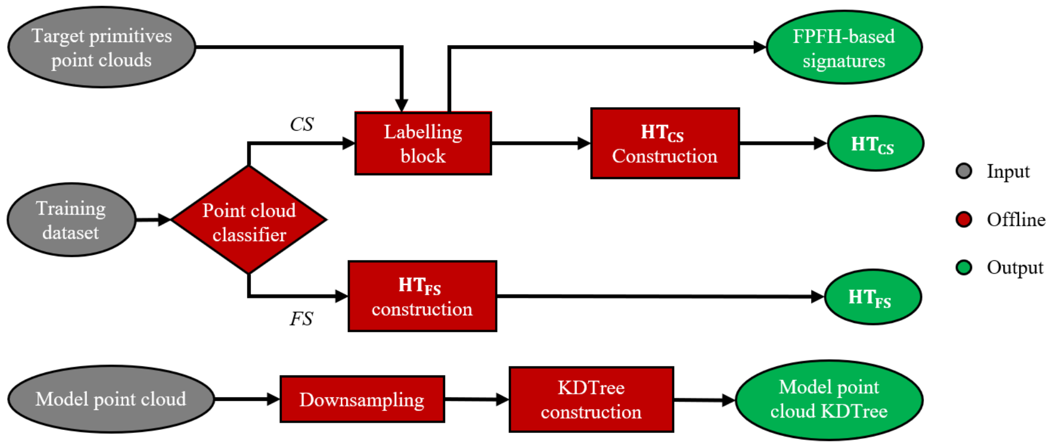

Fig. 2 Label-based RANSAC Offline Block Diagram.

Satellite geometries often present flat surfaces. When a point cloud is completely planar, there is no use in deriving the FPFH signature of its points and the planarity can be quickly checked based on normal unit vectors' variance, even before the FPFH computation. Therefore, for the construction of the HTs, a classification of the point clouds of the training dataset is first carried out by the *Point cloud classifier* block in Fig. 2, 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 show local normal variations; hence they can be treated in a separate HT. 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 over the entire point cloud of the set of normal unit vectors computed using the implementation in [30]. 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.

All the resulting CS scans are sent in input to a *Labelling block* which classifies the points into geometrical classes using the knowledge of the ground truth pose parameters. Hence, for each class a FPFH-based signature is computed as the mean FPFH value. This information is crucial in the online phase for extracting candidate points from the data point cloud.

After labelling, the points of the CS scans are converted in TRF, using the knowledge of the ground truth pose parameters, and adopted to build the corresponding HT ($\mathbf{HT_{CS}}$). The $\mathbf{HT_{CS}}$ is a two-level HT that associates two hash codes to each stored point triplet. These hash codes are defined by the corresponding hash keys as follows.

- The first hash code is a triplet of integer numbers, $\boldsymbol{g}_{hash}$, whose value is set using as key, $\boldsymbol{g}_{key}$, the label of points (e.g., $\boldsymbol{g}_{key} = [cylinder, cylinder, edge] \rightarrow \boldsymbol{g}_{hash} = [1, 1, 0]$).
- 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 Eq. (1), 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).

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

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 an 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 $\boldsymbol{d}_{key}$, 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.

The $\mathbf{HT_{CS}}$ is then filled performing the following operations n_{cycles} times. At each cycle, a triplet of points is randomly extracted and $\boldsymbol{d}_{key}$ is computed: if the $\boldsymbol{d}_{key}$ components are all larger than a threshold (d_{thre}), 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 would result in larger pose errors, making the hashing inefficient. For all the retained triplets, the tuple of points is stored in the HT 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 being saved in the HT considering a different order.

Regarding the FS scans, the corresponding HT ($\mathbf{HT_{FS}}$) 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. At this point, the same approach described for the generation of $\mathbf{HT_{CS}}$ is adopted. The only difference is that a triplet is only identified by the second hash code $\boldsymbol{d}_{hash}$ defined according to Eq. (1).

Finally, another offline operation is the construction of a KDTree structure of the model point cloud downsampled with voxel size v_{size}. Downsampling indeed allows a reduction of the online computational time during the pose refinement. Of course, for this operation, also the measured point clouds need to be downsampled online, with the same voxel size.

Fig. 3 shows a block diagram detailing the online phase of the Label-based RANSAC method.

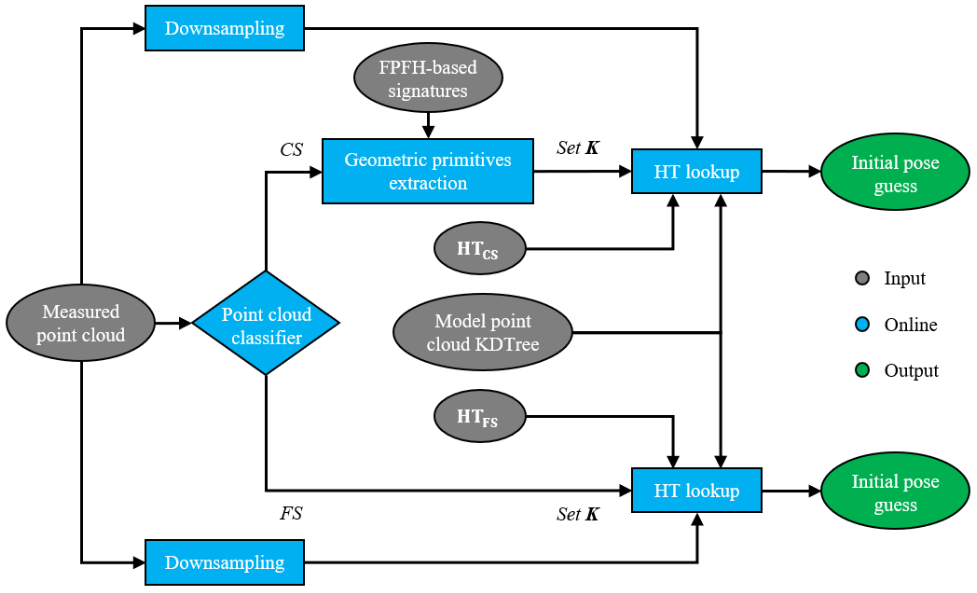

Fig. 3 Label-based RANSAC Online Block Diagram.

Once the measured point cloud is acquired, it is classified as a CS or FS scan. While all the points belonging to a FS scan are part of the set of candidates ($\boldsymbol{K}$) used for the $\mathbf{HT_{FS}}$ lookup, an additional processing step is required to obtain $\boldsymbol{K}$ from a CS scan. The points FPFHs are computed and compared with the FPFH-based signature computed offline. This comparison requires two steps.

- *k-Nearest Neighbour (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}).
- *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.

The best values of the above-introduced tuning parameters for the candidates' selection process (i.e., k_{FPFH} and d_{FPFH}) have been selected through an analysis based on Precision-Recall curves, not presented here for brevity. Fig. 4 shows a block diagram describing the HT lookup phase of the Label-based RANSAC method.

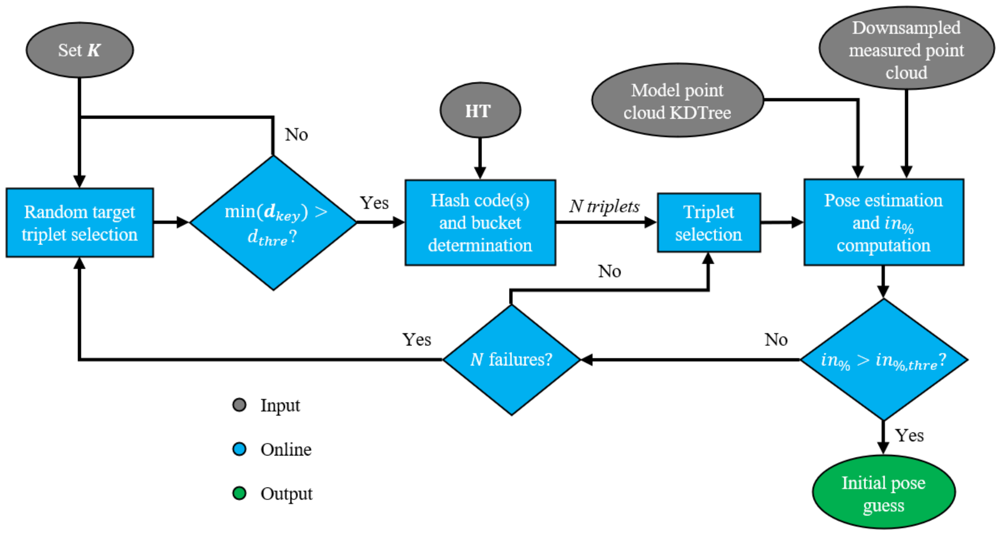
Set $\boldsymbol{K}$
HT
Model point cloud KDTree
Downsampled measured point cloud
Random target triplet selection
$\min(\boldsymbol{d}_{key}) > d_{thre}$?
No
Yes
Hash code(s) and bucket determination
N triplets
Triplet selection
Pose estimation and $in_{\%}$ computation
No
Yes
N failures?
No
$in_{\%} > in_{\%,thre}$?
Yes
Initial pose guess
Input
Online
Output

Fig. 4 HT lookup. This diagram is representative of both the CS and FS cases.

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. At each iteration a random triplet of points is extracted and, after a check on their minimum inter-point distance being larger than d_{thre}, the hash codes $\boldsymbol{g}_{hash}$ and $\boldsymbol{d}_{hash}$ (only $\boldsymbol{d}_{hash}$ in the FS case) 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 match 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 Singular Value Decomposition (SVD) method formally described in Ref. [34]. The quality of this pose guess is then computed by evaluating the number of inliers downstream of the alignment. The inliers percentage is computed leveraging the KDTree of the model point cloud to efficiently find the NNs between model and aligned data point cloud. Specifically, a NN association between data and model points is considered correct if their Euclidean distance is lower than a certain threshold $d_{in,thre}$. In this way, the percentage of inliers $in_{\%}$ is computed as shown in Eq. (2):

$$in_{\%} = \frac{\#inliers}{\#downsampled\ data\ points} * 100 \tag{2}$$

The iterative procedure stops as soon as a threshold representing the minimum desired percentage of inliers ($in_{\%,thre}$) is reached.

2. PA-based RANSAC

The PA-based RANSAC method, uses the Persistence Analysis technique to identify the points with the most distinctive FPFHs in the point clouds. Referring to [12], Persistence Analysis is performed in two main steps.

- *Search for unique points*. After computing the FPFH for each point in the cloud, the mean of all FPFHs is computed (μ – histogram). Then, the distance between the FPFH of each point and the μ – histogram is calculated, using the Kullback-Leibler (KL) Divergence as distance metric [10, 11]. The distribution can be approximated by a Gaussian one, 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 and β is a tuning parameter that controls the width of the interval, are called unique.
- *Search for persistent points*. The previous step is repeated considering spheres of different radius for the FPFH computation. The points which are always unique as the radius varies are called persistent.

The offline phase of the PA-based RANSAC method is very similar to that of the Label-based RANSAC method: it differs for the fact that, by not performing the subdivision into geometric primitives, the only inputs are the training dataset and the model point cloud, while the outputs are $\mathbf{HT_{CS}}$, $\mathbf{HT_{FS}}$ and the KDTree of the model point cloud. Of course, the $\mathbf{HT_{CS}}$ construction procedure is slightly different, being based on PA: the CS scans of the training dataset, expressed in TRF, are first extracted. Then, PA is applied to the resulting dataset to find the set of persistent points to be used for constructing a single-level HT. 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 $\boldsymbol{d}_{hash}$, computed with Eq. (1).

The online phase of the PA-based RANSAC method is also very similar to that already seen for the Label-based RANSAC method: if the measured point cloud is classified as CS scan, PA is applied to extract the set of candidates $\boldsymbol{K}$ to be used for the HT lookup, finally leading to the pose guess using the same procedure adopted for the Label-based RANSAC approach.

3. PPF-based RANSAC

The PPF-based RANSAC method only exploits the information contained in pairs of points with their local normal vectors to identify correspondences [21-23]. The online phase of this algorithm is the simplest among the proposed HT-based algorithms: after classifying the training scans as CS or FS, they are used to lookup $\mathbf{HT_{CS}}$ and $\mathbf{HT_{FS}}$ without any kind of feature extraction. The inputs and outputs are exactly the same as the PA-based RANSAC method.

An additional reference system, called Local Reference Frame (LRF) is used to obtain the hash keys to be associated with the surflet pairs. The unit vectors corresponding to the LRF axes are defined as in Eq. (3):

$$\hat{\boldsymbol{u}} = \frac{\boldsymbol{n}_1}{\|\boldsymbol{n}_1\|}, \qquad \hat{\boldsymbol{v}} = \frac{\boldsymbol{n}_1 \times \boldsymbol{n}_2}{\|\boldsymbol{n}_1 \times \boldsymbol{n}_2\|}, \qquad \hat{\boldsymbol{w}} = \hat{\boldsymbol{u}} \times \hat{\boldsymbol{v}} \tag{3}$$

where $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are respectively the normals in the first and second point of the surflet, $\boldsymbol{p}_1$and $\boldsymbol{p}_2$. The LRF is assumed to be centered in $\boldsymbol{p}_1$. A simplified representation of the LRF is shown in Fig. 5.

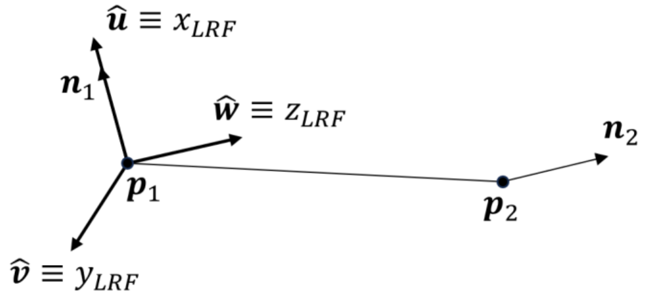
$\hat{\boldsymbol{u}} \equiv x_{LRF}$
$\boldsymbol{n}_1$
$\hat{\boldsymbol{w}} \equiv z_{LRF}$
$\boldsymbol{n}_2$
$\boldsymbol{p}_1$
$\boldsymbol{p}_2$
$\hat{\boldsymbol{v}} \equiv y_{LRF}$

Fig. 5 Graphical representation of a surflet and the defined LRF.

The $\mathbf{HT_{CS}}$ built for the PPF-based RANSAC method has a single level that associates a hash code to each of the stored surflets. So, it relies on surflets associated with a 4D key containing three distance information and one angular information, as detailed below.

- The three distance informations are the three coordinates of the relative position vector between $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, expressed in LRF ($\boldsymbol{d}_{LRF}$).
- 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 Eq. (4) are used:

$$\boldsymbol{R}_{SRF\rightarrow LRF} = [\hat{\boldsymbol{u}} \quad \hat{\boldsymbol{v}} \quad \hat{\boldsymbol{w}}]^T \quad \rightarrow \quad \boldsymbol{d}_{LRF} = \boldsymbol{R}_{SRF\rightarrow LRF}\, \boldsymbol{d}_{SRF} \tag{4}$$

Once the key $\boldsymbol{s}_{key} = [\boldsymbol{d}_{LRF}; \alpha]$ is defined, the hash code is generated using the hash function reported in Eq. (5) and Eq. (6):

$$\boldsymbol{s}_{hash}(i) = floor\left(\frac{m*\boldsymbol{d}_{LRF}(i)}{D}\right), \qquad i = 1, 2, 3 \tag{5}$$

$$\boldsymbol{s}_{hash}(i) = floor\left(\frac{m_a*\alpha}{A}\right), \qquad i = 4 \tag{6}$$

where m is the number of buckets for the coordinates, D is the maximum distance between point pairs, 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 $\mathbf{HT_{CS}}$, 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 between the surflet normals, which must satisfy the condition $\alpha_{min} < \alpha < \alpha_{max}$. The main reason behind this condition is that when α is 0° or 180°, the unit vector $\hat{v}$, which is computed as the cross product between the two normal, would be zero. Moreover, this generally represents an unfavorable 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. 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: $d > d_{thre}$. If this condition is also satisfied, $\boldsymbol{d}_{LRF}$ is computed. Once the key $\boldsymbol{s}_{key}$ is obtained, the hash code $\boldsymbol{s}_{hash}$ is generated through Eq. (5) and Eq. (6) 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. Given the above-mentioned problems in handling surflets with parallel normal unit vectors, the FS scan case is managed using point triplets rather than surflet pairs, as done for the Label-based RANSAC and PA-based RANSAC methods.

The online phase of the PPF-based RANSAC method is very similar to that of the Label and PA-based RANSAC methods, with the exception that, as already anticipated, there are no feature extraction steps: therefore, if the point cloud is classified as a CS scan, the entire point cloud constitutes the set $\boldsymbol{K}$ that enters the HT lookup phase.

During the HT lookup, a surflet is randomly taken from the measured point cloud and, if the conditions set offline are satisfied, $\boldsymbol{s}_{key}$ and its hash code $\boldsymbol{s}_{hash}$ are generated, again using the hash function defined in Eq. (5) and Eq. (6). The search in the bucket corresponding to the obtained hash code generally provides N surflets, each of which is tested for pose estimation, whose goodness is evaluated by estimating $in_{\%}$. If $in_{\%} > in_{\%,thre}$, the initial pose guess has been obtained, otherwise another of the N surflets is selected; if none of the N surflets provides a good pose, the cycle restarts with the random selection of another surflet from the measured point cloud. The HT lookup phase in case of FS scans is the same as in the previous subsections.

The alignment algorithm used for the PPF-based RANSAC method is different to the one adopted by Label-based and PA-based RANSAC methods. 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.

4. *FPFH-based RANSAC*

We compare the three HT-based methods described above with another RANSAC-based algorithm, which is already fully implemented in Open3D and that we name FPFH-based RANSAC [30]. The pre-processing procedure consists in downsampling of the model and the measured point clouds and in estimating the normal vectors and FPFHs for each point of both downsampled point clouds. Then, the FPFH-based RANSAC iteratively extracts n points from the model point cloud, subsequently identifying the set of corresponding points in the measured point cloud by searching for NNs in the FPFH space. For a quick search, Open3D provides algorithms to filter the good matches, based on the distances between correspondences in the two point clouds, on the similarity between the lengths of segments built starting from pairs of corresponding points, and on the orientation of the relative normal unit vectors.

5. *FGR*

Concerning the FGR technique, we again rely on the existing implementation of Open3D [30]. FGR adopts a process that does not rely on the iterative calculation of correspondences; therefore, this algorithm does not follow a RANSAC-based approach. First, point clouds are downsampled, and FPFHs are computed. Then, the key idea is to first establish a set of correspondences $\boldsymbol{K}$ between the data and model point clouds. Specifically, three main steps are performed that aim at determining $\boldsymbol{K}$ based on the similarity between FPFH, through NN 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 $\boldsymbol{K}$ is obtained, an optimization process of an objective follows based on these correspondences (see Ref. [13]).

B. Post-processing

This subsection describes the steps that follow the pose initialization: once the pose has been obtained through one of the global pose estimation algorithms, a pose refinement step follows through the Point-to-Plane (P2L) ICP [35] and some ambiguity reduction checks are performed to correct any incorrect poses due to potential symmetries in the geometry of the reference satellite.

Downstream of the ICP algorithm, checks are implemented to increase the robustness of the proposed methodology against the possibility to get incorrect pose solutions. Consider, for instance, a target that has large planar surface regions. The percentage of inliers required to exit the HT lookup phase could then be reached with an incorrect pose, if the measured point cloud is mainly restricted to one of the satellite planes. Indeed, a version of the true satellite pose that is flipped about this plane has an inlier count almost as larger as the correct pose, a small difference being due only to the small share of points that were measured off plane. 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 ($\boldsymbol{T}_{SRF\rightarrow TRF}$) or to a transformation from SRF to a Flipped Target Reference Frame (FTRF). Therefore, two possible pose solutions can be investigated, which are shown in Eq. (7) and Eq. (8):

$$\boldsymbol{T}_{SRF\rightarrow TRF_1} = \boldsymbol{T}_{ICP} \tag{7}$$

$$\boldsymbol{T}_{SRF\rightarrow TRF_2} = \boldsymbol{T}_{FTRF\rightarrow TRF}\boldsymbol{T}_{ICP} \tag{8}$$

where $\boldsymbol{T}_{FTRF\rightarrow TRF}$ is a matrix dependent on the target reference geometry. In subsection III.A the analyzed flip cases and the $\boldsymbol{T}_{FTRF\rightarrow TRF}$ matrices to correctly manage them are explained. 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.

III. Experiments

In this Section, we introduce the reference geometry used to conduct the analyses and present the performance results of the algorithms. For the sake of completeness, the settings of the tuning parameters characterizing the offline and online phases of the proposed and implemented methods are reported in the Appendix.

A. Scenario

To test the algorithms, we consider the OOS-SIM Client Satellite shown in Fig. 6 (a), which has an approximate 6-fold rotational symmetry of the structure, broken only by minor elements. It also has many large planar regions. This geometry adds complexity to the pose estimation problem.

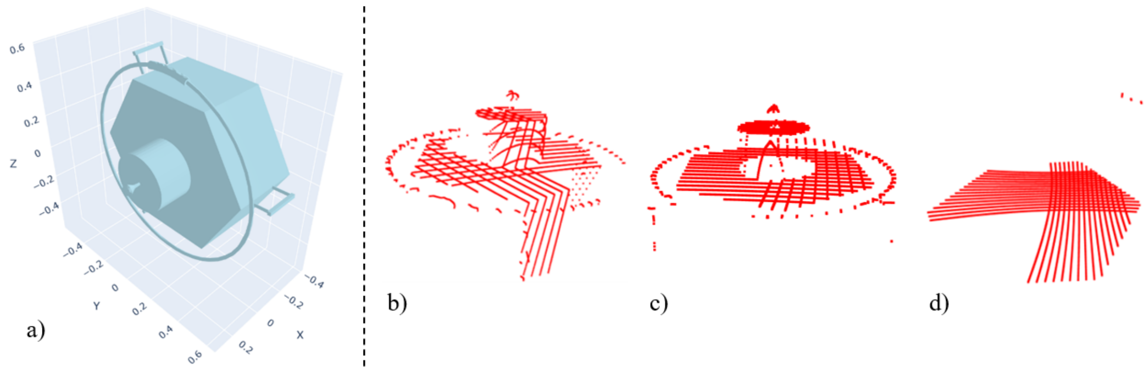

Fig. 6 a) CAD model of the OOS-SIM Client Satellite and TRF representation. b-d) Examples of the point clouds handled for pose estimation. b) CS scan; c) FS scan, upper plane; d) FS scan, lower plane.

For the offline phase and to test the algorithms, a 1000 point cloud dataset is used, 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) [36]. The resulting point clouds appear to be partial, sparse and non-uniform, but are unaffected by data artifacts such as noise and outliers. Some examples are given in Fig. 6 (b-d). Of these 1000 scans, an 80% is used as training dataset while the remaining 20% is used as testing for algorithms' performance assessment.

For the implementation of the Label-based RANSAC method presented in Section II, in particular, a segmentation of the reference geometry into the following geometric primitives has been used: toroid, handles, planes, edges, cylinder and sphere. The decomposition of the geometry applied to the Client Satellite is shown in Fig. 7.

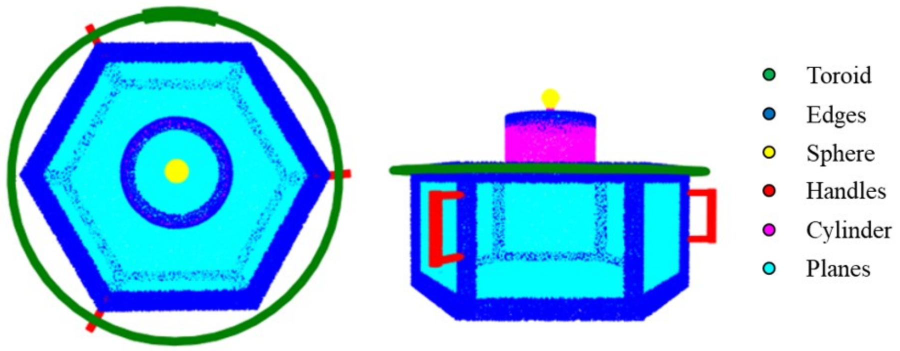

Fig. 7 Target decomposition into geometric primitives.

From preliminary tests we found that cylinder and edge points are the most recognizable through the computation of the FPFH. Hence, the point matching and pose estimation tasks are only executed based on points belonging to these categories.

Finally, regarding the flip checks, we show in Fig. 8 the considered flip cases.

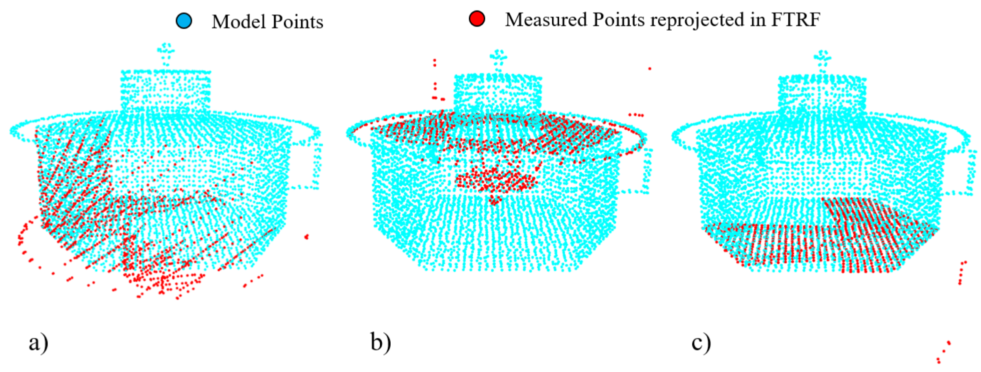

Fig. 8 Flips implemented. a) Flip check performed for all scans. b) FS Flip check: upper plane; c) FS Flip check: lower plane.

Given our 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. Since the FS scan considered (upper or lower) 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.

B. Evaluation metrics

In this subsection, the error metrics for evaluating the performance of the algorithms are defined. Given the quasi-symmetric geometry of the satellite under study, for the method evaluation we consider the symmetric variant of the estimated pose which minimizes the error. Let $\boldsymbol{T}$ be the pose ground truth and $\hat{\boldsymbol{T}}$ 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 [28, 37], given a complete uniform model point cloud $\mathcal{M}$, which are defined in Eq. (9):

$$ADD = \operatorname*{avg}_{\boldsymbol{x} \in \mathcal{M}} \| \boldsymbol{Tx} - \hat{\boldsymbol{T}}\boldsymbol{x} \| , \qquad ADI = \operatorname*{avg}_{\boldsymbol{x}_1 \in \mathcal{M}} \min_{\boldsymbol{x}_2 \in \mathcal{M}} \| \boldsymbol{T}\boldsymbol{x}_1 - \hat{\boldsymbol{T}}\boldsymbol{x}_2 \| \tag{9}$$

Furthermore, for each testing sample, the translational and rotational errors are computed as shown in Eq. (10):

$$t_{err} = \|\boldsymbol{t} - \hat{\boldsymbol{t}}\| , \qquad \phi_{err} = 2 \arccos|\langle \boldsymbol{q}, \hat{\boldsymbol{q}}^{*} \rangle| \tag{10}$$

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 $\hat{\boldsymbol{T}}$. These error metrics are used to compute the Success Rate (SR), which is defined as the percentage of correctly estimated poses, as shown in Eq. (11):

$$SR = \frac{\#\, correct\ estimates}{\#\, test\ samples} * 100 \tag{11}$$

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 the pose refinement. 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.

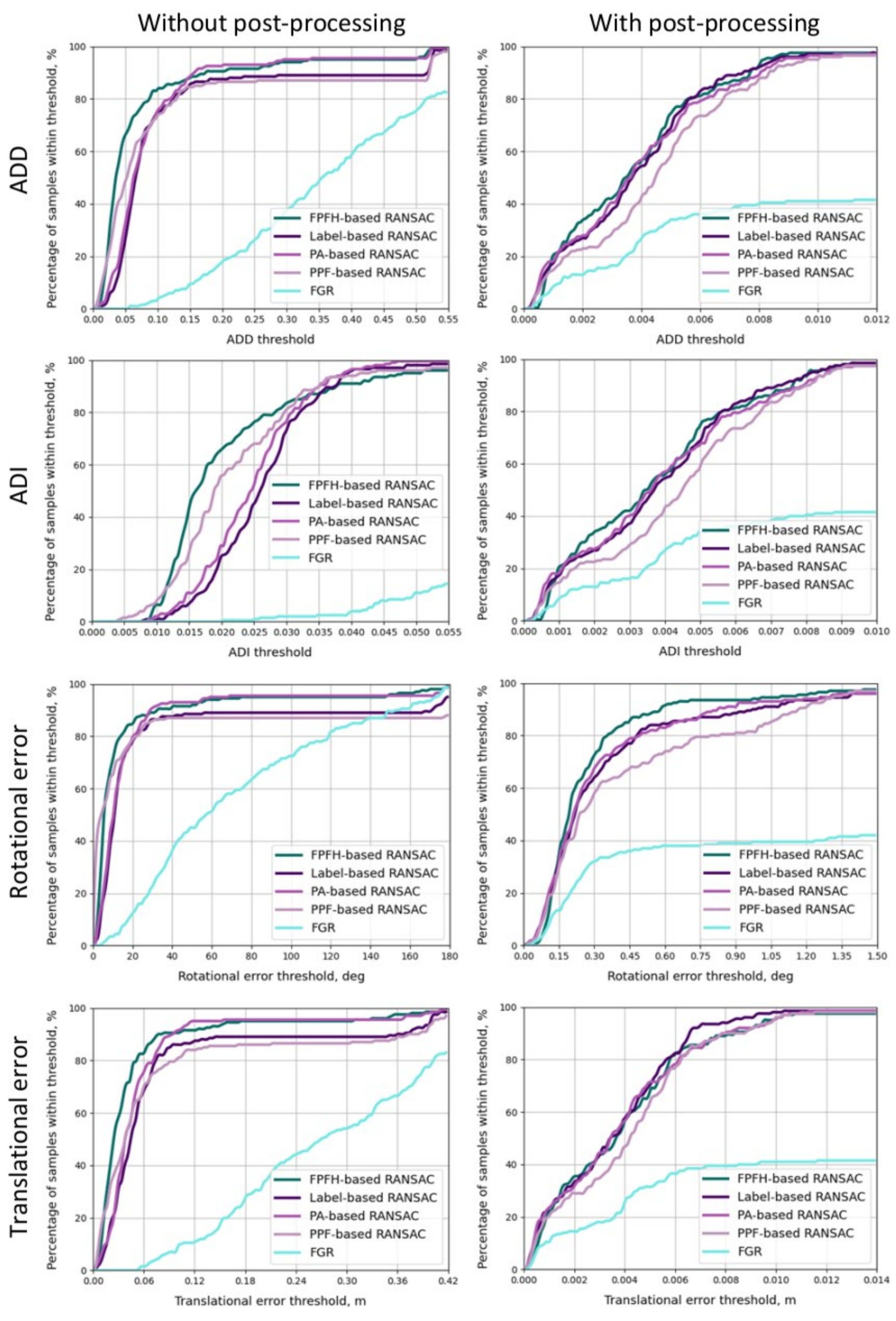

Fig. 9 Comparison results between the algorithms. The metrics compared are ADD, ADI (top 4 plots), rotational error and translational error (bottom 4 plots), without post-processing (left column) and with post-processing (right column)

C. Results

This subsection presents the results of the conducted tests and discusses the achieved performance based on the metrics defined in the previous subsection.

The compared methods are the Label-based RANSAC, PA-based RANSAC, PPF-based RANSAC and the Open3D FPFH-based RANSAC and FGR, appropriately modified by integrating the already existing architectures with the post-processing block (ICP + flip checks) also used for the HT-based algorithms.

Fig. 9 shows the performance comparison between these 5 methods 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. 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 [13], in which the performance of FGR was evaluated on partially overlapping surfaces.

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 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, thus proving that our methods represent promising alternatives, to be further studied with noisy data.

Moreover, 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, further confirming the validity of our alternatives.

Finally, the computational times for both cases without and with post-processing are reported in Table 1.

Table 1 Time Performance.

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

From Table 1, it can be observed that our Label-based and PA-based RANSAC compositions are less performant than FPFH-based RANSAC and PPF-based RANSAC, which instead have comparable computational times. It is important to underline that our HT implementations are not optimized, unlike FPFH-based RANSAC, whose implementation comes from the Open3D library. Therefore, this comparison is to be considered as preliminary.

IV. Conclusion

We have compared four variants of global satellite pose estimation which are based on RANSAC with point correspondences on realistic simulations of LiDAR data. All these variants are promising methods for solving the task in the context of orbital servicing missions. Two of these variants, Label-based RANSAC and PA-based RANSAC, are our own compositions. A slight advantage in accuracy and speed was observed for FPFH-based RANSAC. The speed measurements, however, are just preliminary as not all methods currently have an optimized implementation.

A further comparison to FGR, which treats pose estimation as a global optimization without hard point correspondences, has shown this method to be far inferior and inadequate for the task.

Moreover, we have demonstrated a significant improvement in accuracy through postprocessing the results of global pose estimation with ICP-based refinement and checks for flip errors caused by pose ambiguities present in the LiDAR data under certain conditions.

Future research should consolidate and extend our present results through improvements in data realism or even on real LiDAR data, including different sensor characteristics, and with additional satellite models. In addition, optimized implementations on space hardware are required for confirming the applicability of the studied methods in the context of real orbital missions.

Appendix

Table 2 shows the tuning parameters used for the construction of the HTs.

Table 2 Tuning Parameters Selected for HTs Construction.

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

The algorithms, developed in Python environment, use the open-source Open3D library for 3D data processing operations, including normal and FPFH estimation. Specifically, they are computed using a neighbourhood of points defined by two tuning parameters, which are the search radius r and the maximum number of nearest neighbours max_{nn} [30]. Table 3 shows the tuning parameters used for normal, FPFH estimation and online extraction of the set $\boldsymbol{K}$, respectively, which are the result of a preliminary optimization analysis.

Table 3 HT-based Normal, FPFH Estimation and Feature Extraction Tuning Parameters.

Label-based RANSAC method	*Normals*		*FPFH*		*Cylinder Points Extraction*		*Edge Points Extraction*	
	$r\ [m]$	max_{nn}	$r\ [m]$	max_{nn}	d_{FPFH}	k_{FPFH}	d_{FPFH}	k_{FPFH}
	0.07	40	0.07	100	55	190	50	180
PA-based RANSAC method	*Normals*		*FPFH*		*Persistence Analysis*			
	$r\ [m]$	max_{nn}	$r\ [m]$	max_{nn}	β	$r_i\ [m]$	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*			
	$r\ [m]$	max_{nn}	$r\ [m]$	max_{nn}				
	0.07	40	N/A	N/A				

The performance of the developed HT-based algorithms is compared to FPFH-based RANSAC and FGR techniques as implemented in the Open3D library, but integrating them with the same post-processing procedure performed for the HT-based methods. Table 4 shows the tuning parameters used for normal and FPFH estimation for FPFH-based RANSAC and FGR.

Table 4 FPFH-based RANSAC and FGR Normal and FPFH Estimation Tuning Parameters.

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

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 [30] 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 is selected in such a way as to classify as FS scan and CS scan the exact same samples classified using the threshold of $0.1\ m$ 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, not presented here for brevity.

Table 5 shows, finally, for the HT-based algorithms, the values of the online alignment tuning parameters, selected following an iterative process to identify the optimum ones.

Table 5 Online Phase Tuning Parameters Adopted.

Method	***Alignment Evaluation***				***ICP***		
	v_{size} *[m]*	$d_{in,thre}$ *[m]*	$in_{\%,thre,CS}$ *[%]*	$in_{\%,thre,FS}$ *[%]*	icp_{thre} *[m]*	$n_{it,max}$	icp_{conv} *[m]*
HT-based	0.025	0.05	80	90	0.05	100	10^{-6}

The parameters $in_{\%,thre,CS}$ and $in_{\%,thre,FS}$ represent the percentages of inliers to be satisfied for the pose to be considered good, for CS and FS scans respectively (this threshold is indicated generically in Section II as $in_{\%,thre}$), while icp_{thre} and icp_{conv} are the distance threshold set for the execution of the ICP and a convergence parameter given by the difference between the Root Mean Square Error (RMSE) of two consecutive iterations, respectively.

Regarding FPFH-based RANSAC, instead, the number of points used to compute the initial pose guess is set equal to 3; the input parameters of the *CorrespondenceCheckerBasedOnDistance* and *CorrespondenceCheckerBasedOnEdgeLength* functions [30] 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 $0.05\ m$. The same values of icp_{thre}, $n_{it,max}$ and icp_{conv} are used.

Acknowledgments

This research was funded by the German Aerospace Center (DLR) as part of the projects ION and RICADOS 2.0 about on-orbit servicing. Concerning the authors from the University of Naples "Federico II", this study was carried out within the Space It Up! project funded by the Italian Space Agency, ASI, and the Ministry of University and Research, MUR, under contract n. 2024-5-E.0 - CUP n. I53D24000060005.

References

[1] 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.
doi: https://doi.org/10.1016/j.paerosci.2014.03.002

[2] 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.
doi: https://doi.org/10.1016/j.actaastro.2012.11.009

[3] 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.
doi: https://doi.org/10.1016/j.paerosci.2017.07.001

[4] 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.
doi: https://doi.org/10.1504/IJSPACESE.2014.060600

[5] 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.
doi: https://doi.org/10.1155/2014/461283

[6] 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.
doi: https://doi.org/10.1117/12.2631524

[7] 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.
doi: 10.1109/CCV.1988.589995

[8] 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.
doi: 10.1109/AERO.2008.4526516

[9] 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.
doi: https://doi.org/10.3390/s18041009

[10] 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.
doi: 10.1109/IM.2003.1240284

[11] 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), Baden-Baden, Germany*, 2008, pp. 119-128.
doi: 10.3233/978-1-58603-887-8-119

[12] 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.
doi: 10.1109/ROBOT.2009.5152473

[13] Zhou, Q. Y., Park, J., and Koltun, V., “Fast Global Registration,” *Computer Vision - ECCV 2016*, 2016, pp. 766–782.
doi: https://doi.org/10.1007/978-3-319-46475-6_47

[14] 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.
doi: https://doi.org/10.3390/s150306360

[15] 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.
doi: 10.1109/TAES.2017.2650785

[16] 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.
doi: https://doi.org/10.2514/1.G005042

[17] 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.
doi: 10.1109/IROS.2010.5651280

[18] 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.
doi: 10.1109/ICCVW.2011.6130296

[19] 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.
doi: https://doi.org/10.1007/978-3-642-32717-9_12

[20] Prokudin, S., Lassner, C., and Romero, J., “Efficient Learning on Point Clouds with Basis Point Sets,” *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019, pp. 3072-3081.
doi: 10.1109/ICCVW.2019.00370

[21] 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.
doi: 10.1109/3DV.2015.65

[22] Hinterstoisser, S., Lepetit, V., Rajkumar, N., and Konolige, K., “Going Further with Point Pair Features,” *Computer Vision – ECCV 2016*, 2016, pp. 834–848.
doi: https://doi.org/10.1007/978-3-319-46487-9_51

[23] 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.
doi: 10.1109/ICCAR.2018.8384709

[24] Hillenbrand, U., and Fuchs, A., “An experimental study of four variants of pose clustering from dense range data,” *Computer Vision and Image Understanding*, Vol. 115, No. 10, 2011, pp. 1427-1448.
doi: https://doi.org/10.1016/j.cviu.2011.06.007

[25] Hillenbrand, U., “Consistent parameter clustering: Definition and analysis,” *Pattern Recognition Letters*, Vol. 28, No. 9, 2007, pp. 1112-1122.
doi: https://doi.org/10.1016/j.patrec.2007.01.006

[26] Pensado, E. A., de Santos, L. M. G., Jorge, H. G., and Sanjurjo-Rivo, M., “Deep Learning-Based Target Pose Estimation Using LiDAR Measurements in Active Debris Removal Operations,” *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 59, No. 5, 2023, pp. 5658-5670.
doi: 10.1109/TAES.2023.3262505

[27] Zhang, S., Hu, W., and Guo, W., “6-DoF Pose Estimation of Uncooperative Space Object Using Deep Learning with Point Cloud,” *2022 IEEE Aerospace Conference (AERO)*, 2022, pp. 1-7.
doi: 10.1109/AERO53065.2022.9843444

[28] 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*.

[29] European Cooperation for Space Standardization, ECSS-E-HB-40-02A, “Space engineering – Machine learning qualification handbook”, 2023.

[30] Zhou, Q. Y., Park, J., and Koltun, V., "Open3D: A Modern Library for 3D Data Processing", *ArXiv preprint arXiv:1801.09847*, 2018.
doi: https://doi.org/10.48550/arXiv.1801.09847

[31] Besl, P.J., and McKay, N. D., “A method for registration of 3-D shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, No. 2, 1992, pp. 239-256.
doi: 10.1109/34.121791

[32] 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.
doi: 10.1109/ICRA.2015.7139588

[33] Fischler, M., A., and Bolles, R., C., “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”, *Association for Computing Machinery*, Vol. 24, No. 6, 1981, pp. 381-395.

[34] 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.
doi: 10.1109/TPAMI.1987.4767965

[35] 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.
doi: 10.1109/IM.2001.924423

[36] Shoemake, K., “III.6 - UNIFORM RANDOM ROTATIONS,” *Graphics Gems III (IBM Version)*, 1992, pp. 124-132.
doi: https://doi.org/10.1016/B978-0-08-050755-2.50036-1

[37] Hodaň, T., Matas, J., and Obdržálek, Š., “On Evaluation of 6D Object Pose Estimation,” *Computer Vision – ECCV 2016 Workshops*, 2016, pp. 606-619.
doi: https://doi.org/10.1007/978-3-319-49409-8_52