Journal of Field Robotics, Volume 37, Issue 5, Special Issue on Space Robotics, August 2020, pp. 848-888
DOI:10.1002/rob.21904

Evaluating 3D Local Descriptors and Recursive Filtering schemes for LIDAR based Uncooperative Relative Space Navigation

Odysseas Kechagias-Stamatis*
Signals and Autonomy group
Centre for Electronic Warfare, Information and Cyber
Cranfield Defence and Security, Cranfield University
Shrivenham, SN6 8LA, UK
o.kechagiasstamatis@cranfield.ac.uk

Nabil Aouf
Department of Electrical and Electronic Engineering
City, University of London
Northampton Square
London EC1V 0HB, UK

Vincent Dubanchet
Thales Alenia Space
5 Allée des Gabians
Cannes, FR 06150, FRANCE

Abstract

We propose a Light Detection and Ranging (LIDAR) based relative navigation scheme that is appropriate for uncooperative relative space navigation applications. Our technique combines the encoding power of the 3D local descriptors that are matched exploiting a correspondence grouping scheme, with the robust rigid transformation estimation capability of the proposed adaptive recursive filtering techniques. Trials evaluate several current state-of-the-art 3D local descriptors and recursive filtering techniques on a number of both real and simulated scenarios that involve various space objects including satellites and asteroids. Results demonstrate that the proposed architecture affords a 50% odometry accuracy improvement over current solutions, while also affording a low computational burden. From our trials we conclude that HoD-S combined with the adaptive $\alpha\beta$ filtering poses the most appealing combination for the majority of the scenarios evaluated, as it combines high quality odometry with a low processing burden.

1. Introduction

Autonomous ego-motion estimation, i.e. odometry, for space platforms is currently of high interest due to the increasing number of spacecrafts deployed. Specifically, great research interest is in the

Published by Wiley. This is the Author Accepted Manuscript issued with: Creative Commons Attribution Non-Commercial License (CC:BY:NC 4.0).
The final published version (version of record) is available online at DOI:10.1002/rob.21904. Please refer to any applicable publisher terms of use.

field of relative space odometry of a *Source* spacecraft platform in relation to a non-cooperative *Target* platform, i.e. with unknown attitude. The aim of space odometry is to estimate the six Degrees of Freedom (DoF) relative motion between the *Source* and the *Target*, with three DoF representing the relative translation and three the relative rotation. Applications utilizing the autonomous relative *Source – Target* motion estimation involve close-proximity operations where the distance of the two space platforms is such that possible communication loss or delay between the ground station and the *Source* may lead to a *Source – Target* platform impact. Typical distances for close-proximity operations range from a few meters up to tens of meters and depend on the sensors employed, the type of operation and the target size (Fehse, 2003). The need for autonomous navigation extends to a wide range of tasks such as controlled orbiting, docking and close range rendezvous that are required for space missions such as active debris removal (Bonnal, Ruault, & Desjean, 2013; Yılmaz, Aouf, Majewski, Sanchez-Gestido, & Ortega, 2017), celestial body exploration, e.g. comets and asteroids (Cheng, 2002) and on-orbit servicing (Flores-Abad, Ma, Pham, & Ulrich, 2014).

In the context of close-proximity space navigation, the sensors typically used are electro-optical with the majority of current solutions relying on passive sensors using visual or Infrared (thermal) sensing devices in a monocular or stereo camera configuration, or alternatively 3D LIDAR for active systems (Opromolla, Fasano, Rufino, & Grassi, 2017). Indeed, current space relative navigation solutions involve 2D visual data in a monocular (Krämer, Hardt, & Kuhnert, 2018; C. Liu & Hu, 2014) or a stereo camera configuration (Li, Lian, Guo, & Wang, 2013; Maimone, Cheng, & Matthies, 2007; Tykkala & Comport, 2011; Yang Cheng, Maimone, & Matthies, 2006), 2D Infrared (IR) data (Yılmaz et al., 2017) and 3D Light Detection and Ranging (LIDAR) data (Galante et al., 2012; Gómez Martínez, Giorgi, & Eissfeller, 2017; Naasz & Moreau, 2012;

Opromolla, Di Fraia, Fasano, Rufino, & Grassi, 2017; Opromolla, Fasano, Rufino, & Grassi, 2014, 2015a; Sell, Rhodes, Woods, Christian, & Evans, 2014; Song, 2017; Volpe, Palmerini, & Sabatini, 2017; Woods & Christian, 2016). For a comprehensive review on spaceborne sensors for spacecraft pose determination the reader is referred to (Opromolla, Fasano, et al., 2017).

Utilizing sensors that exploit passive data, i.e. visual or IR, can be an effective solution that is characterized by low hardware complexity, cost and power consumption due to neglecting a transmitting device. Using visual and IR sensors can be an effective solution, however, IR has several advantages over the visual domain such as operating during day and night under several harsh illumination conditions like eclipse and solar glare (Yılmaz et al., 2017). Despite these advantages, the performance of IR odometry depends on the temperature of the *Target* platform, which is affected by both internal parameters, e.g. heat dissipation of the platform's components, and external parameters, e.g. reflection of sun's radiation. This temperature fluctuation can affect the robustness of the IR based local feature detection and matching technique presented in current literature (Yılmaz et al., 2017). On the contrary, 3D LIDAR based odometry outmatches its 2D counterparts (visual and IR) as it operates during day and night, is independent of the *Target's* thermal properties, is capable of revealing the underlying structure of an object (Guo, Sohel, Bennamoun, Lu, & Wan, 2013a), can provide full *Target* pose estimation (6-DoF), can discriminate the target from the background, is robust in poor visibility conditions and has a greater operating distance compared to the maximum stereo odometry range that poses acceptable accuracy (Opromolla, Fasano, et al., 2017). Despite these advantages, LIDAR sensors and the associated software that exploits the 3D *Target* information acquired, impose higher hardware requirements compared to the visual and IR sensors that in principle produce 2D data. However, due to the advantages of LIDAR sensors against visual / IR sensors and given that the former have

already been placed on space platforms (Kornfeld et al., 2003a), it is clear that LIDAR based navigation is an overall appealing and affordable option. An open case is the processing recourses required for 3D data manipulation. Space platforms use space-graded field programmable gate arrays (FPGA) and recent work (Estébanez Camarena, Feetham, Scannapieco, & Aouf, 2018) demonstrated that it is feasible FPGA boards to perform 2D computer vision based navigation. Hence, there is the potential for FPGA boards to perform complex 3D navigation utilizing LIDAR data. However, the aim of this work is to evaluate the conceptual validity and performance of several 3D local descriptors and recursive filtering schemes rather than suggesting a readily available navigation system.

On that basis and driven by the advantages of LIDAR, current literature suggests quite a few LIDAR based relative navigation solutions (Galante et al., 2012; Gómez Martínez et al., 2017; Naasz & Moreau, 2012; Opromolla, Di Fraia, et al., 2017; Opromolla et al., 2014, 2015a; Sell et al., 2014; Song, 2017; Volpe et al., 2017; Woods & Christian, 2016). However, these present a number of deficiencies:

a. Registration of the two sequential point clouds current methods mostly relies on the Iterative Closest Point (ICP) method (Besl & McKay, 1992) that may settle in a local rather than a global optimum solution.

b. Current algorithms involve an off-line training process that uses a 3D model of the expected *Target* platform (A. B. Dietrich & McMahon, 2018; A. P. Rhodes, Christian, & Evans, 2017). However, these models are not available for unknown *Targets* such as unexplored comets, asteroids or space debris, or for known *Target* space platforms that are corrupted with an unknown level of corruption.

c. Current space navigation solutions that rely on computer vision concepts involve regional 3D feature description and not local description (A. Rhodes, Kim, Christian, & Evans, 2016; A. P. Rhodes et al., 2017), neglecting the state-of-the-art performance afforded by the latter type. It should be noted that (A. Rhodes et al., 2016) made an attempt using the local feature descriptor Spin Images (Andrew Edie Johnson & Hebert, 1998), however the research concluded that this descriptor is not optimal for space navigation. To the best our knowledge none other local descriptor has been used to date.

d. The majority of current proposals is evaluated on fully simulated scenarios (Gómez Martínez et al., 2017; Opromolla et al., 2014, 2015a; A. P. Rhodes et al., 2017; Song, 2017; Volpe et al., 2017; Woods & Christian, 2016), while only a few algorithms are tested on real but rather simplistic scenarios (Galante et al., 2012; Sell et al., 2014). Recently, orbit determination around small bodies using LIDAR has been proposed (A. B. Dietrich & McMahon, 2018), however this architecture uses a shape model of the *Target* restricting this technique to known *Targets* (deficiency (b)).

e. Space related odometry literature does not involve any type of feature matching refinement schemes, such as a correspondence grouping technique (Yang, Xian, Xiao, & Cao, 2018).

In contrast to space robotics odometry, terrestrial odometry (ground and aerial) is more mature spreading over several data domains. A few examples are 2D visual in a monocular or stereo camera configuration (Cvišić, Ćesić, Marković, & Petrović, 2018; Nemra & Aouf, 2009), 2D IR (Mouats, Aouf, Chermak, & Richardson, 2015), fusion of visual with IR data (Mouats, Aouf, Sappa, Aguilera, & Toledo, 2015), fusion of 3D LIDAR with visual data (Graeter, Wilczynski, &

Lauer, 2018; Zhang & Singh, 2015b), 3D LIDAR fused with inertial data (Neuhaus, Koß, Kohnen, & Paulus, 2019; Zhang & Singh, 2015a), odometry that solely relies on LIDAR data (Deschaud, 2018; Ji et al., 2018) and methods that are based on RGB-D data (Jaimez & Gonzalez-Jimenez, 2015; Kim & Kim, 2016; Zhou, Li, & Kneip, 2019). An extensive review on the odometry methods for terrestrial applications is presented in (Aqel, Marhaban, Saripan, & Ismail, 2016).

Simply extending current algorithms designed for terrestrial LIDAR based robotics odometry into the field of space robotics odometry is not an optimum solution for the following reasons:

a. The space environment lacks surrounding objects forcing space odometry to rely on a limited number of vertices that belong to a single object. Typical terrestrial point cloud scenes comprise of a few hundreds of thousands of vertices and involve many objects, while a space odometry scene has at least one order of magnitude fewer vertices and encloses a single object.

b. Space objects are typically less complex than terrestrial scenes further increasing the already challenging space odometry estimation. The advantage of complex scenes is affording sufficiently unique and non-degenerated geometries enforcing odometry to converge to acceptable solutions.

c. Methods that solely use LIDAR data may also not be appealing as these exploit the hardware properties of terrestrial LIDAR sensors that differ from the spaceborne ones. For example, the method in (Deschaud, 2018) uses the continuous spinning effect of the LIDAR sensor, which for spaceborne LIDAR sensors this is not applicable as either scanning or flash LIDAR sensors are used.

d. Terrestrial odometry that exploits LIDAR data may involve fusing information from additional sensors such as visual data or inertial data (Graeter et al., 2018; Neuhaus et al., 2019; Zhang & Singh, 2015b, 2015a). However, a spaceborne sensor suite selection is constrained by the size, weight and power consumption of each sensor and thus involving two different sensor types, e.g. LIDAR and visual camera, may not be payload efficient.

Additional challenges preventing directly utilizing algorithms for terrestrial applications on space navigation scenarios are the differences between the earth-based point cloud data and the space-based point cloud data, which are linked to the device that they were generated from. Considering point clouds originating from LIDAR, as is the case in this work, the LIDAR earth-based point cloud would be in general denser as its mapping covering area is much focused and smaller than the general LIDAR space-based point clouds. However, the later from a farther range would be mapping a large area on orbit to detect the satellite and the satellite sub-part of interest. In terms of noise, the space-based data would be less affected than their earth counterpart since the space LIDAR would have less noisy returns. This is because the background around the target satellite is empty, while on earth applications, i.e. autonomous cars, the background contains several objects. In terms of range accuracy, the space-based point cloud data would be more accurate than the earth-based point cloud data.

Spurred by the advantages of 3D LIDAR odometry and the deficiencies of the current solutions for space navigation, we suggest a LIDAR based relative navigation architecture appropriate for space applications. The contributions and innovations of this work can be summarized to:

a. We suggest an odometry architecture that combines the concepts of 3D local feature description, feature matching refinement and recursive filtering for the rigid body transformation estimation.

b. The recursive filtering process is adaptive and linked to the quality of the matched features.

c. Opposed to current space-oriented odometry solutions, this architecture does not require any prior knowledge of the *Target* platform extending significantly the usability of the proposed solution.

d. We evaluate the suggested technique on seven scenarios that involve one real and four simulated space objects of various complexity.

e. On each scenario, we evaluate six current 3D local feature descriptors and four recursive filtering schemes that are tested in all possible combinations.

f. We also evaluate the performance of the proposed architecture against five variations of ICP that is widely used for space robotics odometry, and also against two techniques used for registration application in the computer vision domain. To the best of our knowledge such an extensive evaluation is unique in the space-related odometry literature.

It is expected the proposed method to achieve lower odometry errors compared to current ICP based methods, because the feature matching and the geometric correspondence grouping schemes shall provide to the recursive filter only well-established correspondences. These correspondences combined with the adaptive nature infused in the recursive filter affords further improvement and optimization of the odometry performance.

The rest of the paper is organized as follows; Section 2 presents the suggested odometry architecture, while Section 3 compares the accuracy of the proposed pipeline involving several 3D descriptors and recursive filtering combinations against current registration methods on a variety of scenarios. Section 4 analyses the contribution of each module within our architecture, compares our method against a mainstream computer vision-based registration method and also presents the interaction between the number of iterations and the odometry performance for each competitor ICP variant. Finally, Section 5 concludes this paper.

2. Proposed odometry architecture

2.1 LIDAR odometry

The suggested LIDAR relative navigation architecture considers a two-platform setup, i.e. a *Source* platform that incorporates a 3D LIDAR sensor and an uncooperative *Target* platform. The aim of the proposed technique is to estimate the relative position of the *Source* platform to the *Target* platform.

Given two sequential point clouds $\boldsymbol{P}_k = \{p_k^1, ..., p_k^a\}$ and $\boldsymbol{P}_{k+1} = \{p_{k+1}^1, ..., p_{k+1}^b\}$ of the *Target* that are captured from the LIDAR device placed on the *Source* platform at instance *k* and *k+1*, with each vertex being in the form $p_k = (x_k, y_k, z_k)$ and $p_{k+1} = (x_{k+1}, y_{k+1}, z_{k+1})$, aim of the odometry process is to calculate a rigid body transformation:

$$R^* = \begin{bmatrix} R & \vert & T \\ 0 & \vert & 1 \end{bmatrix} \tag{1}$$

with *R* the rotation and *T* the translation component that remap point cloud $\boldsymbol{P}_k$ to $\boldsymbol{P}_{k+1}$:

$$p_{k+1} = Rp_k + T \tag{2}$$

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ z_{k+1} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{3}$$

Then at instance u, the position of the *Source* moving platform relatively to the uncooperative *Target* platform is given by:

$$R_u^* = \prod_{\mu=1}^{u} R_\mu^* \tag{4}$$

The rotation matrix in eq. (2) may originate from Euler or Quaternion angle encodings. In this work we adopt (A. B. Dietrich & McMahon, 2018; A. Dietrich & McMahon, 2017) and use Euler angles.

Typically, R^* is estimated by employing a global registration technique such as the ICP, the ICP variants or the Super 4-point Congruent Sets (S4PCS) (Mellado, Aiger, & Mitra, 2014), involving $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$.

2.2 Recursive Filtered 3D Local Features based LIDAR odometry

Driven by the need of a high performing uncooperative relative navigation architecture for relative space navigation, we suggest an architecture that estimates R^* by combining the concepts of 3D local feature description, geometrical correspondence refinement and adaptive recursive filtering. For the former, we evaluate several current state-of-the-art 3D local feature descriptors (Section 2.2.1) while for the latter we evaluate both linear and non-linear filters (Section 2.2.3). The suggested architecture is presented in Figure 1. It is worth noting that current odometry literature for terrestrial applications suggests global pose-graph optimization, i.e. finding feature matches

between non-adjacent frames or exploiting the loop closure technique. Despite that, in this work we do not adopt these concepts and push the limits by investigating the performance attained by exploiting only sequential point clouds. Preliminary work is presented in (Kechagias-stamatis & Aouf, 2019) without though involving in that work any geometric consistency checks and limiting the evaluation on a single 3D descriptor and to real data scenarios.

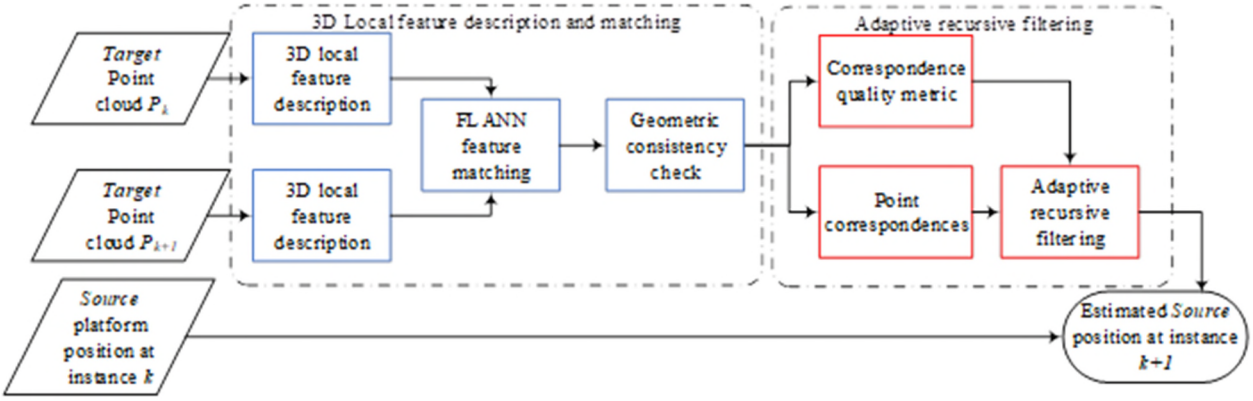


Figure 1: Suggested relative navigation architecture

2.2.1 3D local feature description

Local feature description techniques describe local patches around a point of interest, i.e. keypoint, by encoding the geometric properties and the underlying structure of the local patch (O. Kechagias-Stamatis et al., 2018). Their major advantages are affording robust feature description to partially visible objects (Odysseas Kechagias-Stamatis, Aouf, & Nam, 2017) and being less affected by illumination variation and pose changes (Lei Yunqi, Lai Haibin, & Jiang Xutuan, 2010; Mian, Bennamoun, & Owens, 2006).

Current literature suggests quite a few handcrafted 3D local descriptors with representative ones being the Signatures of Histograms of Orientations (SHOT) (Salti, Tombari, & Di Stefano, 2014), Spin Images (Andrew E. Johnson & Hebert, 1999), Rotational Projection Statistics (RoPS) (Guo

et al., 2013a), Tri-Spin Images (TriSI) (Guo, Sohel, Bennamoun, Lu, & Wan, 2013b), Fast Point Feature Histograms (FPFH) (Rusu, Blodow, & Beetz, 2009), 3D Shape Context (3DSC) (Frome, Huber, Kolluri, Bülow, & Malik, 2004), Unique Shape Context (USC) (Tombari, Salti, & Di Stefano, 2010), Histogram of Distances (HoD) (Odysseas Kechagias-Stamatis & Aouf, 2016), HoD-Short (HoD-S) (Odysseas Kechagias-Stamatis & Aouf, 2017, 2018), Local Feature Statistics Histogram (LFSH) (Yang, Cao, & Zhang, 2016), Multi-attribute Statistics Histograms (MaSH) (Yang, Zhang, & Cao, 2017), Statistic of Deviation Angles on Subdivided Space (SDASS) (Zhao, Le, & Xi, 2019) and Binary Rotational Projection Histogram (BRoPH) (Zou et al., 2018). It is worth noting that lately except from handcrafted local feature descriptors, learned 3D feature descriptors have also been used. Examples are the Point Pair Feature Network (Deng, Birdal, & Ilic, 2018) and the Compact Geometric Features descriptor (Khoury, Zhou, & Koltun, 2017). A downside of both methods is the requirement of offline training prohibiting their usage for odometry that involves an uncooperative and unknown *Target* platform.

In this work we use within our odometry architecture several commonly used 3D feature descriptors and specifically SHOT, TriSI, HoD-S, HoD, FPFH and RoPS. A common principle of all these 3D descriptors is encoding a spherical volume $\boldsymbol{V}$ of radius r that is centered on a keypoint $p(x,y,z)$. For completeness, a short description of the 3D descriptors evaluated is presented.

2.2.1.1 HoD and HoD-S

HoD and HoD-S calculate the probability mass density of the normalized point-pair L_2-norm distance distributions within $\boldsymbol{V}$. The difference between these two 3D descriptors is that HoD encodes the distance distributions in a coarse and a fine manner, while HoD-S only in a coarse manner. Main advantages of both these descriptors over the current ones are neglecting the

requirement of a Local Reference Frame or Axis (LRF/A), being robust to nuisance factors such as noise and being fast to compute. Indeed, HoD and HoD-S are robust to Gaussian noise, occlusion, clutter and point cloud resolution variation (Odysseas Kechagias-Stamatis & Aouf, 2016, 2017, 2018; Odysseas Kechagias-Stamatis et al., 2017).

2.2.1.2 SHOT

SHOT divides the support volume $\boldsymbol{V}$ into a pre-defined number of sub-volumes along the azimuth, the elevation and the radius. For each sub-volume, a 1D histogram is calculated based on the normal variation between the keypoint *p(x,y,z)* (including its surrounding vertices) and the vertices that lie in each sub-volume. SHOT is robust to Gaussian noise and Shot noise, and sensitive to occlusion and clutter (Guo et al., 2016).

2.2.1.3 FPFH

FPFH establishes on $\boldsymbol{V}$ a *Darboux* LRF. Then for each point belonging to $\boldsymbol{V}$, FPFH encodes the angular relationship between the keypoint *p(x,y,z)* and its neighbors as provided by the LRF. Finally, this angular relationship is transformed into a histogram. FPFH is robust to resolution variation and sensitive to Gaussian noise, Shot noise, occlusion and clutter (Guo et al., 2016).

2.2.1.4 RoPS

RoPS establishes on $\boldsymbol{V}$ a LRF. Then $\boldsymbol{V}$ is rotated around each axis of the LRF and is projected on each of the coordinate planes. Finally, each projection undergoes a statistical analysis based on low order moments and entropy, and all the results are concatenated into a histogram. RoPS is robust to Gaussian noise and sensitive to Shot noise, occlusion and clutter (Guo et al., 2016).

2.2.1.5 TriSI

TriSI is an extension of the 3D descriptor Spin Images (SI). For the latter, given a support volume $\boldsymbol{V}$ centered at point *p(x,y,z)*, a LRA is aligned with the normal vector of the plane fitted to the vertices within $\boldsymbol{V}$. Then a 2D array accumulator of user defined dimensions is placed on the LRA and the SI descriptor is generated by accumulating the neighboring points into each bin of the 2D array as the array rotates around the LRA. TriSI uses the same technique as the SI but substitutes the LRA with an LRF and calculates a SI descriptor for each axis of the reference frame. Finally, the three SI descriptors are concatenated to from a single descriptor. TriSI is robust to Gaussian noise, Shot noise and resolution variation, while it is sensitive to occlusion and clutter (Guo et al., 2016).

2.2.2 Local feature matching

Let $\boldsymbol{F}_k = \{f_k^1, ..., f_k^{N_i}\}$ and $\boldsymbol{F}_{k+1} = \{f_{k+1}^1, ..., f_{k+1}^{N_j}\}$ be two sets of 3D features for point clouds $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ respectively that are the output of a 3D descriptor. We match feature f_k^i from $\boldsymbol{F}_k$ with its nearest feature f_{k+1}^j from $\boldsymbol{F}_{k+1}$ based on an L_2-norm Nearest Neighbor metric (Mikolajczyk & Schmid, 2005):

$$f_k^i \square f_{k+1}^j \longleftarrow \arg \min_{n=1,2,...,N_j} \left(\left\| f_k^i - f_{k+1}^j \right\|_2 \right) < \tau \tag{5}$$

where *ι,j* are the feature indexes and the threshold τ is set to 1 to reduce the dependency between the threshold value and the metric used (O. Kechagias-Stamatis et al., 2018). We speedup the feature correspondence process of Eq. (5) by employing the Fast Library for Approximate Nearest Neighbors (FLANN) technique (Muja & Lowe, 2009). FLANN is a widely used library that performs fast approximate nearest neighbor searches in high dimensional spaces. It uses either a hierarchical k-means tree search with a priority search order scheme, or a multiple randomized kd-

trees scheme. The search scheme and the optimum FLANN parameters are automatically chosen from the FLANN library and depend on the dataset.

Feature matching is then performed by using geometric consistency checks (GCC) (Chen & Bhanu, 2007). Specifically, the correspondences obtained from FLANN (Eq. (5)) are clustered into hypotheses, using their true physical geometric consistency. Geometric consistency aims at reducing mismatches by grouping correspondences into clusters that are geometrically consistent. For the latter, a list of descriptor correspondences is created $H_u = \left\{ p_k^u, p_{k+1}^u \right\}$, where p_k^u and p_{k+1}^u are the *Target* correspondences at instance *k* and *k+1* from the FLANN matching stage (Eq. (5)):

$$H_u = \left\{ p_k^u, p_{k+1}^u \right\} \longleftarrow f_k^i \square f_{k+1}^j \tag{6}$$

Given a seed correspondence from H_u, the first cluster is initialized and all correspondences $H_v = \left\{ p_k^m, p_{k+1}^m \right\}$, $v < u$ not yet grouped that are geometrically consistent with the cluster are added to it. The consistency check for a pair of correspondences H_u, H_v is valid if the following distance relation holds:

$$\left| \| p_k^u - p_k^m \|_2 - \| p_{k+1}^u - p_{k+1}^m \|_2 \right| < \varepsilon \tag{7}$$

ε being the threshold tolerance for their consensus set that is experimentally set during tuning. Finally, the matched feature pairs $\{ f_k^i, f_{k+1}^j \}$ belonging to the cluster with the largest cardinality are considered as feature matches, while their associated vertices $\Omega = \left\{ p_k^i, p_{k+1}^j \right\}$ are considered as point correspondences.

It is worth noting that we use a GCC module instead of the popular Random Sample and Consensus (RANSAC) (Fischler & Bolles, 1981) because the latter has a longer execution time than GCC, which can be up to two orders of magnitude (Yang et al., 2018) and thus is inappropriate for odometry applications examined in this work. For completeness, in Section 4 we demonstrate the efficiency of GCC by substituting in the proposed odometry method the GCC module with RANSAC.

2.2.3 Recursive filtering

The aim of the recursive filtering is to remove the noise from a signal while retaining useful information. Hence, in the context of odometry, given the correspondences Ω we solve Eq. (2) by suggesting an adaptive recursive filtering scheme based on the quality of the correspondences Ω. The proposed architecture is based on a recursive optimal state estimation of the state variable $x_k = [r_{11}\ r_{12}\ r_{13}\ r_{21}\ r_{22}\ r_{23}\ r_{31}\ r_{32}\ r_{33}\ t_x\ t_y\ t_z]^T$ that encompasses the rigid transformation between $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ by exploiting the correspondences Ω. For our trials, we consider an initialization value for $x_k = [1\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 0\ 0\ 0]^T$. It should be noted that in contrast to our solution, (A. P. Rhodes et al., 2017) uses a Multiplicative Extended Kalman Filter (MEKF) to smooth *R** rather than estimating R^* as done in this work. The adaptive recursive filters evaluated in this paper are:

2.2.3.1 Adaptive *H*∞ Filter

The *H*∞ filter is a recursive optimal state estimator with x_k the state variable vector and $\psi_k = [x^k, y^k, z^k]^{\mathrm{T}}$ the measurement vector that contains the 3D coordinates of the point correspondences p_{k+1}^j belonging to $\boldsymbol{P}_{k+1}$, which are included in Ω. The adaptive *H*∞ filter is given by:

$$x_k = \Phi\ x_{k-1} + w_{k-1} \tag{8}$$

$$\psi_k = H_k x_k + v_k \tag{9}$$

where Φ is the state transition matrix and H the measurement model matrix. We set $\Phi = R_0^* = [I \mid T_0]$ with I the identity matrix and $T_0 = [0\ 0\ 0]^T$, w and v are the model and the measurement noise factors respectively with covariance matrices $W \square N(0, \sigma_w^2 J_{12})$ and $V \square N(0, M\sigma_v^2 J_3)$ where σ_w and σ_v are small positive values and J is the unity matrix. H_k contains the actual measured 3D coordinates of p_k^i belonging to $\boldsymbol{P}_k$ that are included in Ω:

$$H_k = M\begin{bmatrix} x_{k+1} & y_{k+1} & z_{k+1} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & x_{k+1} & y_{k+1} & z_{k+1} & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & x_{k+1} & y_{k+1} & z_{k+1} & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

with M an adaptive coefficient that aims at adjusting the measurement noise covariance based on the quality of the matched features $\{f_k^i, f_{k+1}^j\}$ defined as:

$$M = 10 \cdot \sum \left(\left| f_k^i - f_{k+1}^j \right| \right) \tag{11}$$

In contrast to the typical *H*∞ filter (Simon, 2000), in this work we suggest an adaptive measurement model matrix H_k. The constant in Eq. (11) is experimentally estimated to fine tune the overall filter performance. The problem that the *H*∞ filter is trying to solve is the $\min_{\hat{x}} \max_{w,v} G$ where G is defined as:

$$G = \frac{average\left(\left\| x_k - \hat{x}_k \right\| \right)_Q}{average\left(\left\| w_k \right\| \right)_W + average\left(\left\| v_k \right\| \right)_V} \tag{12}$$

subject to $G < 1/\gamma$, with Q being a weighting matrix and γ a small constant number representing the required accuracy of the filter. The *H*∞ filter equations solving Eq. (12) are:

$$L_k = \left(I - gQP_{k-1} + H_k^T V^{-1} H_k P_{k-1}\right)^{-1} \tag{13}$$

$$K_k = \Phi P_{k-1} L_k H_k^T V^{-1} \tag{14}$$

$$P_k = \Phi P_{k-1} L_k \Phi^T + W \tag{15}$$

$$\hat{x}_{k+1} = \Phi \hat{x}_k + K_k \left(\psi_k - H\hat{x}_k\right) \tag{16}$$

where $Q = Idt$ with $dt = 10^{-5}$ and $g = 0.1$ being regulating parameters. The number of iterations of the *H*∞ filter is the cardinality of Ω and ultimately the final $\hat{x}$ after all iterations is transformed into R^*, which is input to Eq. (4) in order to estimate the LIDAR odometry. The parameters of the adaptive *H*∞ filter as well as of the rest adaptive filters evaluated in this work are tuned based on scenario 1.

2.2.3.2 Adaptive *Kalman* Filter

Using the same notation as for the *H*∞ filter, the *Kalman* filter is given by (Simon, 2001):

$$x_k = \Phi x_{k-1} + \mathrm{B} q_k + w_{k-1} \tag{17}$$

$$\psi_k = H_k x_k + v_k \tag{18}$$

with *B* a matrix and *q* a known input to the system. The *Kalman* filter equations are:

$$K_k = A P_k H_k^T \left(H_k P_k H_k^T + VM\right)^{-1} \tag{19}$$

$$\hat{x}_{k+1} = \left(\Phi \hat{x}_k + Bu_k\right) + K_k\left(\psi_k - H_k \hat{x}_k\right) \tag{20}$$

$$P_k = \Phi P_{k-1} \Phi^T + W - \Phi P_k \mathrm{H}_k^T V^{-1} H_k P_k \Phi^T \tag{21}$$

where K is the Kalman gain and P the estimation error covariance, with $\sigma_v = 1$ and $\sigma_w = 5 \cdot 10^{-3}$ that are experimentally defined to gain optimum odometry performance.

2.2.3.3 Adaptive *αβ* Filter

Using the same notation as for the $H\infty$ filter, the $\alpha\beta$ filter is given by (Penoyer, 1993):

$$x_k = x_{k-1} + V_k \Delta t \tag{22}$$

$$r_k = x_{k-1} - x_k \tag{23}$$

$$x_k = x_k + \alpha r_k \tag{24}$$

$$V_k = V_k - \frac{\beta}{\Delta t} x_k \tag{25}$$

where r_k is the residual error, $V_k = 1$, $\Delta t = 10^{-5}$, $\alpha = 10^{-6}$ and $\beta = M$ as computed by Eq. (11).

2.2.3.4 Adaptive State-Dependent Riccati Equation (*SDRE*) Filter

This is a non-linear filter (Arnold & Laub, 1984) that solves the equation:

$$A^T XA - X - A^T XB(B^T XB + R)^{-1} B^T XA = 0 \tag{26}$$

where $A = Q = I \cdot \sigma_w$ and $B = IM$.

3. Experiments

3.1 Experimental setup

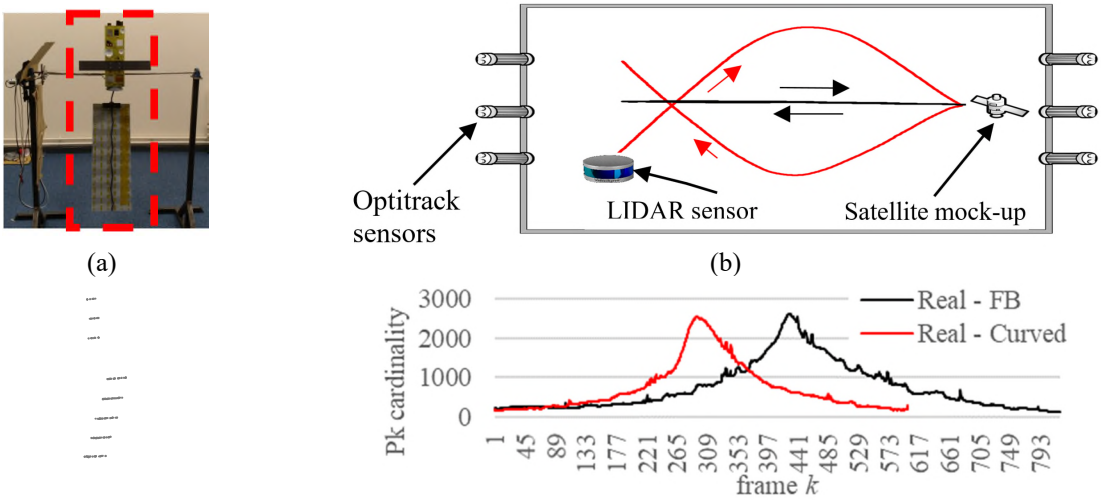
We evaluate the proposed architecture for LIDAR based uncooperative relative navigation on seven scenarios that consider as the *Target* platform one real space platform mock-up and four simulated space objects. We assume that the *Target* is not rotating in any significant way and the reference frame where odometry is searched for is fixed on the *Source* and rotating with it. However, for the scenarios involving real data, the Source platform has a minor tumbling during data acquisition.

The first two scenarios consider a moving *Source* platform that utilizes a VLP-16 Velodyne Puck Lite LIDAR to capture the point cloud of a scaled EnviSat mock-up *Target*. The ground truth of the *Source* is estimated by capturing its position via an Optitrack device ("Optitrack," 2018). Optitrack can provide in sub-millimeter accuracy the position of objects that are within its field of view and are visible in the Near Infrared (NIR) band. Therefore, we placed on the VLP-16 highly NIR reflective markers. The real data scenarios involve a straightforward – backward trajectory (*Real-FB*) and a highly curved forward – backward trajectory (*Real-Curved*). Both scenarios are challenging as they involve real point clouds and are presented in Figure 2 along with the point cloud cardinality per frame.

Main characteristics of these scenarios are the poor features on the *Target* point cloud that is mostly presented by flat surfaces. This is because the original Envisat, and thus our scaled mock-up, comprises of flat surfaces, on top of which several antennas are placed. The mock-up is acquired

by the 16 vertical beams of the VLP-16, constraining the vertical vertex cardinality to 16 vertices per acquisition instance neglecting the fine details of the mock-up. However, this situation can be considered as a simulation of a low-resolution point cloud that is acquired by a space graded LIDAR sensor at greater distances or by a low-cost low-resolution space-graded LIDAR device.

It is worth noting that the VLP-16 we used in our trials is not an optimum choice for spaceborne platforms mainly due the 360° horizontal field of view (FoV), the 16 vertical beams limiting the *Target* details per sweep in the vertical axis, and finally, due to the relatively short operating range compared to the majority of currently available space graded LIDAR sensors. Table 1 compares a number of space graded LIDAR sensors against the VLP-16 we use in our trials. From Table 1 it is obvious that for a short operating distance, the accuracy provided by space graded LIDAR sensors is higher compared to the commercial VLP-16. However, we believe that using the VLP-16 to acquire real data is both valid and valuable for the following reasons: First, we evaluate the performance of our architecture against the competitor odometry methods on real data that contain arbitrary minor *Source* tumbling. Second, we limit the 360° horizontal FoV by post processing the point cloud acquired and discard vertices that do not belong to the *Target*. Third, the low point cloud resolution may resemble a scenario of a space graded LIDAR at a great distance or utilizing a low-cost low-resolution sensor. Overall and given that e VLP-16 poses a decent cost-effective alternative to validate space navigation algorithms that has already been used by the relevant community (Opromolla, Di Fraia, et al., 2017), we believe that the quality of the acquired point clouds is sufficient for conceptual verification of the proposed architecture.



(a) (b) (c) (d)

Figure 2: *Real -FB* and *Real-Curved* scenarios (a) Scaled EnviSat satellite mock-up (b) top view of the acquisition setup and the trajectory plots of the LIDAR sensor, i.e. *Source*, moving along the black trajectory for the *Real-FB* and the red trajectory for the *Real-Curved* scenarios, (c) $\boldsymbol{P_k}$ point cloud example (d) $\boldsymbol{P_k}$ cardinality per frame k (colored arrows indicate the corresponding trajectory direction)

The simulation and the test data used in this paper present a few differences, without causing loss in performance, to the real space scenarios in terms of the LIDAR sensor characteristics used and in terms of the operation range. The LIDAR sensor technology envisaged/ used in space (NEPTEC, Jena-Optronik used on ESA programs) are denser, more accurate comparing to the technology used to generate our data in this paper. In space the close-range operation for this relative navigation problem could be from 100m down to 2m close to the target and the LIDAR technology mentioned above and used in space would cover up well to that range. All the experiments for relative navigation available in the literature are of similar range (few meters) as we proposed in our simulation and real data. The accuracy achieved by our algorithm in our tests is optimal and we strongly believe that it would be kept similar if we went for real space scenarios as in this case the sensor would be of better quality (range of operation, accuracy although with less or similar density as our LIDAR system we used in our experiment is not of great quality in terms of density)

and the noise is much less than that of our experimental data since in space the background is quite obscure and no noisy returns are expected comparing to our lab testbed.

Scenario three studies the odometry performance of a simulated Ellipse of Inspection (*Sim-EoI*) trajectory of the *Source* platform around the *Target*. The latter space platform is developed by Thales Alenia Space (France) and is inspired from the Globalstar-2 and Iridium constellations. The *Sim-EoI* trajectory is a realistic space-oriented scenario that considers the influence of the Earth's mass, the Sun's sunlight power with respect to each spectral band and the typical physical size of the *Target* and the *Source* platform. In the *Sim-EoI* trajectory the *Source* platform performs a complete translational motion along the ellipsoid. Figure 3 presents the satellite model, the *Sim-EoI* trajectory and the point cloud cardinality per frame.

Table 1: Comparison of existing space-graded LIDAR sensors and VLP-16

Sensor	FoV (H x V)[1]	Points	Maximum Range (m)	acquisition mode	refresh rate (Hz)	range accuracy at max range (cm)	angular resolution (rad)	power (W)	weight (kg)
LAMP (Kornfeld et al., 2003b; Liebe et al., 2003)	10°x10°	n/a	2500	scanning beam	10,000	≤ 2.6	10^{-3}	33	6.4
RVS-3000 (Jena Optronik, 2019)	40°x40°	n/a	100	scanning beam	1-4	n/a	n/a	50	8
DragonEye (ASC, 2019a)	45°x45°	32,768	1500	flash LIDAR	5-30	15	n/a	35	3
GoldenEye (ASC, 2019b)	45°x45°	32,768	3000	flash LIDAR	5-10	10	n/a	50	6.5
VLP-16	360°x30°	37,500[2]	100	16 vertical laser beams spinning 360°	5-20	≤ 3	H: $\leq 7x10^{-3}$ V:35 $x10^{-3}$	8	0.59

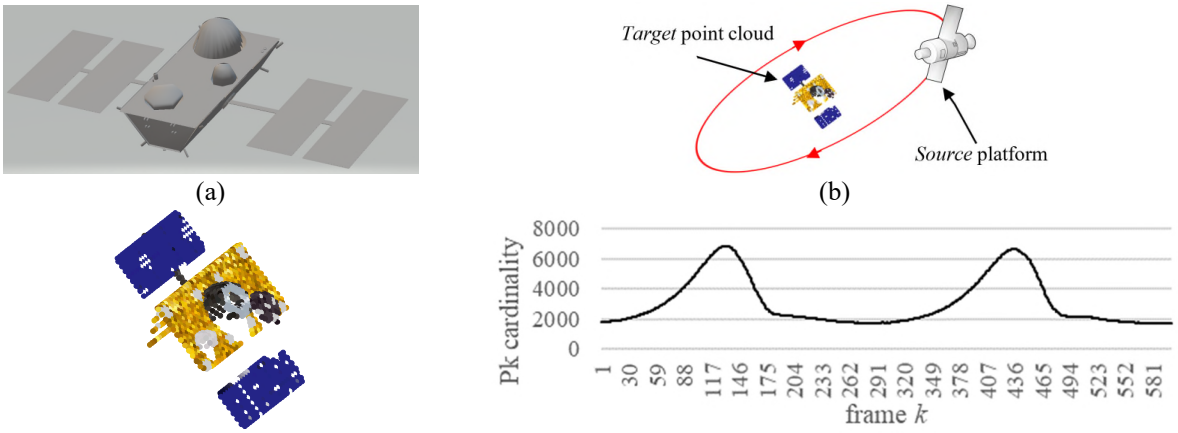
[1] Nomenclature: H stands for Horizontal, V for Vertical and n/a for not available

[2] In the single return mode VLP-16 generates approx. 300,000 points per second for a 360° x 30° FoV. For comparison reasons, for a hypothetical 45° x 30° FoV the points per second generated by VLP-16 would be 37,500

Scenario four named *Sim-Helical* is presented in Figure 4 and simulates a 3D helical trajectory of the *Source* platform that is approaching the Thales Alenia Space model acting as the *Target*. We create self-occluded point cloud views of the *Target* platform emulating realistic views of the virtual LIDAR placed on the *Source* platform by exploiting the Hidden Point Removal (HPR) algorithm (Katz, Tal, & Basri, 2007). The challenge in this scenario is the sparse $\boldsymbol{P}_k$ and complex trajectory as we want to push the limits of 3D space odometry and investigate the performance of the suggested pipeline.

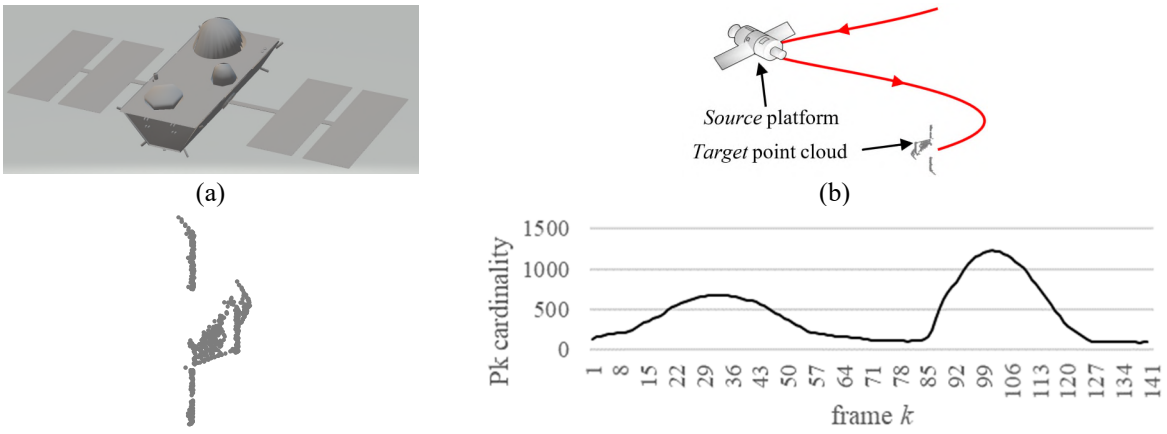
The following scenarios five to seven simulate the *Source* platform orbiting around the *Target* platform in an Ellipsoidal trajectory. Compared to the *Sim-EoI* trajectory, here the *Source* platform performs 2.5 times a complete translation along the ellipsoid aiming at evaluating the performance of the competitor methods on repetitive trajectories. Additionally, in scenarios five to seven consider as the *Target* platform the *Voyager* satellite, the *Orion* space capsule and the *Bennu* asteroid respectively. Computer Aided Design (CAD) models for all three space objects are obtained from (NASA, 2019), while the corresponding point clouds are created by applying the code of (Aldomà, 2011) with 800 views. Finally, similarly to scenario four, we use the HPR algorithm to create self-occluded point cloud views of the *Target* platform. The challenge in scenarios five to seven is multi-fold. First, opposing to the large size of the simulated *Envisat Target* platform evaluated in scenario three, the *Voyager* and the *Orion* space platforms are smaller providing a smaller point cloud cardinality. Second, *Bennu,* lacks distinctive features making the evaluation of the local feature descriptors evaluated quite challenging. Third, the *Orion* and *Bennu* point clouds are quite sparse. It should be noted that we intentionally tuned HPR to create sparse point clouds as we want to investigate the performance of the suggested pipeline in low resolution conditions simulating large *Source – Target* distances, which is the norm as the *Source* approaches

the *Target*, or alternatively we simulate exploiting a low-cost low-resolution LIDAR sensor. The simulated scenarios five to seven namely the *Sim-Voyager*, *Sim-Orion* and *Sim-Bennu* are presented in Figure 5.



(a) (b) (c) (d)

Figure 3: *Sim-EoI* scenario (a) satellite model (b) trajectory plot, (c) $\boldsymbol{P_k}$ example (d) $\boldsymbol{P_k}$ cardinality per frame k



(a) (b) (c) (d)

Figure 4: *Sim-Helical* scenario (a) satellite model (b) trajectory plot, (c) $\boldsymbol{P_k}$ example based on HPR simulation (d) $\boldsymbol{P_k}$ cardinality per frame k

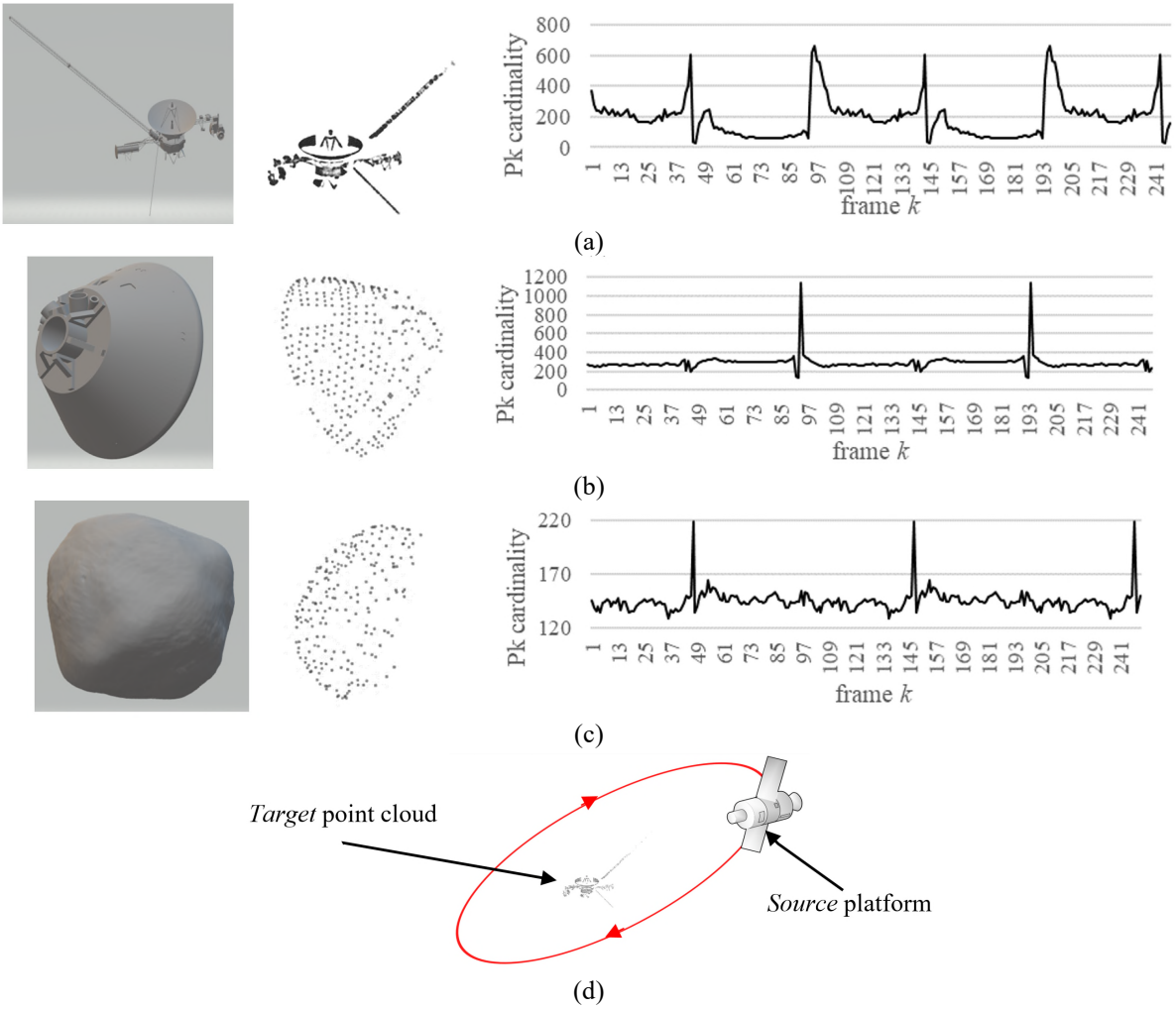


Figure 5: Simulated scenarios presenting the space object model, $\boldsymbol{P}_k$ based on HPR simulation and $\boldsymbol{P}_k$ cardinality per frame k respectively for the (a) *Sim-Voyager* scenario, (b) *Sim-Orion* scenario and (c) *Sim-Bennu* scenario, along with the generic Ellipsoidal trajectory of all three scenarios is presented in (d). For better readability we visualize in (d) only the Voyager model.

It should be noted that the CAD models we use to generate the *Target* point clouds are accurate, but the point clouds created are not ideal. This is because the method of (Aldomà, 2011) we use to generate the point clouds from CAD models and the HPR algorithm we employ to generate viewing dependent point clouds, affect the point cloud accuracy during reconstruction.

The differences between the real and the simulated *Target* point clouds of our trials can be summarized as follows. First, the vertical cardinality of the real LIDAR data is fixed to 16 vertices, while for the synthetic there is not such a constraint. Second, the level of details of the real data is

lower compared to the simulated. Third, real data during acquisition involve minor *Source* tumbling. Fourth, due to the real data acquisition setup, i.e. high LIDAR refresh rate and slow-moving *Source* platform, the inter-motion between $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ is lower compared to the simulated data where the fictitious dynamics of the *Source* and *Target* platforms are greater. However, despite these differences, both scenario classes are useful for evaluation. Except from these differences, both real and simulated data are sparse aiming at simulating large *Source* – *Target* distances or spaceborne LIDAR sensors that are less expensive and less accurate. Despite these differences, we believe that in the context of conceptual validation of our odometry method and in terms of comparing its performance against current solutions, both data modalities are acceptable.

Figure 6 shows the ground truth trajectory of each scenario along with the most accurate trajectory obtained by the proposed method, i.e. best performing feature description and filtering combination. In our trials we describe all vertices belonging to each point cloud $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ with the 3D descriptors introduced in Section 2.2.1. The proposed pipeline is in MATLAB while descriptors are implemented either in MATLAB or in C++/PCL using a MEX wrapper. The parameters of each descriptor are fixed either to the ones originally proposed by their authors or to their PCL implementation (Alexandre, 2012; Guo et al., 2016; Odysseas Kechagias-Stamatis & Aouf, 2016). An exception is the support radius r that we tune individually for each descriptor during the *Real-FB* scenario.

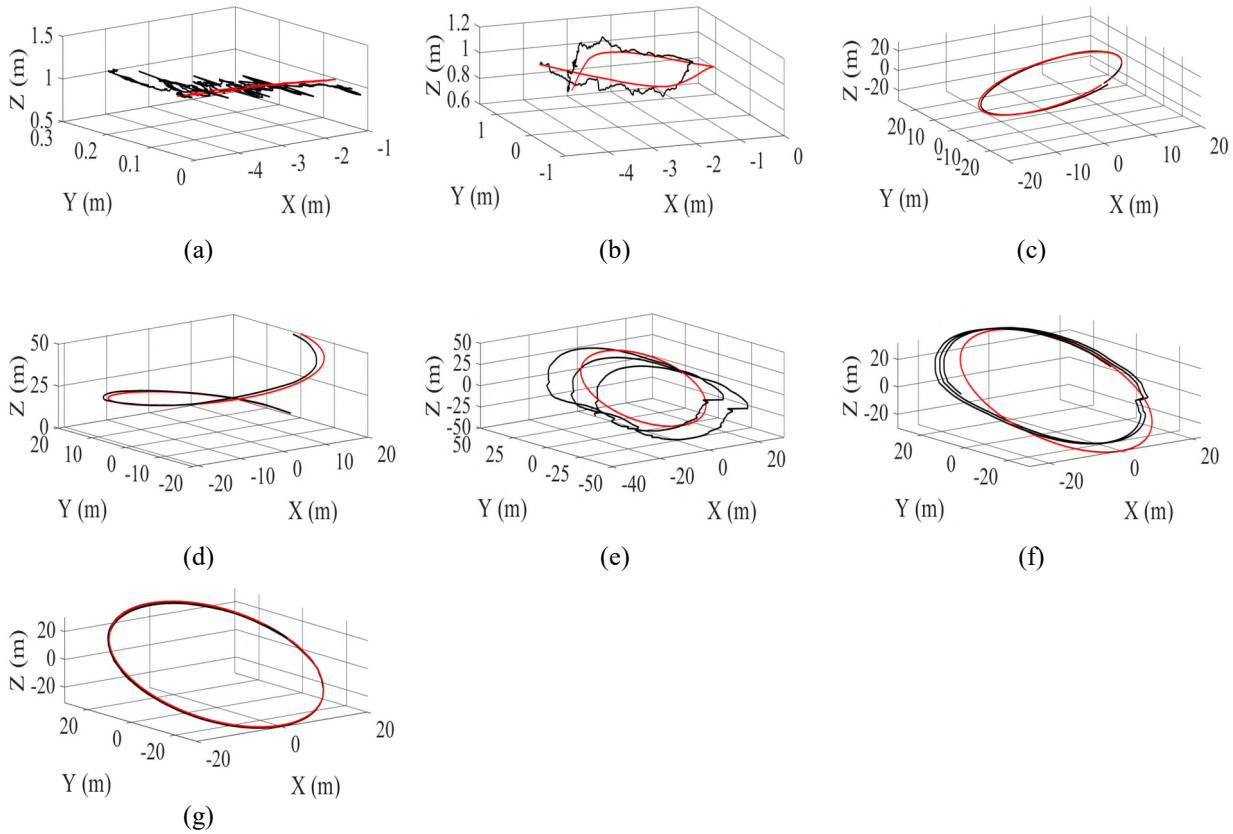


Figure 6: Ground truth trajectories (in red) along with the top performing method (in black) per the scenario (a) *Real – FB* scenario with HoD-S/ *αβ* (b) *Real – Curved* scenario with HoD-S/ *αβ* (c) *Sim – EoI* scenario with HoD-S/ *αβ* (d) *Sim-Helical* scenario with HoD-S/ *αβ* (e) *Sim – Voyager* scenario with HoD-S/ *SDRE* (f) *Sim – Orion* scenario with SHOT/ *Kalman* (g) *Sim – Bennu* scenario with HoD-S/ *αβ*. The *Real-FB* scenario considers a forward – backward motion, while the *Sim – Voyager, Sim – Orion* scenario and *Sim – Bennu* scenarios involve 2.5 times a complete motion translation along the elliptical trajectory.

Indeed, we use the *Real-FB* scenario 1 not only to evaluate the odometry performance of each method but also to tune the free parameters of every module of the proposed architecture and the parameters of the competitor methods presented shortly. Regarding the proposed method, we tune the encoding radius r of each descriptor, the geometric consistency threshold tolerance ε of eq. (7) and the regulating parameters of each adaptive filtering scheme. Tuning is performed via a *try and evaluation* scheme aiming to achieve the lowest possible translational and rotational error and also the lowest possible processing time. It should be noted that during tuning, odometry accuracy and

computational burden are of equal importance. Regarding the sensitivity of each method to these parameters:

a. Encoding radius: We confirm that SHOT is fairly stable and is less affected by the encoding radius, while TriSI, FPFH and RoPS gain a peak performance and then drop (Guo et al., 2016, 2015). During tuning, we identified this peak performance at $20 \times Tr$, with Tr the average $\boldsymbol{P}_{k+1}$ resolution. Regarding HoD and HoD-S, as the encoding radius increases these tend to increase their encoding capability and provide more distinctive features. However, regardless of the descriptor, as the encoding radius increases the processing time to compute the descriptor increases exponentially (Guo et al., 2016). Thus, the encoding radius of HoD and HoD-S is set to $80 \times Tr$ to balance odometry performance with computational burden. Table 2 presents the parameters of each descriptor evaluated in this work along with the main traits per method. It is worth noting that even though USC and 3DSC are widely used, our initial results during the tuning process of each descriptor showed that both failed to provide valid correspondences due to the highly sparse *Target* point cloud where USC and 3DSC are quite sensitive confirming (Guo et al., 2016).

Table 2: 3D descriptors evaluated

Descriptor	Descriptor Length	radius r of volume $\boldsymbol{V}$	Implementation platform	Operating principle	Robustness	Sensitivity
SHOT	352	$20 \times Tr$	C++ (MEX wrapper)	Angular variations	Gaussian noise, Shot noise	occlusion, clutter
TriSI	675	$20 \times Tr$	MATLAB	Accumulating points	Gaussian noise, Shot noise, resolution variation	occlusion, clutter
HoD-S	40	$80 \times Tr$	MATLAB	L_2-norm distances with coarse encoding	Gaussian noise, Shot noise, resolution variation	occlusion, clutter
HoD	240	$80 \times Tr$	MATLAB	L_2-norm distances with coarse and fine encoding	Gaussian noise, Shot noise, resolution variation	occlusion, clutter
FPFH	33	$20 \times Tr$	C++ (MEX wrapper)	Angular variations	resolution variation	Gaussian noise, Shot noise, occlusion, clutter
RoPS	135	$20 \times Tr$	MATLAB	Low order statistics	Gaussian noise	Shot noise, occlusion, clutter

b. Geometric consistency threshold: This defines how strict the geometric point-pair distance comparisons are (eq. (7)). The threshold dependents on the encoding quality of the 3D descriptor and on the characteristics of the $\boldsymbol{P}_k$ such as being sparse. Hence, to increase the robustness of our method to sparse *Target* point clouds and to potential feature description mismatches, we use an adaptive threshold that is set to $2 \times Tr$, with Tr the average $\boldsymbol{P}_{k+1}$ resolution. Due to its adaptive nature, the sensitivity of the geometric consistency threshold is relatively low.

c. Adaptive filtering regulating parameters: These define for each iteration how much the measurements, i.e. the feature correspondence coordinates for our odometry architecture, affect the prediction step, i.e. the updated R^*. These require to be finely tuned as the output of the filter

is quite sensitive to these parameters. Tuning is performed based on the *Real-FB* scenario.

Opposing to current literature (Opromolla et al., 2014, 2015a; Opromolla, Fasano, Rufino, & Grassi, 2015b; Woods & Christian, 2016) we compare the accuracy of the suggested architecture not only against an optimally tuned ICP point-to-point registration process, but also against point-to-plane ICP, point-to-point and point-to-plane Sparse ICP (Bouaziz, Tagliasacchi, & Pauly, 2013), x84 ICP (Fusiello, Castellani, Ronchetti, & Murino, 2002) and S4PCS (Mellado, 2017). To the best of our knowledge such an extended comparison has not been presented yet in the current literature.

For completeness, the ICP point-to-point variant aims at aligning $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$, where $\boldsymbol{P}_k$ is kept fixed and $\boldsymbol{P}_{k+1}$ is transformed via the R^* to match $\boldsymbol{P}_k$. During each transformation, ICP iteratively revises R^* (eq. (1)) to minimize the Euclidean point-pair distances between $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$. The point-to-point ICP, presented thereafter as ICP point, is a 4-step process:

a. Each vertex $p_{k+1} = (x_{k+1}, y_{k+1}, z_{k+1})$ belonging to $\boldsymbol{P}_{k+1}$ is matched to its closest point in $\boldsymbol{P}_k$ using a Euclidean distance metric. A match (inlier) is considered if the absolute Euclidean translational distance is less than 0.01.

b. The matrices R and T of R^* (eq. (1)) are estimated using a root mean square point-to-point distance metric minimization method that will best align each $p_{k+1} = (x_{k+1}, y_{k+1}, z_{k+1})$ with its matched vertex in $\boldsymbol{P}_k$.

c. The estimated R^* from step (b) is applied on $\boldsymbol{P}_{k+1}$.

d. If for the three latest consecutive iterations the average absolute translational and rotational difference between $\boldsymbol{P}_k$ and the transformed $\boldsymbol{P}_{k+1}$ of step (c) is less than 0.01 and 0.009 respectively, or the number of iterations has reached a pre-defined number, then ICP stops iterating. The output

R^* from ICP is the latest estimated R^*. If these thresholds are not met, then ICP re-iterates starting from step (a).

Current literature proposes various numbers of ICP iterations ranging from as few as 12 (Opromolla et al., 2014), up to 20 (L. Liu, Zhao, & Bo, 2016) or 30 (Opromolla et al., 2015b), aiming at balancing the odometry accuracy and the computational burden imposed by ICP. Spurred by the current literature trends, in this work the number of iterations for ICP is set to 20. We also evaluate a second variant of ICP, namely the point-to-plane (presented thereafter as ICP plane), that estimates the $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ point correspondences and then calculates the error metric based on the distance of the tangent plane of the corresponding point. For this ICP variant we use the same thresholds as for the ICP point, while the number of maximum iterations is increased to 1000. To compensate the sparse nature of the *Target* point cloud, we also compare the proposed architecture against Sparse ICP (Bouaziz et al., 2013) in a point-to-point and a point-to-plane scheme, presented in the evaluation section as S-ICP point and S-ICP plane respectively. We use the code of (Langlois, 2018) with 20 iterations. For completeness, an analysis between the interaction of the number of iterations for all ICP variants with the odometry performance and processing time is presented in Section 4.

Furthermore, in our experimental section we also evaluate the x84 ICP variant (Fusiello et al., 2002). The difference between ICP and x84 ICP is that the former uses as a threshold to reject outliers based on the standard deviation of the point pair distances, while x84 ICP the median absolute deviation. For the x84 ICP we use the code of (Birdal, 2015) that is properly modified to facilitate the x84 ICP outlier rejection scheme. The maximum number of iterations is set to 20. We also challenge our proposed method against S4PCS which is an optimized version of the 4PCS global registration technique (Aiger, Mitra, & Cohen-Or, 2008). 4PCS is an iterative process that

extracts all coplanar 4-point sets from $\boldsymbol{P}_{k+1}$ that are approximately related by a rigid transformation to a given planar 4-points from $\boldsymbol{P}_k$. The operating principle of 4PCS is that under rigid motion between $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ a number of coplanar 4-point sets from $\boldsymbol{P}_{k+1}$ remain invariant under affine transformations. Thus, 4PCS estimates R^* between the randomly chosen coplanar 4-point sets from $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ and retains the optimum R^* based on a similarity score. S4PCS affords a speedup over 4PCS by eliminating redundant 4-point congruent sets, i.e. sets that are related by a rigid transformation, and by indexing the coplanar 4-point sets for fast retrieval. In this work we use the S4PCS code of (Mellado, 2017). Based on the tuning scenario *Real-FB*, we set an estimated overlap ratio of 70% between $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ and a registration accuracy δ=0.01, while the rest of the parameters are the default ones.

Finally, we also challenge our technique against the global/ local inlier voting (Buch, Yang, Kruger, & Petersen, 2014) that involves a dual layer correspondence check comprising of GCC and RANSAC followed by a singular value decomposition scheme. However, for better readability comparison is presented in a compact form in Section 4.

Alternatively, current literature also suggests employing computer vision concepts (A. Rhodes et al., 2016; A. P. Rhodes et al., 2017) and specifically proposes applying coarse *Target* pose estimation utilizing the Oriented Unique and Repeatable Clustered Viewpoint Feature Histogram (OUR-CVFH) regional descriptor (Aldoma, Tombari, Rusu, & Vincze, 2012) or the local feature descriptor Spin Images (Andrew Edie Johnson & Hebert, 1998), that are then followed by a fine registration process based on ICP point-to-point. Regarding OUR-CVFH, in our preliminary trials we observed that it is not able to cluster the sparse *Target* point clouds of our scenarios and considered the entire *Target* point cloud as a single cluster. This inevitably degraded OUR-CVFH to the inferior VFH descriptor that did not manage to provide any feature matches. Additionally,

in our preliminary trials, we confirmed (A. Rhodes et al., 2016) stating that the Spin Images descriptor is affected by the low resolution of $\boldsymbol{P_k}$ and thus is not an optimum solution for space odometry. In fact, we identified that Spin Images suffer from an ambiguous local reference axis (LRA), the estimation of which is affected by a sparse *Target* point cloud. Due to the drawbacks of Spin Images and OUR-CVFH on sparse point clouds, in our experimental section we will not evaluate these two descriptors.

It should be noted that in any case, i.e. evaluating the proposed odometry architecture or the ones suggested by the literature, we consider that the initial position and pose R^* of the *Source* is known and that all methods aim to build-up an accurate odometry solution. It is assumed that this prior knowledge is obtained before commencement of any of the methods examined in this work and can be based on Earth-based range and Doppler measurements or spacecraft-based optical images (A. B. Dietrich & McMahon, 2018). Additionally, it is worth noting that changing the point cloud resolution, accuracy and adding noise presents an additional challenge to the already challenging scenarios. However, the impact of each of these nuisances is related to their severity and to the robustness of the 3D feature descriptors and GCC module. For better readability and to keep the paper in a reasonable length, we will not involve any robustness to nuisance factors evaluation.

3.2 Evaluation criteria

Odometry performance metrics are based on the average and the maximum tri-axial translational error between the ground truth (GT) position of the moving platform as tracked by the Optitrack and the estimated one as provided by the proposed architecture:

$$e_{avg}^{T} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(\left\|avg(T_k^{GT}) - avg(T_k^{opt})\right\|^2\right)} \tag{27}$$

$$e_{\max}^{T} = \sqrt{\max\left(\left\|avg(T_k^{GT}) - avg(T_k^{opt})\right\|^2\right)} \tag{28}$$

where N is the number of point cloud instances per scenario, $T_k^{opt} = T_{LIDAR}^{opt} \cdot T_k^{LIDAR}$ is the transformed translation at instance k from the LIDAR reference frame to the Optitrack reference frame in order to make its comparison with the GT translation applicable, and $avg(\cdot)$ averages the tri-axial translation into a single value. In addition to these metrics, we also calculate the drift, i.e. RMSE, between the estimated endpoint and the GT endpoint, the corresponding translational error T_{error} as a percentage over the distance travelled and the average processing time t per *Target* scene.

Similarly to eq. (27) and eq. (28), we calculate the average rotational error:

$$e_{avg}^{R} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(\left\|avg(R_k^{GT}) - avg(R_k^{opt})\right\|^2\right)} \tag{29}$$

$$e_{\max}^{R} = \sqrt{\max\left(\left\|avg(R_k^{GT}) - avg(R_k^{opt})\right\|^2\right)} \tag{30}$$

Additionally, we also use the pose-graph comparison method of (Burgard et al., 2009)

$$e^{RT} = \frac{1}{N}\sum_{k=1}^{N}\left(trans\left(R_{GT}^{*} \circ R_k^{*}\right)^2 + rot\left(R_{GT}^{*} \circ R_k^{*}\right)^2\right) \tag{31}$$

where $trans(\cdot)$ is the translational and $rot(\cdot)$ the rotational part of the R^* and R_{GT}^{*} transformation matrices respectively between $\boldsymbol{P_k}$ and $\boldsymbol{P_{k+1}}$ and $\circ$ is the inverse motion composition operator as defined in (Lu & Milios, 1997). The advantage of using the e^{RT} metric is two-fold. First, it uses only relative relations between the R^* and R_{GT}^{*}, and thus it is more objective as it is not influenced by any reference frame. Second, it encompasses in a single value both translational and rotational

errors affording an easy but complete comparison between the competitor methods. Note that the GT translation T_i^{opt} and GT rotation R_i^{opt} are provided by the Optitrack. Thus, for the synthetic scenarios the corresponding GT translation and rotation is the fictitious position per instance *k* of the *Source* platform. Each of these performance metrics presented next are averaged over 20 simulations with a standard deviation that is in the order of 10% the average value. However, for better readability and to preserve the number of figures and tables to a reasonable amount, we will only present the averaged performance metrics.

Given the large amount of feature description and recursive filtering combinations evaluated in our odometry architecture, challenging our method against six methods offered by current literature and utilizing a large number of performance metrics, for the sake of readability, all scenarios will be analyzed based on the e^{RT} metric. We prefer e^{RT} against the other metrics as is encompasses both translational and rotational errors in a single metric. However, for the sake of completeness, all error metrics and the corresponding odometry plots are presented in the Appendix section.

3.3 Odometry trials on real LIDAR data

3.3.1 *Real-FB* trajectory

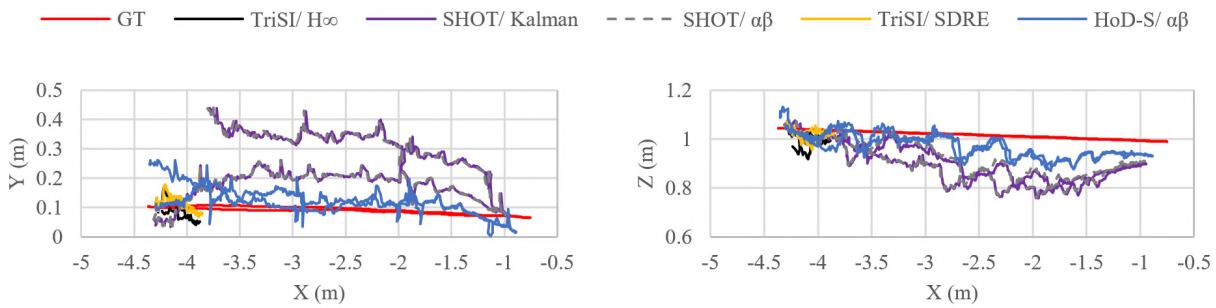
This scenario considers a straight-line forward - backward motion of the *Source* with respect to the *Target*. Point clouds are acquired by a LIDAR device that is placed on the *Source* platform. The average point cloud size per *Target* platform comprises of 882 vertices demonstrating the sparse nature and the limited structure of the EnviSat *Target* point cloud (Figure 2 (c)). Performance metrics are presented in Table A1 (see Appendix) with the top performing method per error being highlighted in red. Additionally, Figure A1 presents the corresponding odometry

trajectory obtained per 3D descriptor and recursive filtering method along with all competitor methods. For better readability, methods with large errors are discarded and not presented in the corresponding plots.

From Table A1 it is evident that quite a few 3D local description and recursive filtering combinations are more accurate compared to the majority of the competitor methods. Top performance affording lowest e^{RT} at a low processing time, is provided by the combination of HoD-S with the adaptive $\alpha\beta$ filter, i.e. HoD-S/ $\alpha\beta$. It should be noted that in terms of error, SHOT/ $\alpha\beta$ is slightly more accurate than HoD-S/ $\alpha\beta$. However, the latter is three times faster to execute than SHOT/ $\alpha\beta$ and thus HoD-S/ $\alpha\beta$ is considered as an overall more appealing option. This is because HoD-S neglects estimating a LRF/A and thus is more robust to sparse point clouds. Additionally, neglecting a LRF/A and relying on a small description length, makes HoD-S the most processing efficient descriptor among the ones evaluated. In terms of pure odometry accuracy, SHOT attains the lowest e^{RT} error, but interestingly, from Table A1 it is evident that HoD-S is the most accurate descriptor in terms of translational error, and TriSI in terms of rotational error. This reveals that SHOT achieves the best balance between the translational and the rotational error. It is worth noting that current literature offers several evaluations on the robustness of 3D local feature descriptors to mesh/ point cloud resolution variations (Guo et al., 2016, 2015, 2013a; Salti et al., 2014). However, these papers examine the case where the features from a dense point cloud, e.g. *Source*, are matched against the features of a sparse point cloud, e.g. *Target*. In contrast to this, here we involve a *Source* and a *Target* point cloud that present some variation and are both sparse. Finally, FPFH presents the largest errors due to its short feature descriptor (33 elements) and its sensitive LRF to low resolution point clouds. Interestingly, Table A1 presents TriSI as the optimum descriptor for the adaptive *SDRE* filter. However, from Table

A1 and Figure A1, is it clear that TrISI provides one of the largest translational errors indicating that the e^{RT} metric is biased by the very low rotational errors of TriSI. In fact, that sparse nature of $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ forces TriSI to fail providing enough geometrically consistent feature matches to the recursive filtering module, and thus the latter does not iterate properly, forcing the estimated R^* to preserve its initialization value.

In terms of processing efficiency, as expected HoD-S with HoD are the fastest to execute as both neglect a LRF/A estimation, and RoPS with TriSI are the least processing efficient due to involving a complex LRF estimation process. It is worth noting that despite HoD-S and HoD being implemented in MATLAB, these are still faster to execute than FPFH and SHOT that are in implemented in C++/PCL. From Table A1, we also observe that the adaptive *αβ* and *Kalman* filters afford similar accuracy, with *αβ* being slightly more accurate, demonstrating that due to the minor motion between $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$, a linear motion estimation model with two parameters, as is the *αβ* filter, is sufficient for space odometry. For better visualization of which filter is performing better, in Figure 7 we compare the best feature descriptor from each filter.



(a) (b)

Figure 7: Forward – Backward *Real-FB* scenario, odometry performance per filtering method with the lowest e^{RT} metric along with the overall top performing in terms of accuracy and processing efficiency HoD-S/ *αβ*. SHOT/ *Kalman* and SHOT/ *αβ* present quite similar results highlighting the effectiveness of a linear 2-parameter motion estimation model. Despite TriSI presenting the lowest e^{RT} error for the SDRE filter, the error is biased by a low rotational error as the translational one is quite large.

Regarding the competitor methods, the ICP point and plane variants are the fastest to execute and are the most accurate methods among all competitor techniques evaluated. Despite that, in terms of accuracy these are still inferior to most of the proposed combinations. From Table A1 and Figures 7 (i) and (j), we observe that ICP point is slightly more accurate than ICP plane, despite $\boldsymbol{P}_k$ mostly comprising of flat surfaces. An explanation is that $\boldsymbol{P}_k$ has quite a few groups of vertices that are horizontally aligned (Figure 2 (c)) with these groups having a substantial vertical distance between them. Hence, depending on the frame-to-frame motion of the *Target* point cloud, the plane estimation involved in the ICP plane process is affected by this structure, degrading the overall performance of ICP plane. Similarly, ICP point fails to present an accurate odometry solution because the structure of $\boldsymbol{P}_k$ forces this ICP variant to converge to an appealing solution only if most of these "horizontal groups of vertices" are aligned. Accordingly, ICP x84 and both S-ICP variants fail to provide an accurate odometry. In addition to that, sparse point clouds force ICP to perform suboptimal (Opromolla, Fasano, et al., 2017). For the S-ICP variants, due to the vertex geometry these in most cases provide an R^* with zero rotation and translation, which is clearly presented in Figure A1 (i) and (j).

3.3.2 *Real-Curved* trajectory

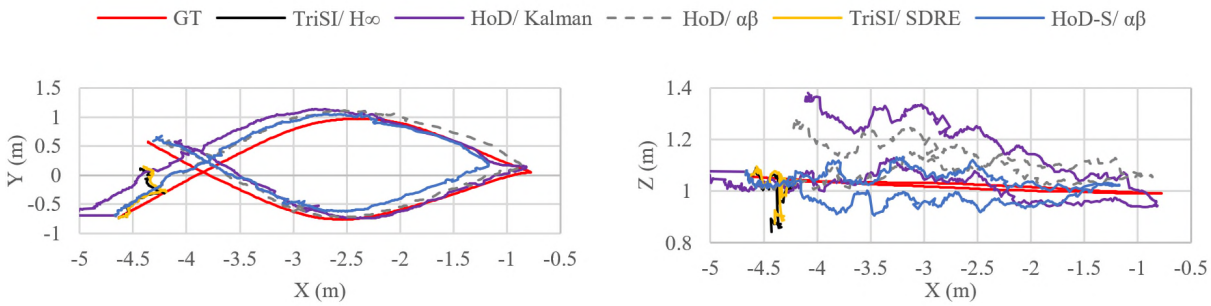
This scenario also considers real LIDAR data but is more challenging compared to the *Real-FB* scenario because in addition to the poor structure and sparse *Target* point cloud, this trajectory is also highly curved. Hence, most of the descriptors attains a larger e^{RT} error compared to the *Real-FB* scenario. From analyzing Table A2 and Figure A2, we conclude that HoD/ *Kalman* attains the lowest error with HoD-S/ $\alpha\beta$ following next. However, given that the latter requires half the processing time of the former, and that the e^{RT} based performance difference between these two

combinations is relatively small, we conclude that the overall optimum choice for this scenario is the HoD-S/ $\alpha\beta$. A commonality between the two real scenarios is that Trisi affords the lowest e^{RT} error for a few filters, but again this metric is biased by the low rotational error of TriSI. In fact, TriSI has the highest e^{T}_{avg} odometry error for the reasons already presented in Section 3.3.1.

For the adaptive *Kalman* and $\alpha\beta$ filters, HoD-S is the optimum choice as it affords the lowest e^{RT} error and simultaneously requires the lowest processing time. Finally, similarly to the *Real-FB* scenario, in this scenario FPFH also attains the largest errors, which are more evident in the Z-axis. In terms of processing efficiency, except for TriSI, the hierarchy is the same as for the *Real-FB* scenario.

Regarding the accuracy of the recursive filtering schemes, we notice from Table A2 and Figure A2 that the adaptive $\alpha\beta$ filter gains the lowest e^{RT} error with the adaptive Kalman to follow. This is due to the minor motion between $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ that can be simulated with a 2-parameter linear model. To highlight which filter is performing better, in Figure 8 we compare the best feature descriptor from each filter. A further analysis at a higher level involving the overall performance of each descriptor and filtering method over all scenarios is presented in Section 4.

As expected, the processing burden of every 3D local feature descriptor and recursive filtering combination is higher than the time needed by any ICP variant, mainly due to the feature matching/ correspondence grouping process. Except from being processing efficient, all ICP methods are less accurate posing a non-optimal odometry solution for the reason presented in Section 3.3.1. Similarly to the *Real-FB* scenario, in this trajectory ICP x84 and both S-ICP variant also to fail to provide an accurate odometry.



(a) (b)

Figure 8: *Real-Curved* scenario, odometry performance per filtering method with the lowest e^{RT} metric along with the overall top performing in terms of accuracy and processing efficiency HoD-S/ *αβ*. TriSI/SDRE presents the lowest e^{RT}, however it is biased by a low rotational error as the translational one is quite large forcing this combination to preserve the filter's initialization *R**.

3.4 Odometry trials on synthetic LIDAR data

3.4.1 *Sim-EoI trajectory*

This is a synthetic data scenario that involves a simulated Ellipsoidal trajectory of the *Source* platform around the *Target* platform. This is a realistic scenario with an average *Target* point cloud size of 3099 vertices that are affected by several external parameters described in the introduction of Section 3. Results on the Sim-EoI trajectory are presented in Table A3 and Figure A3.

The combination of HoD-S with the adaptive *αβ* filter exhibits the highest accuracy. However due to the large vertex cardinality of $\boldsymbol{P}_k$ and the number of correspondences, the processing time required by HoD-S is now one of the highest. Computationally, FPFH affords overall the smallest burden due to the fewer number of correspondences it provides, but its accuracy is one of the poorest. On the contrary, RoPS and TriSI require the highest processing time due to the processing burden implied by their complex LRF estimation, in combination with the high cardinality of the *Target* point cloud. In terms of translational error, HoD-S presents the lowest error, while similarly to the *Real-FB* and *Real-Curved* scenarios, TriSI affords the lowest rotational error. However, in

contrast to TriSI's translational performance presented in the scenarios earlier, here it achieves a moderate accuracy. This is because as in this scenario TriSI has 1572 geometrically consistent correspondences Ω, and thus the recursive filtering scheme performs 1572 iterations settling to a more accurate rigid body transformation R^*.

Regarding the accuracy of the recursive filtering schemes, we notice from Table A3 that the adaptive $\alpha\beta$ filter achieves the lowest average error, with the $H\infty$ and *SDRE* filters following closely. Surprisingly, the adaptive *Kalman* filter fails to provide an appealing accuracy with any of the evaluated descriptors. In Figure 9 we compare the best feature descriptor from each filter. The Kalman filter constantly attains the highest errors for all feature descriptors, while the latter ones perform quite well with the rest of the filtering methods. This is because all modules within the proposed architecture are tuned based on the sparse *Real-FB* scenario. Considering that the *Sim-EoI* scenario examined here is denser, we observe that the parameters of the Kalman filter are more sensitive to the point cloud cardinality, while the rest of the filters are more robust. Thus, for this scenario Kalman requires to have its parameters re-tuned, loosing though its generalization. For the sake of the latter, we preserve the same parameter values as already tuned in the *Real-FB* scenario.

Concerning the performance of the competitor methods, both S-ICP variants provide a highly accurate odometry in an appealing execution time. The remaining competitor techniques attain an accuracy that is one order of magnitude larger compared to S-ICP. This is because despite this trial having more vertices compared to the rest of the scenarios, the number of vertices is still small for an ICP process to iterate properly (Opromolla, Fasano, et al., 2017).

In any case, it should be noted that the errors in this trial are larger compared to the real scenarios mainly due to the length of each scenario as it is well known that odometry errors build-up as the length of the trajectory increases. Therefore, at this point where we investigate the absolute performance per scenario, cross comparing each method among all scenarios is not realistic. However, in Section 4 we normalize the all the errors attained by each method and over all scenarios, such as to make a cross-scenario comparison possible.

Despite a few descriptors and filtering combinations achieving an appealing accuracy, the processing time required by each combination does not pose an overall valid choice for space applications. Hence, for point clouds with a large cardinality as in this trial, adopting a subsampling scheme or utilizing a keypoint detection method shall afford a lower computational burden. However, for the sake of generalization and given that in the rest of the trials such an additional process within our architecture is not mandatory, we retain for this trail our architecture as it is and demonstrate it weakness.

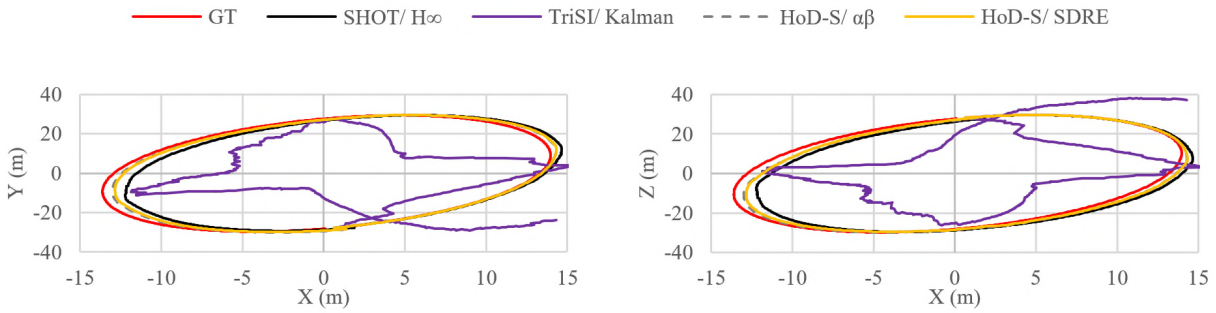


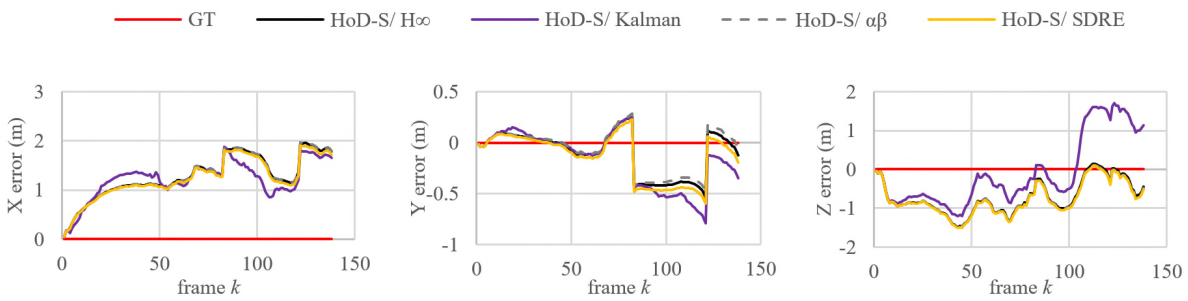
Figure 9: *Sim -EoI* scenario, odometry performance per filtering method with the lowest e^{RT} metric. Due to the $\boldsymbol{P}_k$ cardinality of this scenario, the parameters of Kalman fail to attain an appealing odometry accuracy.

3.4.2 *Sim-Helical* trajectory

This scenario involves synthetic data and is quite challenging due to the large curvature of the trajectory in the X-Y plane, the large translational disposition in all three axes and the small *Target* point cloud size that has an average size of only 438 vertices. This trajectory considers a 360° rotation and simultaneous translation as presented in Figure 6 (d). We create self-occluded point cloud views of the *Target* platform emulating realistic views of the virtual LIDAR placed on the *Source* platform by exploiting the HPR algorithm. Even though in the context of space odometry this trajectory might be extreme, we intentionally push the limits of odometry and investigate the performance of the evaluated methods. Table A4 presents the detailed performance metrics, while Figure A4 the error, i.e. difference of the GT trajectory and the estimated one per method, for each of the three axes. For this scenario we prefer presenting the corresponding errors per method rather than the estimated trajectory, due to the small errors attained by the majority of the feature descriptors and recursive filtering methods.

From Table A4 it is evident that the HoD-S descriptor attains the lowest e^{RT} error among all descriptors evaluated. Additionally, HoD-S achieves the lowest error per filter with any of the filtering methods that it is combined with. Next to follow is SHOT, which poses the second-best choice for each filter. From Table A4 and Figure A4 it is clear that TriSI is the least accurate descriptor as it achieves the highest e^{RT} error for any of the filters evaluated. Given that RoPS and TriSI share the same LRF estimation method and that RoPS gains a better odometry accuracy, the encoding capability of TriSI on this scenario is limited. This is because the Spin Images descriptor that is included in the TriSI descriptor is prone to highly sparse structures, in combination to the large frame-to-frame motion between $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$. Regarding processing efficiency, HoD-S is the fastest to compute because it neglects a LRF/A estimation process.

We further analyze the interplay between HoD-S, which is the descriptor offering the smallest errors, and the filtering methods evaluated in this work, and present the results in Figure 10. From this figure we observe that *H∞*, *αβ* and *SDRE* have a very similar performance, with Kalman being less accurate especially in the Z-axis.



(a) (b) (c)

Figure 10: *Sim -Helical* scenario, odometry performance per filtering method with the lowest e^{RT} metric

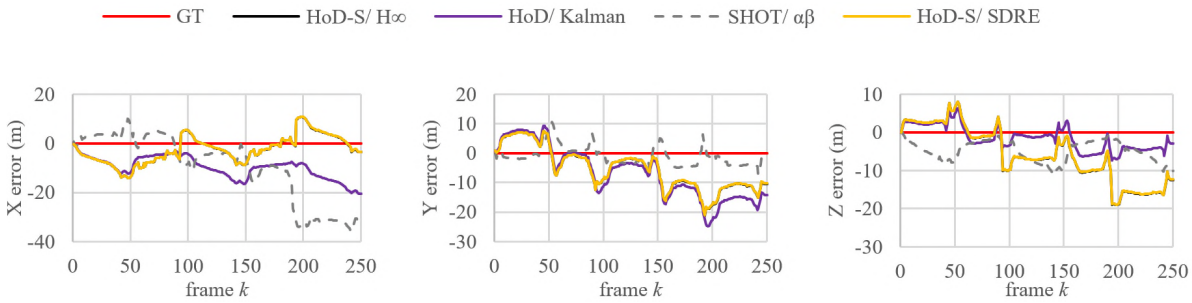
Considering the competitor methods, the top performing ICP plane has half the accuracy of HoD-S/ αβ. As already mentioned in Section 1, the proposed method can present an accurate odometry due to the feature matching and the geometric correspondence grouping schemes that provide to the recursive filter only well-established correspondences. These correspondences combined with the adaptive nature infused in the recursive filter afford an odometry trajectory with low errors. Despite that, ICP is highly processing efficient requiring only 18ms. Even though literature suggests that ICP attains a low accuracy on sparse point clouds (Opromolla, Fasano, et al., 2017) and we confirmed that in the previous trials, this trial despite being sparse, the frame-to-frame motion between $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ is such allowing ICP to settle to a more accurate solution.

3.4.3 *Sim-Voyager* trajectory

This scenario considers the *Source* platform orbiting in an ellipsoidal trajectory around the Voyager space platform. This is a synthetic scenario with an average $\boldsymbol{P}_k$ cardinality of 190 vertices, where the various *Target* platform poses are created using the HPR algorithm. In order to assess the performance our architecture on repetitive trajectories, this scenario considers the *Source* platform moving along the ellipsoidal trajectory 2.5 times. Thus, in contrast to the trajectories evaluated so far, this is the longest one with a GT length of 788 meters.

The results on this scenario are presented in Table A5 and Figure A5. As expected, due to the long trajectory length evaluated here, the performance of al methods is substantially higher. However, as already stated, a direct cross-scenario performance comparison is biased towards the shortest trajectories. Hence, in Section 4, we normalize the performance of each method such as to make it independent of the distance travelled and thus make a cross-scenario performance comparison feasible. The performance achieved by each method is quite similar among the filters evaluated. This demonstrates that establishing correct feature matches is very important as miss-matches will negatively influence the odometry output of the filter. However, as demonstrated in Section 34.1, for the Kalman filter the training and testing scenarios should involve a *Target* with a similar level of sparsity. Optimum scheme is the SHOT/ $\alpha\beta$, while the overall tope performing descriptor is SHOT with HoD following closely. However, due to the large computational burden of SHOT and given that the next best performing combination is HoD-S/ SDRE that is also 10 times faster to execute, we select the latter as the optimum method. Considering the performance of the filtering methods, the overall top performing is $H\infty$ with $\alpha\beta$ being the next optimum choice. In terms of processing efficiency, once again HoD-S and HoD are the fastest to compute. The least accurate method is HoD-S/ Kalman posing large translational and rotational errors.

Figure 11 presents the performance of the most appealing schemes per filtering method, where we observe that SHOT/ $\alpha\beta$ shows a substantial accuracy drop in the X-axis after frame 200, i.e. after two complete ellipsoidal translations. Additionally, Figure 11 highlights that all methods present a repetitive error that coincides with the corresponding relative *Source – Target* position.



(a) (b) (c)

Figure 11: *Sim -Voyager* scenario, odometry performance per filtering method with the lowest e^{RT} metric.

Considering the competitor methods, ICP point attains an e^{RT} error that is close but inferior to the well performing proposed combinations, with ICP plane following. However, as expected, both these ICP variants are much faster to execute. Despite the average $\boldsymbol{P}_k$ cardinality is only 190, both S-ICP variants are less accurate than their standard ICP counterparts because $\boldsymbol{P}_k$ is not sparse. Interestingly, all errors are sinusoidal-based that coincides with the relative *Source-Target* position. This is more evident for the S4PCS, where the error amplitude is the highest presented in Figure A5 (m)-(o).

3.4.4 *Sim-Orion* trajectory

This scenario is identical to the *Sim-Voyager*, but the Voyager satellite is substituted with the Orion space capsule. Main characteristics of this trial are the flat surfaces of Orion that include less distinctive features. The average $\boldsymbol{P}_k$ cardinality is 288. In terms of performance, Table A6 and

Figure A6 demonstrate that SHOT and RoPS provide the smallest e^{RT} errors with the former combined with the *Kalman* filter being the optimum choice for this scenario. In contrast to the previous scenarios, HoD and HoD-S perform poor highlighting that for point clouds that are mainly flat with poor distinctive features neglecting a LRF/A has a negative impact on the feature matching process. Similarly to the *Sim-Voyager* scenario, quite a few descriptor and filtering combination present the sinusoidal error. Due to the challenging nature of this scenario, HoD, HoD-S and TriSI present quite large errors and thus for the sake of readability these descriptors are neglected from several plots in Figure A6. In Figure 12 we present the top performing descriptor per filtering method. Interestingly, this figure indicates that all RoPS descriptors attain almost the same performance independently of the filter. Additionally, all RoPS based solutions and the SHOT/ *Kalman* achieve a similar performance in the Y-axis.

Considering the competitor methods, these are of inferior accuracy to the majority of the feature descriptor / recursive filtering combinations of the proposed method. Specifically, both ICP variants present a low translational error but due to their low rotational accuracy, these cannot be considered as overall optimum choices. On the contrary, S4PCS and both S-ICP variants attain low rotational errors and large translational errors.

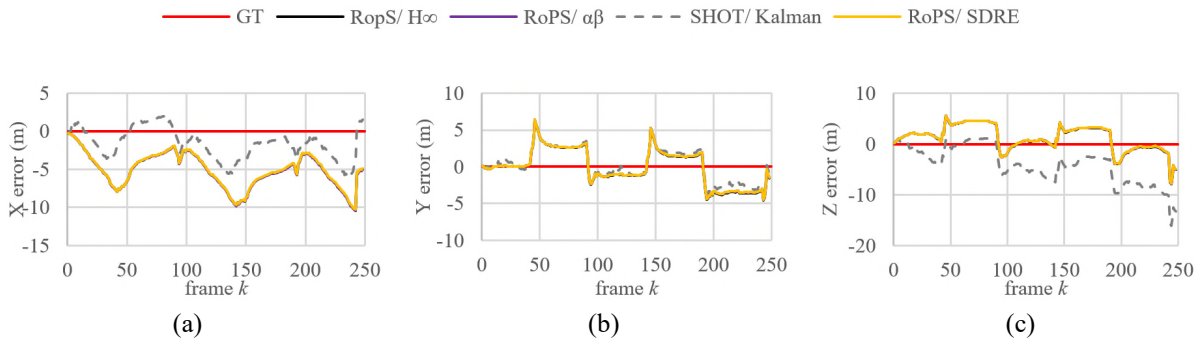
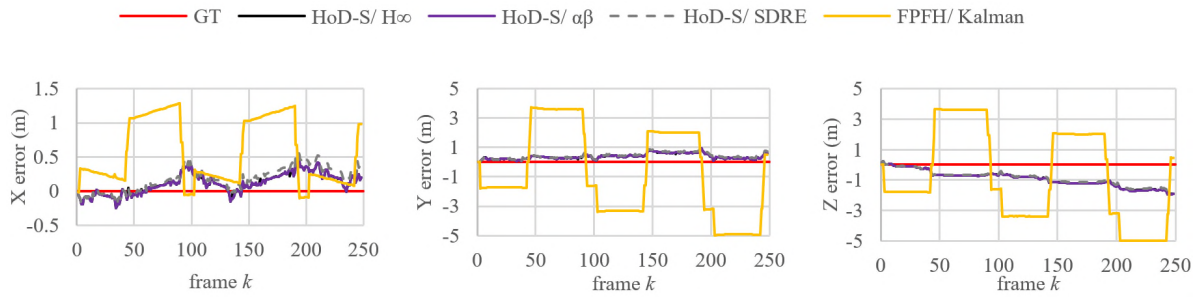


Figure 12: *Sim -Orion* scenario, odometry performance per filtering method with the lowest e^{RT} metric along with the overall top performing in terms of accuracy and processing efficiency HoD-S/ *SDRE*

3.4.5 *Sim-Bennu* trajectory

This also the same trajectory to the *Sim-Voyager* and *Sim-Orion* but considers the Bennu asteroid as the *Target* object. The main challenge of this trial is the very small average point cloud cardinality which is only 151. The results obtained are presented in Figure A7 and Table A7, where it is evident that for all filters but *Kalman*, HoD-S is the optimum choice. This is because, in contrast to the *Sim-Orion* scenario where the *Target* involves perfectly flat surfaces, the surfaces of Bennu despite being smooth, these still allow for HoD-S to efficiently encode the local region where it is applied. Considering the *Kalman* filter, FPFH attains the lowest e^{RT} error. However, it is obvious from Figure 13 that FPFH/ *Kalman* presents a relatively large periodic translational error, and thus this combination cannot be considered as an optimum choice. This error is because the FPFH descriptor lacks providing geometrically consistent point pair correspondences and thus the FPFH feature matches are rejected by the GCC module, forcing the proposed architecture to proceed with the initialization R^*, i.e. a unity rotation matrix and a zero-translation matrix.

Similarly to the previous trials, the competitor method do not pose an optimum choice as these do not offer a balanced rotational and translational error, forcing the e^{RT} error to be quite large.



(a) (b) (c)

Figure 13: *Sim -Bennu* scenario, odometry performance per filtering method with the lowest e^{RT} metric along with the overall top performing in terms of accuracy and processing efficiency HoD-S/ *SDRE*

4. Discussion

In Section 3 we presented the odometry performance of several feature descriptors and recursive filtering combinations against several competitor methods. Trials involved seven scenarios that included real and simulated point cloud data. In this section we further analyze the contribution of each descriptor and filtering method to the odometry performance based on the normalized e^{RT}. Normalization is performed by dividing all e^{RT} metrics by the worst e^{RT} per filter or descriptor depending on the evaluation. This normalization is done for each scenario. In order to cross-compare the normalized e^{RT} between the different scenarios we also divide it with the ground truth distance of each scenario. This is important as it is well known that odometry errors build up with distance and therefore cross-comparing e^{RT} across all scenarios is only possible if it is normalized for the distance travelled. For better readability of the plots we use a logarithmic scale and in order to compensate for values that are less than one, we scale the normalized e^{RT} errors by multiplying them with a fixed constant. Additionally, in order to highlight the performance improvement only

when the descriptor can provide true matches, we discard the descriptor with the lowest odometry accuracy.

4.1 Feature description performance analysis

Figure 14 presents the average normalized e^{RT} metric of all recursive filters evaluated in this work per feature and per scenario *vs.* the total processing time required by the entire odometry architecture. This is important because for odometry applications the trajectory accuracy and the computational burden are equally important. In Figure 14 the color of the marker is related to the scenario, while the shape to the descriptor. Identifying the overall most appealing descriptor is not an easy task because the performance of each descriptor varies for each scenario depending on the characteristics of the trajectory, i.e. *Source – Target* range, frame-to-frame *Target* motion, and on the complexity of the *Target* structure. Figure 14 demonstrates that the average normalized e^{RT} for each descriptor is of the same order, with only minor differences. However, in terms of processing efficiency, the descriptors impose a different processing time that is governed by several parameters including the individual features of each scenario, i.e. level of sparsity, the robustness of each descriptor, the number of geometrically consistent feature matches and finally, the number of iterations for each filter. Overall, we conclude that HoD-S is the most appealing descriptor for the majority of the scenarios, i.e. *Real-FB*, *Sim-Orion*, *Sim-Helical* and *Sim-Bennu*, with FPFH being the most appealing for the *Real-Curved* and the *Sim-EoI* scenarios. These findings confirm that HoD and HoD-S are very robust to low density point clouds (Odysseas Kechagias-Stamatis & Aouf, 2016), while in parallel these also afford a low processing time.

Interestingly, for each scenario all descriptors attain errors of the same order, with the only major differentiation being the processing burden. This claim should not be confused with the individual

findings for each scenario, as in Section 3 we presented the performance of each feature descriptor and filtering method per scenario, while here we evaluate the average performance of each descriptor aver all filtering methods. From Figure 14 it is evident that scenarios exploiting real LIDAR data pose larger errors compared to the synthetic ones. This is important as it highlights that despite creating realistic synthetic scenarios as done in this work, simulating various noise sources, range dependent $\boldsymbol{P}_k$ resolution variation and viewing dependent $\boldsymbol{P}_k$ creation from a complete 3D Target model, cannot be as realistic as real data acquisition. However, the feature description hierarchy for the top performing descriptors is relatively stable confirming that validity of the results and that despite the disadvantages of synthetic vs. real data, exploiting synthetic data is still valuable.

Regarding computational efficiency, in most cases HoD-S is at least one order of magnitude faster to compute than the rest of the descriptors. On the contrary, RoPS imposes the highest processing burden. This is because the former descriptor neglects estimating a LRF/A, while the latter involves a computationally expensive LRF. This is also evident from the TriSI descriptor that exploits the same LRF estimation method as RoPS, and thus is also processing inefficient. Interestingly, despite FPFH and SHOT being implemented in C++/PCL and executed via a MEX wrapper, in most of the scenarios these are less processing efficient than HoD-S and HoD that are entirely implemented in MATLAB. It is worth noting that the *Sim-EoI* scenario imposes the highest processing burden among all scenarios. This is due to the large $\boldsymbol{P}_k$ cardinality, demonstrating that in such cases, down sampling $\boldsymbol{P}_k$ or exploiting a keypoint detection strategy rather encoding all $\boldsymbol{P}_k$ should be considered.

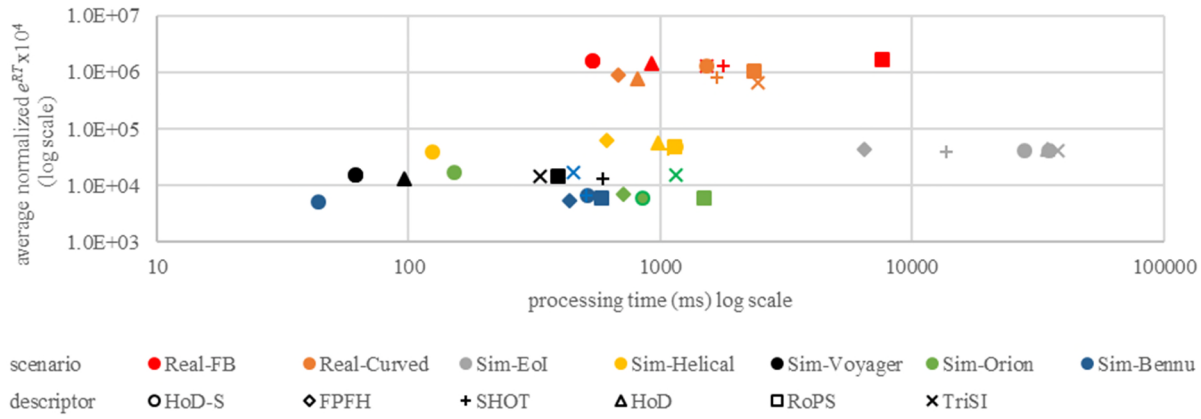


Figure 14: Normalized e^{RT} vs. processing time per feature descriptor and scenario. Marker shape indicates the descriptor type, while color the scenario. Descriptors attain similar average performance, over all filtering methods, with HoD-S in the majority of cases being the fastest descriptor.

Figure 15 presents the interplay between the $\boldsymbol{P}_k$ cardinality and the processing burden per descriptor. This figure highlights that FPFH is the least affected by the $\boldsymbol{P}_k$ cardinality, while HoD and HoD-S are the most affected ones. This is because the processing efficiency of HoD and HoD-S, due to neglecting an LRF/A estimation, is not balanced in high cardinality point cloud scenarios, i.e. *Sim-EoI* scenario, due to their large description radius. Our findings in Figure 15, confirm (Guo et al., 2016) in respect to the relative processing efficiency between SHOT, RoPS, TriSI and FPFH. The minor processing time fluctuations between various $\boldsymbol{P}_k$ cardinality values are because the total processing time considered in this plot is not only affected by the cardinality of $\boldsymbol{P}_k$ but also by the number of correspondences between $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$.

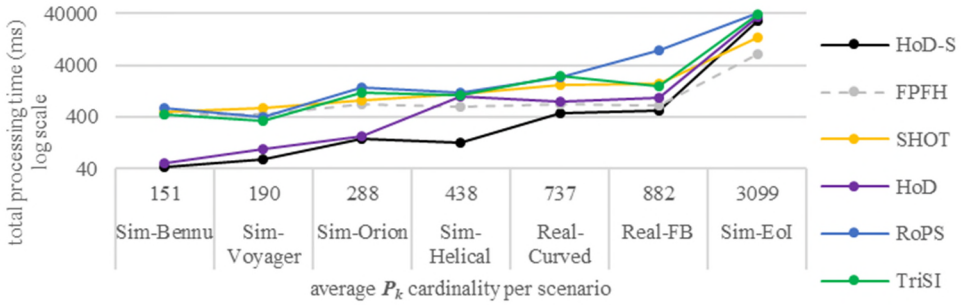


Figure 15: Average processing time per feature descriptor and scenario

4.2 Recursive filtering performance analysis

Next, we evaluate the performance of each filter per scenario, as an average over all feature descriptors evaluated. From Tables 3-9 we observe that the average processing time of each filter per scenario is almost independent from the recursive filtering method used, i.e. the fastest filter has only a 2% speed-up against the slowest one. Therefore, in Figure 16 we decouple the average normalized e^{RT} error from the total processing time per scenario and presents only the error. The results clearly demonstrate the superiority of the adaptive *Kalman* filtering as in five out of seven trials it affords a greater odometry accuracy over its competitor filters. Even for the *Real-FB* and *Sim-Bennu* scenarios where the adaptive *Kalman* filter is not the top performing one, it's e^{RT} error is only 2.5% and 8.3% greater compared to the top performing filter respectively. This is because opposing to the $\alpha\beta$ filter that is restricted by two states and uses static manually defined filter gains, the *Kalman* filter is not restricted and relies on a time-dependent automatically updated estimate of the state covariance, which is based on user defined covariance noise models. SDRE is a non-linear filter and since the *Target* point cloud between $\boldsymbol{P_k}$ and $\boldsymbol{P_{k+1}}$ is small, it can be better approximated with linear models. However, as already mentioned, these observations consider the average performance per filter and scenario, and do not prohibit the case where a filter attains a low average performance but a very appealing odometry accuracy if combined with a specific

descriptor. Similarly to Figure 14, Figure 16 also highlights the difference between using real and synthetic data. In fact, scenarios involving real data impose errors that are at least one order of magnitude greater compared to synthetic data. This is mainly due to the poor structure of the $\boldsymbol{P}_k$ acquired by the LIDAR sensor and due to the minor *Source* platform tumbling.

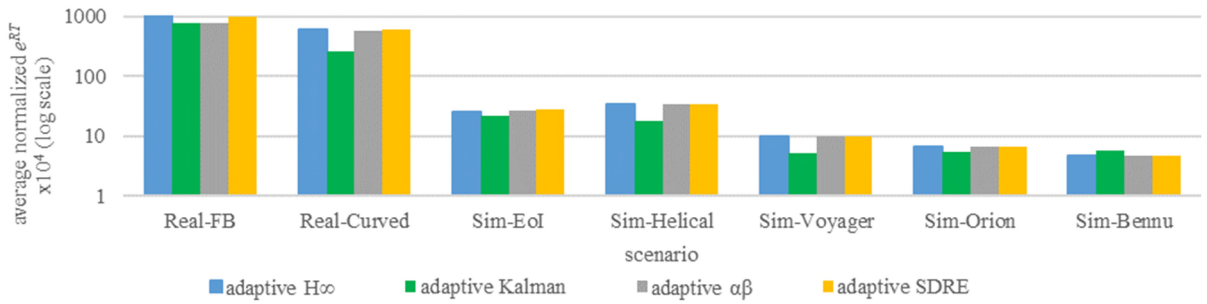


Figure 16: Average processing time per feature descriptor and scenario. Real data-based scenarios have present larger errors compared to the synthetic due to the lack of features on the $\boldsymbol{P}_k$ and the *Source* tumbling of the former ones.

In Figure 17 we further analyze the performance of each filter by presenting the e^{RT}, e^{T}_{avg} and e^{R}_{avg} errors per filter and scenario. From this figure it is also obvious that the data modality, i.e. real vs. synthetic, has a great impact on the odometry accuracy. As already presented in Figure 16, the e^{RT} error in the real scenarios is higher compared to the error in the simulated ones. Interestingly, in the real LIDAR data and in the *Sim-EoI* scenario, which has a high $\boldsymbol{P}_k$ cardinality, the e^{T}_{avg} is smaller than the e^{R}_{avg}, while in the remaining synthetic scenarios with a low $\boldsymbol{P}_k$ cardinality this is reversed. This effect is because of the frame-to-frame motion of the real-data scenarios that is less smooth compared to the synthetic ones and also due to the high $\boldsymbol{P}_k$ cardinality of the *Sim-EoI* scenario. Furthermore, given that the *Sim-EoI* and the *Sim-Voyager*, *Sim-Orion* and *Sim-Bennu* scenarios adopt the same trajectory, the difference between the e^{R}_{avg} and e^{T}_{avg} errors can be related to the $\boldsymbol{P}_k$ sparsity in combination to the amount of details/ distinctive features in each $\boldsymbol{P}_k$. It should be noted that the calculation of the error metrics presented in Figure 17 is based on eq. (27), (29)

and (31) respectively, with the former two equations being conceptually different compared to the third one. Therefore, the e^{R}_{avg} and e^{T}_{avg} errors can significantly differ from the e^{RT} error, e.g. the errors of the *Kalman* filter for the *Sim-EoI* scenario presented in Figure 17.

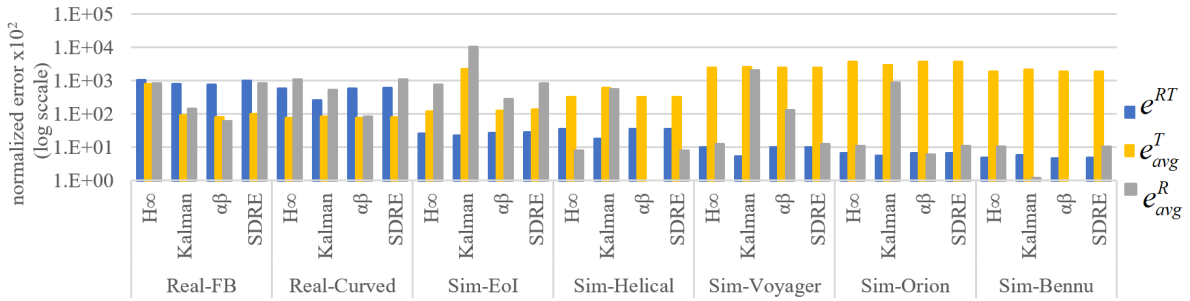


Figure 17: Odometry accuracy breakdown per recursive method and scenario

For completeness, we also investigate the importance of the recursive filtering module by substituting it in our odometry pipeline with the ICP point-to-point registration process, which is a commonly used registration method in the space navigation literature. Hence, in this trial we apply the ICP point-to-point on the correspondences produced by the GCC process, rather than on the entire $\boldsymbol{P_k}$ as done in the trials of Section 3. The performance attained with and without the recursive filtering module per scenario is presented in Table 3, where it is clear that the recursive filtering process has a great impact on the odometry accuracy. However, exploiting a recursive filtering scheme increases the total computational time.

Table 3: Odometry performance with and without the recursive filtering module

Scenario	top combination	e_{avg}^{T}		e_{avg}^{R}		e^{RT}		t (ms)	
		with filtering	without filtering	with filtering	without filtering	with filtering	without filtering	with filtering	without filtering
Real-FB	HOD-S/ αβ	0.10	11.58	0.99	-77.59	1.79	298.93	1753	1481
Real-Curved	HOD-S/ αβ	0.16	2.05	1.44	137.10	2.75	140.36	816	801
Sim-EoI	HOD-S/ αβ	0.70	37.88	2.91	-163.74	9.23	246.67	27954	26362
Sim-Helical	HOD-S/ αβ	1.61	10.36	0.01	18.56	12.53	45.06	112	66
Sim-Voyager	HoD-S/ SDRE	14.20	28.52	0.14	119.44	40.78	177.50	57	27
Sim-Orion	SHOT/ Kalman	5.32	49.83	0.01	91.69	24.25	176.40	858	714
Sim-Bennu	HOD-S/ αβ	1.16	147.22	0.01	164.89	10.65	365.75	41	27

4.3 Performance analysis of the top performing description – filtering combination

In Table 4 we summarize the most appealing module per scenario, i.e. descriptor, filtering method and the combination of the former two modules, along with the special features per scenario. From the results presented in Figure 14 and summarized in Table 4, it can be concluded that overall in terms of odometry accuracy and computational burden, HoD-S is the most appealing feature descriptor, with FPFH to follow. Considering the best performing filtering scheme, from Figure 16 we observe that the adaptive *Kalman* is the optimum choice, with the adaptive $\alpha\beta$ filter to follow. A summary of the top performing filtering method per scenario is presented in Table 4. Interestingly, the most appealing descriptors and recursive filtering methods per scenario, as concluded from the scenarios in Section 3, do not necessarily coincide with the most appealing combination of these two modules individually. This is because in the former analysis on the optimum descriptor per scenario, we considered the overall performance of all recursive filtering methods evaluated in this work.

Table 4: Top performing module per scenario

Scenario	Top descriptor	Top recursive filtering	Top combination	Scenario features
Real-FB	HoD-S	*αβ*	HoD-S / *αβ*	Low density – poor structure
Real-Curved	FPFH	*Kalman*	HoD-S / *αβ*	Low density – high curvature – poor structure
Sim-EoI	FPFH	*Kalman*	HoD-S / *αβ*	High density – high curvature
Sim-Helical	HoD-S	*Kalman*	HoD-S / *αβ*	Very low density – high curvature – large 3D translation
Sim-Voyager	HoD-S	*Kalman*	HoD-S / *SDRE*	Very low density – high curvature
Sim-Orion	HoD-S	*Kalman*	SHOT / *Kalman*	Very low density – High curvature – poor structure
Sim-Bennu	HoD-S	*αβ*	HoD-S / *αβ*	Very low density – High curvature

HoD-S is appealing because of its robustness to low and very low density point clouds along with their processing efficiency confirming the findings of (Odysseas Kechagias-Stamatis & Aouf, 2016). Despite that, in the *Sim-Orion* scenario SHOT combined with the adaptive *Kalman* filter present the most appealing combination. This is because the *Target* involved in this scenario has a poor structure comprising mostly of flat surfaces, indicating that for that type of scenarios involving a LRF in the description process is crucial.

We further analyze the performance of the top performing combinations per scenario, by comparing them against the corresponding top performing competitor solutions. Evaluation considers the odometry accuracy expressed via the normalized e^{RT} error metric and the computational burden imposed by each solution. From Figure 18 it is obvious that the proposed method is more accurate, while ICP based methods are generally faster to execute. It should be noted that the results of Figure 18 are normalized but not scaled, i.e. multiplied with a fixed constant. By combining the information presented in Figure 18 and the average $\boldsymbol{P_k}$ cardinality per scenario shown in Figure 9, the following conclusions can be made: First, for highly sparse point clouds, e.g. *Sim-Voyager* and *Sim-Bennu* scenarios, the proposed method is more accurate than the top performing competitor method evaluated in this work and is faster to execute. For sparse $\boldsymbol{P_k}$,

e.g. *Real-FB*, *Real-Curved*, *Sim-Helical* and *Sim-Orion*, identifying the most appealing solutions depends on the nature of $\boldsymbol{P_k}$, i.e. real vs. simulated data. Specifically, for the real LIDAR data case, the top performing ICP variant is one order of magnitude faster than the proposed method, with the e^{RT} error being of the same order.

From the trials presented in this paper, it is clear that the proposed odometry is more accurate compared to current space-oriented navigation techniques. One limitation of our method is that for point clouds exceeding 200 vertices, its computational burden is higher than the burden imposed by the ICP variants. However, in this paper we focus on cases simulating a low-resolution point cloud that is acquired by a space graded LIDAR sensor at greater distances or by a low-cost low-resolution space-graded LIDAR device. Thus, this limitation considers a broader usage of our architecture and not for the cases examined here. Regarding the computational burden, partially this is because our architecture is a blend of MATLAB and C++ and thus it is not optimized in terms processing efficiency. However, given that our odometry architecture comprises of several processes, i.e. feature description, matching, geometric consistency checks and recursive filtering, implementing it in C++ shall still be more processing costly compared to ICP, but with a smaller time difference. However, as already stated, this paper is focusing on the conceptual validity of the proposed method rather than to a readily available solution. An additional limitation of our method is its sensitivity to *Targets* with smooth surfaces because 3D local feature descriptors are prone to mismatches affecting the odometry accuracy. Despite that, in scenarios that involve *Targets* with flat surfaces, the proposed odometry solution still manages to attain lower odometry errors compared to typical ICP techniques.

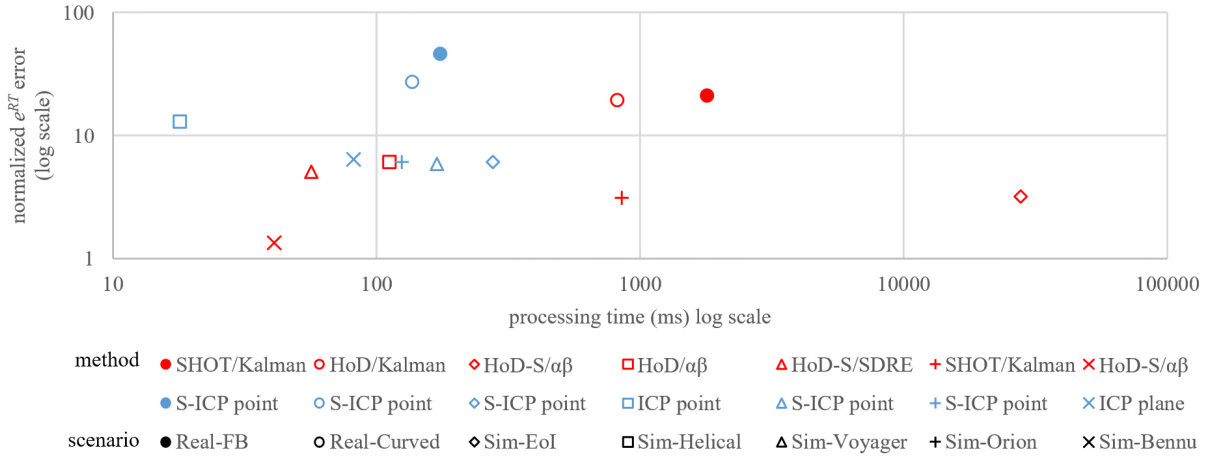


Figure 18: Normalized e^{RT} of the top performing proposed vs. competitor solution per scenario

4.4 GCC module performance analysis

We also investigate the influence of the GCC module by substituting it in our odometry pipeline with RANSAC, which is a commonly used in computer vision applications to define the inliers of two data sets. Table 5 compares the odometry performance of our original architecture against the one using RANSAC. We implement the latter with a false alarm rate of 0.1, an inlier ratio of 99% and performing at least 50,000 iterations. From Table 5, it is evident that GCC is up to one order of magnitude more accurate than RANSAC, while the normalized rotational accuracy is up to two orders of magnitude more accurate. This performance difference is more evident in the real scenarios highlighting that RANSAC is more prone to real LIDAR point clouds that suffer from minor noise and sensor tumbling during acquisition.

In terms of processing burden, we partially confirm (Yang et al., 2018), which states that RANSAC imposes up to two orders of magnitude additional computational time. In fact, our trials confirm the former statement but only for *Target* point clouds with a cardinality that is less than approximately 700 vertices. In fact, for larger point clouds we observe that GCC and RANSAC require the same order of execution time, with RANSAC though being faster to execute. This is

because for each iteration RANSAC exploits only a random sample out of the corresponding vertices and then applies the estimated model on the entire input data, while GCC involves in each iteration all the corresponding vertices as produced by the coarse matching process of eq. (5). Therefore, given that the inter-motion between $\boldsymbol{P_k}$ and $\boldsymbol{P_{k+1}}$ is small, the number of corresponding vertices produced by eq. (5) is directly related to the point cloud cardinality. In simple words, GCC is faster to execute because if $\boldsymbol{P_k}$ and $\boldsymbol{P_{k+1}}$ have a small cardinality, then eq. (5) produces less correspondences that need to be evaluated by the GCC module.

Table 5: Odometry performance with and without the GCC module

Scenario	top combination	normalized e^{T}_{avg}		normalized e^{R}_{avg}		normalized e^{RT}		t (ms)	
		GCC	RANSAC	GCC	RANSAC	GCC	RANSAC	GCC	RANSAC
Real-FB	HoD-S / $\alpha\beta$	0.013	0.543	0.133	11.690	0.241	12.421	1753	1442
Real-Curved	HoD-S / $\alpha\beta$	0.013	0.351	0.116	11.215	0.222	11.800	816	677
Sim-EoI	HoD-S / $\alpha\beta$	0.002	0.190	0.010	0.550	0.032	0.761	27954	9896
Sim-Helical	HoD-S / $\alpha\beta$	0.008	0.518	0.001	-0.553	0.061	1.776	112	3102
Sim-Voyager	HoD-S/ SDRE	0.018	0.067	0.001	0.226	0.052	0.300	57	3128
Sim-Orion	SHOT/ Kalman	0.007	0.058	0.001	-0.175	0.031	0.347	858	3857
Sim-Bennu	HOD-S/ αβ	0.001	0.071	0.001	-0.190	0.014	0.348	41	3409

4.5 Comparison against the Correspondence by Local and Global Voting method

We further challenge the performance of the proposed pipeline against the Correspondence by Local and Global Voting (CLGV) (Buch et al., 2014). This technique involves four steps, namely feature description, correspondence refinement via GCC and RANSAC, and finally Singular Value Decomposition (SVD) to estimate the rigid transformation between $\boldsymbol{P_k}$ and $\boldsymbol{P_{k+1}}$. Table 6 highlights that the proposed solution is more appealing compared to CLGV, attaining odometry accuracy that is one order of magnitude more accurate and more processing efficient. This is because the SVD suffers from ambiguity in the orientation of the singular vectors (Tomasi, 2016) affecting the

estimation of R^* (eq. (1)). Regarding computational effort, CLGV imposes a large processing burden mainly due to the two iterative processes involved, i.e. GCC and RANSAC.

Table 6: Proposed vs. Global / Local voting odometry

Scenario	top combination	normalized e_{avg}^{T}		normalized e_{avg}^{R}		normalized e^{RT}		t (ms)	
		proposed	CLGV	proposed	CLGV	proposed	CLGV	proposed	CLGV
Real-FB	HoD-S / *αβ*	0.013	0.554	0.133	14.447	0.241	15.260	1753	1382
Real-Curved	HoD-S / *αβ*	0.013	0.328	0.116	7.183	0.222	7.653	816	645
Sim-EoI	HoD-S / *αβ*	0.002	0.185	0.010	0.510	0.032	0.705	27954	39249
Sim-Helical	HoD-S / *αβ*	0.008	0.319	0.001	0.613	0.061	0.958	112	3016
Sim-Voyager	HoD-S/ SDRE	0.018	0.061	0.001	-0.140	0.052	0.378	57	3245
Sim-Orion	SHOT/ Kalman	0.007	0.103	0.001	-0.204	0.031	0.379	858	4221
Sim-Bennu	HOD-S/ αβ	0.001	0.087	0.001	-0.213	0.014	0.345	41	3482

4.6 Interplay between performance and number of iterations for the ICP variants

In Section 3, we set the number of maximum iterations per ICP method, i.e. ICP point 20, ICP plane 1000, ICP x84 20, while the iterations for both S-ICP variants are set to 20. To support this choice per ICP method, we evaluate each ICP variant by setting the maximum number of iterations to 20 and 1000. Evaluation considers the scenarios presented in Section 3 and the corresponding results are presented in Figure B1. For better readability, the best performance per normalized error is highlighted in red, if it is attained by setting to 20 iterations, and in blue if setting to 1000 iterations.

From the results obtained it is clear that the chosen number of iterations used in Section 3 is the optimum one for each ICP variant. However, from further analyzing the results of Figure B1, we observe that the ICP plane variant settles in up to 20 iterations, for both the real and the simulated data scenarios. Accordingly, the ICP plane variant requires more iterations to settle and thus setting to the threshold up to 1000 is mandatory. It is worth noting that for both variants, the normalized

translational, rotational and e^{RT} errors equally benefit from the number of iterations set. However, interestingly this is not the case for the ICP x84 variant, as the translational and e^{RT} errors benefit from a low number of iterations, while the rotational error from a large number. This behavior is present in both the real and the synthetic data scenarios. The S-ICP point variant, presents overall a higher odometry accuracy when the iterations are up to 20. However, this is only valid for the synthetic data scenarios, as real data scenarios require more iterations. Given that during the parameter setup we set the parameters of each method to be universal, i.e. selected for an overall optimum odometry accuracy on both real and synthetic data scenarios, we set iterations to 20. Finally, the S-ICP plane variant is quite stable requiring up to 20 iterations. The results obtained confirm our previous findings that real and simulated data require a different parameter setup.

5. Conclusion

LIDAR based odometry for space relative navigation is challenging due to the absence of background, the limited structure and the sparsity of the *Target* point cloud. As demonstrated, several ICP variants, the S4PCS, and the Correspondence Local/ Global voting method, which are currently widely used for point cloud registration and odometry applications, do not guarantee an accurate space odometry trajectory. Spurred by this, we suggest a robust architecture appropriate for space odometry that combines the concepts of 3D local feature description, geometric correspondence grouping for feature matching refinement and adaptive recursive filtering.

The accuracy of the proposed pipeline is tested on seven scenarios that include both real and synthetic point cloud data, on four space objects comprising of quite a few different satellites, a space capsule and an asteroid. In our trials we evaluate several current 3D local descriptor and

recursive filtering combinations and demonstrate that the proposed architecture is 50% more accurate compared to current solutions. Our trials highlighted that HoD-S combined with the adaptive $\alpha\beta$ filtering poses the most appealing combination for the majority of the scenarios evaluated, affording a high quality odometry performance with a low processing burden. Additional advantages of the proposed architecture over current LIDAR space odometry architectures are; being independent of an off-line training process and not requiring *a priori* knowledge of the *Target* platform.

Acknowledgements

This research was supported by the European Union as part of the H2020 project "Integrated 3D Sensors (I3DS)" under grant Nº 730118. The authors would like to thank Thales Alenia Space for providing the simulated data for the *Sim-EoI* scenario 3. The authors would also like to thank the anonymous reviewers for their constructive comments.

References

Aiger, D., Mitra, N. J., & Cohen-Or, D. (2008). 4-Points Congruent Sets for Robust Pairwise Surface Registration. *ACM Transactions on Graphics*, *27*(3), 1. https://doi.org/10.1145/1360612.1360684

Aldomà, A. (2011). render view. Retrieved March 24, 2015, from https://github.com/PointCloudLibrary/pcl/blob/master/apps/include/pcl/apps/render_views_tesselated_sphere.h

Aldoma, A., Tombari, F., Rusu, R. B., & Vincze, M. (2012). OUR-CVFH – Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in*

Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 7476 LNCS, pp. 113–122). https://doi.org/10.1007/978-3-642-32717-9_12

Alexandre, L. A. (2012). 3D Descriptors for Object and Category Recognition : a Comparative Evaluation. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, *34*(8), 1–6. https://doi.org/10.1109/TPAMI.2011.263

Aqel, M. O. A., Marhaban, M. H., Saripan, M. I., & Ismail, N. B. (2016). Review of visual odometry: types, approaches, challenges, and applications. *SpringerPlus*, *5*(1). https://doi.org/10.1186/s40064-016-3573-7

Arnold, W. F., & Laub, A. J. (1984). Generalized eigenproblem algorithms and software for algebraic Riccati equations. *Proceedings of the IEEE*, *72*(12), 1746–1754. https://doi.org/10.1109/PROC.1984.13083

ASC. (2019a). DragonEye 3D Flash LIDAR Space Camera™. Retrieved March 19, 2019, from http://www.advancedscientificconcepts.com/products/older-products/dragoneye.html

ASC. (2019b). GoldenEye 3D Flash LIDAR™ Space Camera. Retrieved March 14, 2019, from http://www.advancedscientificconcepts.com/products/portable.html

Besl, P. J., & McKay, N. D. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *14*(2), 239–256. https://doi.org/10.1109/34.121791

Birdal, T. (2015). ICP Registration using Efficient Variants and Multi-Resolution Scheme. Retrieved March 7, 2019, from https://uk.mathworks.com/matlabcentral/fileexchange/47152-icp-registration-using-

efficient-variants-and-multi-resolution-scheme?s_tid=prof_contriblnk

Bonnal, C., Ruault, J. M., & Desjean, M. C. (2013). Active debris removal: Recent progress and current trends. *Acta Astronautica*, *85*, 51–60. https://doi.org/10.1016/j.actaastro.2012.11.009

Bouaziz, S., Tagliasacchi, A., & Pauly, M. (2013). Sparse Iterative Closest Point. *Computer Graphics Forum*, *32*(5), 113–123. https://doi.org/10.1111/cgf.12178

Buch, A. G., Yang, Y., Kruger, N., & Petersen, H. G. (2014). In Search of Inliers: 3D Correspondence by Local and Global Voting. In *2014 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2075–2082). IEEE. https://doi.org/10.1109/CVPR.2014.266

Burgard, W., Stachniss, C., Grisetti, G., Steder, B., Kümmerle, R., Dornhege, C., … Tardós, J. D. (2009). A comparison of SLAM algorithms based on a graph of relations. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, 2089–2095. https://doi.org/10.1109/IROS.2009.5354691

Chen, H., & Bhanu, B. (2007). 3D free-form object recognition in range images using local surface patches. *Pattern Recognition Letters*, *28*(10), 1252–1262. https://doi.org/10.1016/j.patrec.2007.02.009

Cheng, A. F. (2002). Near Earth asteroid rendezvous: mission summary. *Asteroids III*, *1*, 351–366.

Cvišić, I., Ćesić, J., Marković, I., & Petrović, I. (2018). SOFT-SLAM: Computationally efficient stereo visual simultaneous localization and mapping for autonomous unmanned aerial vehicles. *Journal of Field Robotics*, *35*(4), 578–595. https://doi.org/10.1002/rob.21762

Deng, H., Birdal, T., & Ilic, S. (2018). PPFNet: Global Context Aware Local Features for Robust 3D Point Matching. In *2018 IEEE/CVF Conference on Computer Vision and Pattern*

Recognition (pp. 195–205). IEEE. https://doi.org/10.1109/CVPR.2018.00028

Deschaud, J.-E. (2018). IMLS-SLAM: Scan-to-Model Matching Based on 3D Data. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2480–2485). IEEE. https://doi.org/10.1109/ICRA.2018.8460653

Dietrich, A. B., & McMahon, J. W. (2018). Robust Orbit Determination with Flash Lidar Around Small Bodies. *Journal of Guidance, Control, and Dynamics*, *41*(10), 2163–2184. https://doi.org/10.2514/1.G003023

Dietrich, A., & McMahon, J. W. (2017). Orbit Determination Using Flash Lidar Around Small Bodies. *Journal of Guidance, Control, and Dynamics*, *40*(3), 650–665. https://doi.org/10.2514/1.G000615

Estébanez Camarena, M., Feetham, L. M., Scannapieco, A., & Aouf, N. (2018). FPGA-based multi-sensor relative navigation in space: Preliminary analysis in the framework of the I3DS H2020 project. In *69 th International Astronautical Congress (IAC),* (pp. 1–8). Bremen: Internation Astronautical Federation.

Fehse, W. (2003). The drivers for the approach strategy. *Automated Rendezvous and Docking of Spacecraft; Cambridge University Press: Cambridge, UK*, 124–126.

Fischler, M. a, & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381–395. https://doi.org/10.1145/358669.358692

Flores-Abad, A., Ma, O., Pham, K., & Ulrich, S. (2014). A review of space robotics technologies for on-orbit servicing. *Progress in Aerospace Sciences*, *68*, 1–26.

https://doi.org/10.1016/j.paerosci.2014.03.002

Frome, A., Huber, D., Kolluri, R., Bülow, T., & Malik, J. (2004). Recognizing Objects in Range Data Using Regional Point Descriptors. In *ECCV* (Vol. 3023, pp. 224–237). https://doi.org/10.1007/978-3-540-24672-5_18

Fusiello, A., Castellani, U., Ronchetti, L., & Murino, V. (2002). Model Acquisition by Registration of Multiple Acoustic Range Views. In A. Heyden, G. Sparr, M. Nielsen, & P. Johansen (Eds.), *Computer Vision --- ECCV 2002* (pp. 805–819). Berlin, Heidelberg: Springer Berlin Heidelberg.

Galante, J., Van Eepoel, J., Strube, M., Gill, N., Gonzalez, M., Hyslop, A., & Patrick, B. (2012). Pose Measurement Performance of the Argon Relative Navigation Sensor Suite in Simulated-Flight Conditions. In *AIAA Guidance, Navigation, and Control Conference* (pp. 1–26). Reston, Virigina: American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2012-4927

Gómez Martínez, H., Giorgi, G., & Eissfeller, B. (2017). Pose estimation and tracking of non-cooperative rocket bodies using Time-of-Flight cameras. *Acta Astronautica*, *139*(February), 165–175. https://doi.org/10.1016/j.actaastro.2017.07.002

Graeter, J., Wilczynski, A., & Lauer, M. (2018). LIMO: Lidar-Monocular Visual Odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 7872–7879). IEEE. https://doi.org/10.1109/IROS.2018.8594394

Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., & Kwok, N. M. (2016). A Comprehensive Performance Evaluation of 3D Local Feature Descriptors. *International Journal of Computer Vision*, *116*(1), 66–89. https://doi.org/10.1007/s11263-015-0824-y

Guo, Y., Bennamoun, M., Sohel, F., Lu, M., Wan, J., & Zhang, J. (2015). Performance Evaluation of 3D Local Feature Descriptors. In *Computer Vision -- ACCV 2014* (pp. 178–194). https://doi.org/10.1007/978-3-319-16808-1_13

Guo, Y., Sohel, F., Bennamoun, M., Lu, M., & Wan, J. (2013a). Rotational Projection Statistics for 3D Local Surface Description and Object Recognition. *International Journal of Computer Vision*, *105*(1), 63–86. Computer Vision and Pattern Recognition. https://doi.org/10.1007/s11263-013-0627-y

Guo, Y., Sohel, F., Bennamoun, M., Lu, M., & Wan, J. (2013b). TriSI : A Distinctive Local Surface Descriptor for 3D Modeling and Object Recognition. In *8th International Conference on Computer Graphics Theory and Applications*. Barcelona, Spain. https://doi.org/10.5220

Jaimez, M., & Gonzalez-Jimenez, J. (2015). Fast Visual Odometry for 3-D Range Sensors. *IEEE Transactions on Robotics*, *31*(4), 809–822. https://doi.org/10.1109/TRO.2015.2428512

Jena Optronik. (2019). RVS-3000. Retrieved March 19, 2019, from http://www.jena-optronik.de

Ji, K., Chen, H., Di, H., Gong, J., Xiong, G., Qi, J., & Yi, T. (2018). CPFG-SLAM:a Robust Simultaneous Localization and Mapping based on LIDAR in Off-Road Environment. *IEEE Intelligent Vehicles Symposium, Proceedings*, *2018–June*(Iv), 650–655. https://doi.org/10.1109/IVS.2018.8500599

Johnson, A. E., & Hebert, M. (1998). Efficient multiple model recognition in cluttered 3-D scenes. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 671–677). https://doi.org/10.1109/CVPR.1998.698676

Johnson, A. E., & Hebert, M. (1999). Using spin images for efficient object recognition in cluttered

3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *21*(5), 433–449. https://doi.org/10.1109/34.765655

Katz, S., Tal, A., & Basri, R. (2007). Direct visibility of point sets. *ACM Transactions on Graphics*, *26*(3), 24. https://doi.org/10.1145/1276377.1276407

Kechagias-stamatis, O., & Aouf, N. (2019). H∞ LIDAR Odometry for Spacecraft Relative Navigation. *IET Radar, Sonar & Navigation*, (1). https://doi.org/10.1049/iet-rsn.2018.5354

Kechagias-Stamatis, O., & Aouf, N. (2016). Histogram of distances for local surface description. In *2016 IEEE International Conference on Robotics and Automation (ICRA)* (Vol. 2016–June, pp. 2487–2493). Stockholm, Sweden: IEEE. https://doi.org/10.1109/ICRA.2016.7487402

Kechagias-Stamatis, O., & Aouf, N. (2017). Evaluating 3D local descriptors for future LIDAR missiles with automatic target recognition capabilities. *The Imaging Science Journal*, *65*(7), 428–437. https://doi.org/10.1080/13682199.2017.1361665

Kechagias-Stamatis, O., & Aouf, N. (2018). A New Passive 3-D Automatic Target Recognition Architecture for Aerial Platforms. *IEEE Transactions on Geoscience and Remote Sensing*, 1–10. https://doi.org/10.1109/TGRS.2018.2855065

Kechagias-Stamatis, O., Aouf, N., Gray, G., Chermak, L., Richardson, M., & Oudyi, F. (2018). Local feature based automatic target recognition for future 3D active homing seeker missiles. *Aerospace Science and Technology*, *73*, 309–317. https://doi.org/10.1016/j.ast.2017.12.011

Kechagias-Stamatis, O., Aouf, N., & Nam, D. (2017). 3D Automatic Target Recognition for UAV Platforms. In *2017 Sensor Signal Processing for Defence Conference (SSPD)* (pp. 1–5).

London, UK: IEEE. https://doi.org/10.1109/SSPD.2017.8233223

Khoury, M., Zhou, Q.-Y., & Koltun, V. (2017). Learning Compact Geometric Features. In *2017 IEEE International Conference on Computer Vision (ICCV)* (pp. 153–161). IEEE. https://doi.org/10.1109/ICCV.2017.26

Kim, D.-H., & Kim, J.-H. (2016). Effective Background Model-Based RGB-D Dense Visual Odometry in a Dynamic Environment. *IEEE Transactions on Robotics*, *32*(6), 1565–1573. https://doi.org/10.1109/TRO.2016.2609395

Kornfeld, R. P., Bunker, R. L., Cucullu, G. C., Essmiller, J. C., Hadaegh, F. Y., Christian Liebe, C., … Wong, E. C. (2003a). New millennium ST6 autonomous rendezvous experiment (ARX). In *2003 IEEE Aerospace Conference Proceedings (Cat. No.03TH8652)* (Vol. 1, pp. 1–380). IEEE. https://doi.org/10.1109/AERO.2003.1235067

Kornfeld, R. P., Bunker, R. L., Cucullu, G. C., Essmiller, J. C., Hadaegh, F. Y., Christian Liebe, C., … Wong, E. C. (2003b). New millennium ST6 autonomous rendezvous experiment (ARX). In *2003 IEEE Aerospace Conference Proceedings (Cat. No.03TH8652)* (Vol. 1, pp. 1–380). IEEE. https://doi.org/10.1109/AERO.2003.1235067

Krämer, M. S., Hardt, S., & Kuhnert, K. (2018). Image Features in Space - Evaluation of Feature Algorithms for Motion Estimation in Space Scenarios. In *Proceedings of the 7th International Conference on Pattern Recognition Applications and Methods* (pp. 300–308). Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications. https://doi.org/10.5220/0006555303000308

Langlois, P.-A. (2018). ICP Sparse. Retrieved from https://github.com/palanglois/icpSparse

Lei Yunqi, Lai Haibin, & Jiang Xutuan. (2010). 3D face recognition by SURF operator based on depth image. In *2010 3rd International Conference on Computer Science and Information Technology* (Vol. 9, pp. 240–244). IEEE. https://doi.org/10.1109/ICCSIT.2010.5563632

Li, L., Lian, J., Guo, L., & Wang, R. (2013). Visual odometry for planetary exploration rovers in sandy terrains. *International Journal of Advanced Robotic Systems*, *10*, 1–7. https://doi.org/10.5772/56342

Liebe, C. C., Abramovici, A., Bartman, R. K., Bunker, R. L., Chapsky, J., Cheng-Chih Chu, … Wright, M. (2003). Laser radar for spacecraft guidance applications. In *2003 IEEE Aerospace Conference Proceedings (Cat. No.03TH8652)* (Vol. 6, p. 6_2647-6_2662). IEEE. https://doi.org/10.1109/AERO.2003.1235190

Liu, C., & Hu, W. (2014). Relative pose estimation for cylinder-shaped spacecrafts using single image. *IEEE Transactions on Aerospace and Electronic Systems*, *50*(4), 3036–3056. https://doi.org/10.1109/TAES.2014.120757

Liu, L., Zhao, G., & Bo, Y. (2016). Point cloud based relative pose estimation of a satellite in close range. *Sensors (Switzerland)*, *16*(6). https://doi.org/10.3390/s16060824

Lu, F., & Milios, E. (1997). Globally Consistent Range Scan Alignment for Environment Mapping. *Autonomous Robots*, *4*(4), 333–349. https://doi.org/10.1023/A:1008854305733

Maimone, M., Cheng, Y., & Matthies, L. (2007). Two years of Visual Odometry on the Mars Exploration Rovers. *Journal of Field Robotics*, *24*(3), 169–186. https://doi.org/10.1002/rob.20184

Mellado, N. (2017). Super4PCS.

Mellado, N., Aiger, D., & Mitra, N. J. (2014). Super 4PCS Fast Global Pointcloud Registration via Smart Indexing. *Computer Graphics Forum*, *33*(5), 205–215. https://doi.org/10.1111/cgf.12446

Mian, A. S., Bennamoun, M., & Owens, R. (2006). Three-Dimensional Model-Based Object Recognition and Segmentation in Cluttered Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(10), 1584–1601. https://doi.org/10.1109/TPAMI.2006.213

Mikolajczyk, K., & Schmid, C. (2005). Performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(10), 1615–1630. https://doi.org/10.1109/TPAMI.2005.188

Mouats, T., Aouf, N., Chermak, L., & Richardson, M. A. (2015). Thermal Stereo Odometry for UAVs. *IEEE Sensors Journal*, *15*(11), 6335–6347. https://doi.org/10.1109/JSEN.2015.2456337

Mouats, T., Aouf, N., Sappa, A. D., Aguilera, C., & Toledo, R. (2015). Multispectral Stereo Odometry. *IEEE Transactions on Intelligent Transportation Systems*, *16*(3), 1210–1224. https://doi.org/10.1109/TITS.2014.2354731

Muja, M., & Lowe, D. G. (2009). Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *International Conference on Computer Vision Theory and Applications (VISAPP '09)* (pp. 1–10). Lisboa, Portugal. https://doi.org/10.1.1.160.1721

Naasz, B., & Moreau, M. (2012). Autonomous RPOD technology challenges for the coming decade. *Advances in the Astronautical Sciences*, *144*, 403–425.

NASA. (2019). NASA 3D models. Retrieved March 7, 2019, from

https://nasa3d.arc.nasa.gov/models

Nemra, A., & Aouf, N. (2009). Robust Airborne 3D Visual Simultaneous Localization and Mapping with Observability and Consistency Analysis. *Journal of Intelligent and Robotic Systems*, *55*(4–5), 345–376. https://doi.org/10.1007/s10846-008-9306-6

Neuhaus, F., Koß, T., Kohnen, R., & Paulus, D. (2019). MC2SLAM: Real-Time Inertial Lidar Odometry Using Two-Scan Motion Compensation. In T. Brox, A. Bruhn, & M. Fritz (Eds.), *Pattern Recognition* (pp. 60–72). Cham: Springer International Publishing.

Opromolla, R., Di Fraia, M. Z., Fasano, G., Rufino, G., & Grassi, M. (2017). Laboratory test of pose determination algorithms for uncooperative spacecraft. In *4th IEEE International Workshop on Metrology for AeroSpace, MetroAeroSpace 2017 - Proceedings* (pp. 169–174). Padua, Italy. https://doi.org/10.1109/MetroAeroSpace.2017.7999558

Opromolla, R., Fasano, G., Rufino, G., & Grassi, M. (2014). Spaceborne LIDAR-based system for pose determination of uncooperative targets. In *2014 IEEE International Workshop on Metrology for Aerospace, MetroAeroSpace 2014 - Proceedings* (pp. 265–270). Benevento, Italy. https://doi.org/10.1109/MetroAeroSpace.2014.6865932

Opromolla, R., Fasano, G., Rufino, G., & Grassi, M. (2015a). A model-based 3D template matching technique for pose acquisition of an uncooperative space object. *Sensors (Switzerland)*, *15*(3), 6360–6382. https://doi.org/10.3390/s150306360

Opromolla, R., Fasano, G., Rufino, G., & Grassi, M. (2015b). Uncooperative pose estimation with a LIDAR-based system. *Acta Astronautica*, *110*, 287–297. https://doi.org/10.1016/j.actaastro.2014.11.003

Opromolla, R., Fasano, G., Rufino, G., & Grassi, M. (2017). A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations. *Progress in Aerospace Sciences*, *93*(July), 53–72. https://doi.org/10.1016/j.paerosci.2017.07.001

Optitrack. (2018). Retrieved May 22, 2018, from https://optitrack.com/

Penoyer, R. (1993). The Alpha-Beta Filter. *C User's Journal*, *11*(7), 73–86.

Rhodes, A., Kim, E., Christian, J. A., & Evans, T. (2016). LIDAR-based Relative Navigation of Non-Cooperative Objects Using Point Cloud Descriptors. In *AIAA/AAS Astrodynamics Specialist Conference*. Reston, Virginia: American Institute of Aeronautics and Astronautics. https://doi.org/10.2514/6.2016-5517

Rhodes, A. P., Christian, J. A., & Evans, T. (2017). A Concise Guide to Feature Histograms with Applications to LIDAR-Based Spacecraft Relative Navigation. *Journal of the Astronautical Sciences*, *64*(4), 414–445. https://doi.org/10.1007/s40295-016-0108-y

Rusu, R. B., Blodow, N., & Beetz, M. (2009). Fast Point Feature Histograms (FPFH) for 3D registration. In *2009 IEEE International Conference on Robotics and Automation* (pp. 3212–3217). Kobe, Japan: IEEE. https://doi.org/10.1109/ROBOT.2009.5152473

Salti, S., Tombari, F., & Di Stefano, L. (2014). SHOT: Unique signatures of histograms for surface and texture description. *Computer Vision and Image Understanding*, *125*, 251–264. https://doi.org/10.1016/j.cviu.2014.04.011

Sell, J. L., Rhodes, A., Woods, J. O., Christian, J. A., & Evans, T. (2014). Pose Performance of LIDAR-Based Navigation for Satellite Servicing. *AIAA/AAS Astrodynamics Specialist*

Conference, (August), 1–14. https://doi.org/10.2514/6.2014-4360

Simon, D. (2000). From here to infinity. *Embedded Systems Programming*, *14*(11), 20–32.

Simon, D. (2001). Kalman Filtering. *Embedded Systems Programming*, 72–79.

Song, J. (2017). Sliding window filter based unknown object pose estimation. In *2017 IEEE International Conference on Image Processing (ICIP)* (pp. 2642–2646). IEEE. https://doi.org/10.1109/ICIP.2017.8296761

Tomasi, C. (2016). *Orthogonal matrices and the singular value decomposition*. University of Duke.

Tombari, F., Salti, S., & Di Stefano, L. (2010). Unique shape context for 3d data description. In *Proceedings of the ACM workshop on 3D object retrieval - 3DOR '10* (p. 57). New York, New York, USA: ACM Press. https://doi.org/10.1145/1877808.1877821

Tykkala, T., & Comport, A. I. (2011). A dense structure model for image based stereo SLAM. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on* (pp. 1758–1763). Shangai, China. https://doi.org/10.1109/ICRA.2011.5979805

Volpe, R., Palmerini, G., & Sabatini, M. (2017). Monocular and Lidar Based Determination of Shape , Relative Attitude and Position of a Non-Cooperative , Unknown Satellite. In *International Astronautical Congress (IAC 2017)* (pp. 25–29). Adelaide, Australia.

Woods, J. O., & Christian, J. A. (2016). Lidar-based relative navigation with respect to non-cooperative objects. *Acta Astronautica*, *126*, 298–311. https://doi.org/10.1016/j.actaastro.2016.05.007

Yang Cheng, Maimone, M., & Matthies, L. (2006). Visual Odometry on the Mars Exploration

Rovers. In *2005 IEEE International Conference on Systems, Man and Cybernetics* (Vol. 1, pp. 903–910). Waikoloa, HI, USA. https://doi.org/10.1109/ICSMC.2005.1571261

Yang, J., Cao, Z., & Zhang, Q. (2016). A fast and robust local descriptor for 3D point cloud registration. *Information Sciences*, *346–347*(August), 163–179. https://doi.org/10.1016/j.ins.2016.01.095

Yang, J., Xian, K., Xiao, Y., & Cao, Z. (2018). Performance evaluation of 3D correspondence grouping algorithms. In *Proceedings - 2017 International Conference on 3D Vision, 3DV 2017* (pp. 467–476). https://doi.org/10.1109/3DV.2017.00060

Yang, J., Zhang, Q., & Cao, Z. (2017). Multi-attribute statistics histograms for accurate and robust pairwise registration of range images. *Neurocomputing*, *251*, 54–67. https://doi.org/10.1016/j.neucom.2017.04.015

Yılmaz, O., Aouf, N., Majewski, L., Sanchez-Gestido, M., & Ortega, G. (2017). Using infrared based relative navigation for active debris removal. In *10th International ESA Conference on Guidance, Navigation & Control Systems* (pp. 1–16). Salzburg, Austria.

Zhang, J., & Singh, S. (2015a). LOAM: Lidar Odometry and Mapping in Real- time. *IEEE Transactions on Robotics*, *32*(July), 141–148. https://doi.org/10.15607/RSS.2014.X.007

Zhang, J., & Singh, S. (2015b). Visual-lidar odometry and mapping: low-drift, robust, and fast. In *2015 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 2174–2181). IEEE. https://doi.org/10.1109/ICRA.2015.7139486

Zhao, B., Le, X., & Xi, J. (2019). A novel SDASS descriptor for fully encoding the information of a 3D local surface. *Information Sciences*, *483*, 363–382.

https://doi.org/10.1016/j.ins.2019.01.045

Zhou, Y., Li, H., & Kneip, L. (2019). Canny-VO: Visual Odometry With RGB-D Cameras Based on Geometric 3-D–2-D Edge Alignment. *IEEE Transactions on Robotics*, *35*(1), 184–199. https://doi.org/10.1109/TRO.2018.2875382

Zou, Y., Wang, X., Zhang, T., Liang, B., Song, J., & Liu, H. (2018). BRoPH: An efficient and compact binary descriptor for 3D point clouds. *Pattern Recognition*, *76*, 522–536. https://doi.org/10.1016/j.patcog.2017.11.029

Appendix A

For better readability this appendix presents the detailed performance metrics and odometry plots per method and scenario

Table A1: Performance metrics for the *Real-FB* scenario

Descriptor	drift (m)	Terror (%)	$e^{T}_{\max}$	e^{T}_{avg}	max Error (m)			Average error (m)			e^{T}_{avg}	$e^{R}_{\max}$	e^{RT}	t (ms)
					X	Y	Z	X	Y	Z				
adaptive *H*∞ recursive filtering														
HoD-S	0.16	2.11	***0.17***	***0.09***	***0.05***	0.13	0.09	***0.05***	***0.03***	0.05	9.12	13.58	9.91	***560***
FPFH	3.56	48.02	3.56	2.32	0.91	3.19	1.28	0.62	1.91	0.76	9.11	13.57	13.38	687
SHOT	0.58	7.77	0.59	0.23	0.56	0.15	0.09	0.12	0.09	0.10	8.61	12.77	9.89	1794
HoD	***0.15***	***2.00***	0.36	0.17	0.34	0.07	***0.01***	0.12	0.04	***0.03***	9.12	13.57	10.05	950
RoPS	0.84	11.39	0.85	0.25	0.19	0.68	0.48	***0.05***	0.15	0.13	9.13	13.57	10.26	2376
TriSI	0.26	3.56	3.13	1.69	3.13	***0.01***	***0.01***	1.39	***0.03***	0.04	3.35	***4.55***	***6.32***	1536
adaptive *Kalman* recursive filtering														
HoD-S	0.84	11.29	1.29	0.79	1.12	0.08	0.64	0.50	***0.03***	0.41	2.34	4.90	4.48	***557***
FPFH	2.41	32.49	2.82	1.78	0.70	2.37	1.36	0.56	1.24	0.73	2.48	4.55	6.04	687
SHOT	0.58	7.84	0.59	0.23	0.57	0.16	0.06	***0.12***	0.10	0.11	0.27	0.51	***1.56***	1794
HoD	***0.26***	***3.52***	***0.35***	***0.21***	***0.30***	0.06	0.16	***0.12***	0.04	0.12	0.91	1.83	2.09	946
RoPS	1.64	22.09	1.64	0.81	0.64	0.65	1.37	0.29	0.13	0.53	2.60	4.66	4.45	2375
TriSI	***0.26***	3.57	3.13	1.69	3.13	***0.01***	***0.01***	1.39	***0.03***	***0.04***	***0.05***	***0.07***	3.02	1536
adaptive *αβ* recursive filtering														
HoD-S	0.16	2.20	***0.18***	***0.10***	***0.11***	0.07	0.13	0.05	0.04	0.05	0.99	1.34	1.79	***518***
FPFH	3.58	48.21	3.58	2.33	0.87	3.23	1.26	0.60	1.93	0.77	0.60	0.81	4.87	679
SHOT	0.56	7.54	0.57	0.23	0.54	0.17	0.08	0.12	0.10	0.11	0.26	0.35	***1.57***	1784
HoD	***0.15***	***2.00***	0.38	0.18	0.36	0.07	0.01	0.13	0.04	***0.03***	0.99	1.36	1.94	903
RoPS	0.86	11.53	0.86	0.26	0.15	0.70	0.48	***0.04***	0.16	0.13	0.66	0.96	1.75	23620
TriSI	0.26	3.53	3.13	1.69	3.13	***0.01***	***0.01***	1.40	***0.03***	0.04	***0.06***	***0.08***	3.03	1535
adaptive *SDRE* recursive filtering														
HoD-S	0.85	11.41	0.86	0.40	***0.45***	0.54	0.49	0.18	0.23	0.16	9.12	13.58	10.85	***521***
FPFH	4.19	56.46	4.19	2.54	1.31	3.61	1.69	0.82	2.12	0.63	9.11	13.57	13.74	684
SHOT	1.13	15.28	1.14	0.46	0.92	0.52	0.44	***0.17***	0.28	0.13	8.61	12.77	10.24	1752
HoD	0.85	11.43	***0.85***	***0.38***	0.48	0.53	0.47	0.14	0.20	0.18	9.12	13.57	10.71	907
RoPS	1.23	16.63	1.24	0.53	0.50	1.04	0.42	0.22	0.35	0.17	9.13	13.57	11.07	2376
TriSI	***0.27***	***3.57***	3.10	1.67	3.10	***0.02***	***0.03***	1.37	***0.02***	***0.04***	***3.35***	***4.55***	***6.28***	1536
competitor schemes														
ICP point	0.44	5.90	1.41	0.64	1.30	0.50	0.22	0.43	0.15	0.13	84.49	133.01	86.32	5
ICP plane	1.90	25.61	1.91	0.97	1.79	0.07	0.67	0.61	0.40	0.35	-171.1	-111.87	191.71	9
ICP x84	8.75	117.97	11.36	9.57	8.69	4.94	4.56	6.65	3.78	4.24	-165.4	-121.96	206.69	40
S4PCS	1.26	16.98	1.91	1.09	0.10	1.89	0.19	0.16	0.73	0.45	77.01	141.53	79.53	1225
S-ICP point	0.13	1.70	3.52	1.97	3.51	0.20	0.11	1.64	0.08	0.07	0.01	0.01	3.42	145.76
S-ICP plane	0.13	1.70	3.52	1.97	3.51	0.20	0.11	1.64	0.08	0.07	0.01	0.01	3.42	150.16

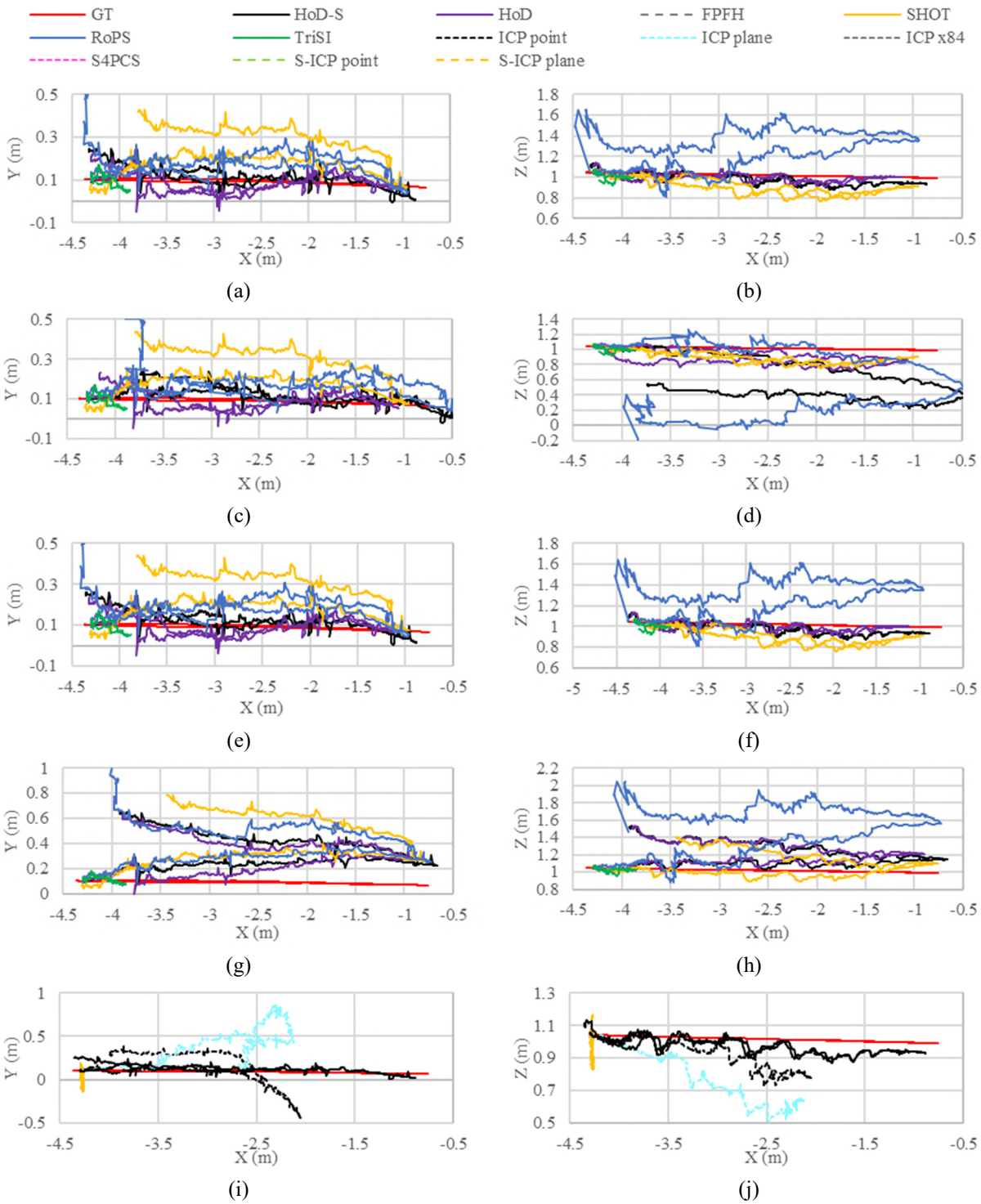


Figure A1: Forward – Backward *Real-FB* odometry plots of the 3D descriptors combined with the adaptive filter (a)-(b) $H\infty$, (c)-(d) *Kalman,* (e)-(f) $\alpha\beta$, (g)-(h) *SDRE*, and (i)-(j) the competitor methods along with the top performing method HoD-S/ $\alpha\beta$, which is the black trajectory closest to the GT (due to large errors and for better readability we crop RoPS from (a),(b),(c),(e),(g) , omit FPFH from (a) – (h) and ICP x84 and S4PCS from (i) –(j)). TriSI due to the sparse point clouds fails to provide an adequate number of geometrically consistent feature matches, forcing the recursive filtering process not to iterate properly and thus R^* mostly preserves its initialization value.

Table A2: Performance metrics for the *Real-Curved* scenario

Descriptor	drift (m)	Terror (%)	e^{T}_{max}	e^{T}_{avg}	max Error (m)			Average error (m)			e^{T}_{avg}	e^{R}_{max}	e^{RT}	t (ms)
					X	Y	Z	X	Y	Z				
adaptive *H*∞ recursive filtering														
HoD-S	0.27	2.15	***0.28***	***0.16***	0.20	***0.04***	0.20	***0.08***	0.07	0.08	12.22	17.95	13.51	***481***
FPFH	1.19	9.60	1.60	1.01	0.29	0.62	1.43	0.22	0.53	0.71	11.99	17.90	14.60	690
SHOT	0.72	5.86	0.93	0.71	0.80	0.45	0.11	0.57	0.34	0.04	11.05	16.72	13.55	1698
HoD	***0.06***	***0.51***	0.47	0.23	0.42	0.22	***0.03***	0.16	0.10	0.05	12.00	17.96	13.42	817
RoPS	0.91	7.35	0.91	0.37	***0.17***	0.05	0.89	0.16	***0.06***	0.26	12.27	18.18	14.08	2373
TriSI	0.51	4.12	3.45	1.93	3.43	0.37	***0.03***	1.50	0.49	***0.03***	***4.27***	***6.10***	***7.83***	2452
adaptive *Kalman* recursive filtering														
HoD-S	0.40	3.23	0.53	0.31	***0.29***	***0.05***	0.30	0.25	0.09	0.09	14.85	22.70	16.66	***480***
FPFH	1.20	9.68	1.60	1.02	0.30	0.62	1.43	0.22	0.54	0.72	1.92	3.62	4.53	690
SHOT	0.71	5.78	0.93	0.71	0.80	0.45	0.10	0.57	0.33	***0.04***	0.61	1.03	3.10	1697
HoD	***0.06***	***0.50***	***0.48***	***0.24***	0.42	0.22	***0.03***	***0.16***	0.11	0.05	0.98	1.90	***2.41***	816
RoPS	0.36	2.90	1.01	0.65	0.42	0.39	0.56	0.18	***0.05***	0.51	13.57	20.16	15.35	2372
TriSI	0.51	4.11	3.45	1.93	3.43	0.36	***0.03***	1.50	0.49	0.03	***0.21***	***0.25***	3.77	2452
adaptive *αβ* recursive filtering														
HoD-S	0.25	2.03	***0.27***	***0.16***	***0.17***	***0.05***	0.20	***0.09***	0.08	0.08	1.44	1.81	2.75	***464***
FPFH	1.21	9.75	1.60	1.02	0.30	0.62	1.43	0.23	0.54	0.72	0.57	0.85	3.21	683
SHOT	0.72	5.84	0.94	0.72	0.81	0.45	0.10	0.58	0.33	***0.04***	0.43	0.60	2.93	1691
HoD	***0.05***	***0.41***	0.49	0.24	0.43	0.22	***0.03***	0.17	0.11	0.05	1.21	1.47	***2.65***	809
RoPS	0.91	7.35	0.91	0.38	0.19	0.06	0.89	0.18	***0.06***	0.25	1.15	1.59	2.99	2357
TriSI	0.51	4.11	3.45	1.93	3.43	0.36	0.03	1.50	0.49	0.03	***0.14***	***0.16***	3.70	2451
adaptive *SDRE* recursive filtering														
HoD-S	0.74	5.99	0.77	0.41	0.49	***0.34***	0.49	0.16	0.22	0.22	12.22	17.95	14.14	***466***
FPFH	1.55	12.50	1.56	1.02	0.24	0.68	1.37	0.17	0.68	0.57	11.99	17.90	14.54	685
SHOT	***0.46***	3.74	0.79	0.56	0.68	***0.34***	0.22	0.44	0.21	0.13	11.05	16.72	13.40	1675
HoD	0.56	4.54	***0.66***	***0.39***	0.43	0.40	0.29	0.19	0.25	0.14	12.00	17.96	13.92	812
RoPS	1.24	10.05	1.24	0.50	***0.12***	0.35	1.19	***0.06***	***0.13***	0.37	12.27	18.18	13.96	2360
TriSI	***0.46***	***3.72***	3.43	1.91	3.41	0.35	***0.05***	1.48	0.48	***0.04***	***4.27***	***6.10***	***7.84***	2453
competitor schemes														
ICP point	2.70	21.81	2.78	1.68	1.68	2.07	0.53	0.66	1.14	0.16	104	-179.13	107.90	6
ICP plane	1.90	25.61	1.91	0.97	1.79	0.07	0.67	0.61	0.40	0.35	-171	-111.87	191.71	9
ICP x84	7.08	57.23	7.46	4.24	6.30	3.66	1.59	2.48	2.07	1.17	135	-142.43	140.29	53
S4PCS	4.60	37.18	4.60	2.89	3.52	2.53	1.54	1.47	1.73	0.81	141	-124.33	146.51	1219
S-ICP point	0.13	1.70	3.52	1.97	3.51	0.20	0.11	1.64	0.08	0.07	0.01	0.01	3.42	137
S-ICP plane	0.13	1.70	3.52	1.97	3.51	0.20	0.11	1.64	0.08	0.07	0.01	0.01	3.42	145

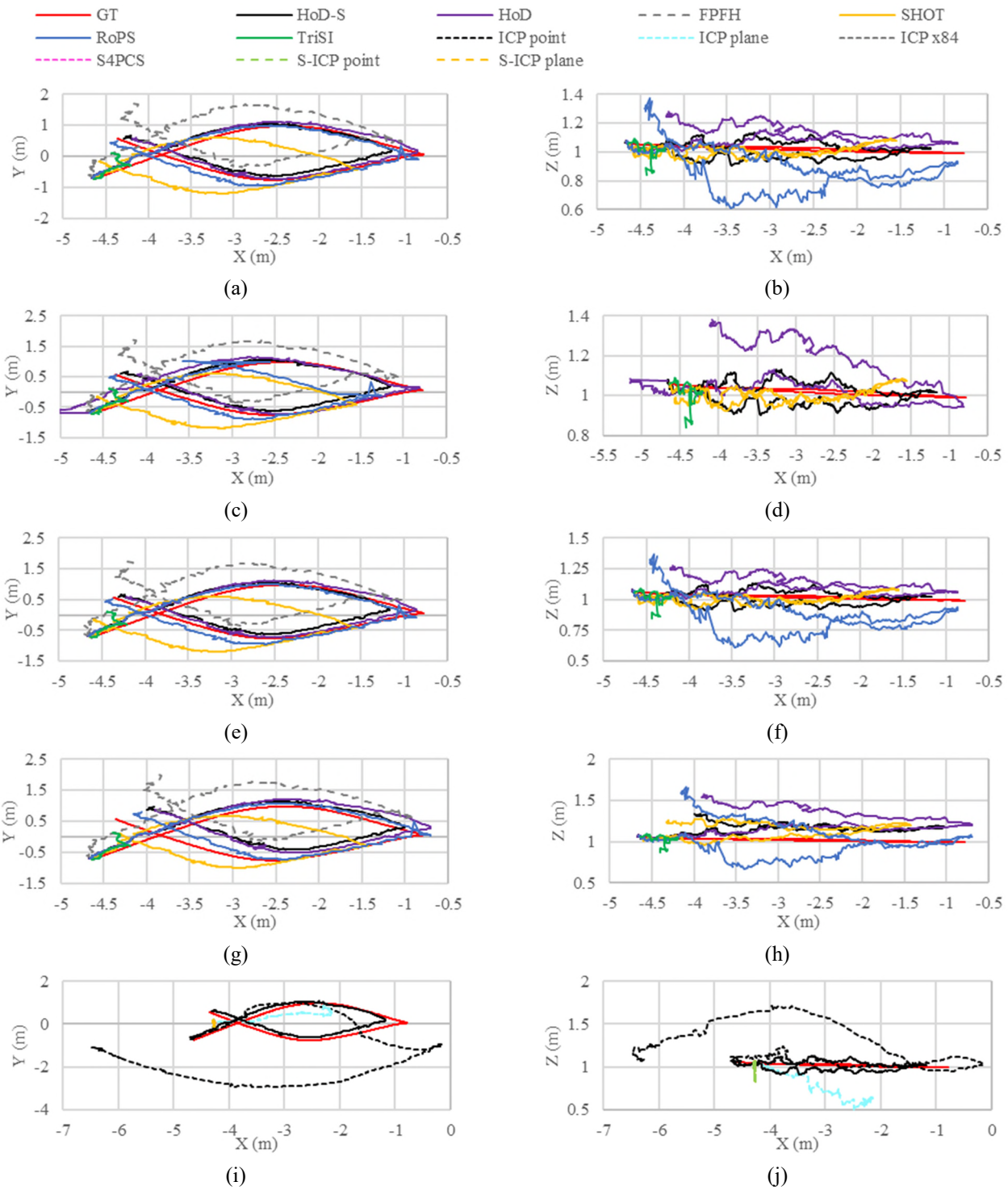


Figure A2: *Real-Curved* scenario odometry plots of the 3D descriptors combined with the adaptive filter (a)-(b) $H\infty$, (c)-(d) *Kalman,* (e)-(f) $\alpha\beta$, (g)-(h) *SDRE*, and (i)-(j) competitor methods along with the top performing proposed method HoD-S/ $\alpha\beta$, which is the black trajectory closest to the GT (due to large errors and for better readability we omit FPFH from (b), (d), (f), (h), RoPS from (d), ICP x84 from (i)-(j)). TriSI due to the sparse point clouds fails to provide an adequate number of geometrically consistent feature matches, forcing the recursive filtering process not to iterate properly and thus R^* mostly preserves its initialization value.

Table A3: Performance metrics for the *Sim-EoI* scenario

Descriptor	drift (m)	Terror (%)	$e^{T}_{\max}$	e^{T}_{avg}	max Error (m)			Average error (m)			e^{T}_{avg}	$e^{R}_{\max}$	e^{RT}	t (ms)
					X	Y	Z	X	Y	Z				
adaptive *H*∞ recursive filtering														
HoD-S	***1.16***	***0.40***	***1.19***	***0.69***	0.92	0.53	0.50	***0.48***	0.21	0.20	8.81	10.88	15.86	28157
FPFH	2.56	0.88	2.59	1.49	2.40	0.94	***0.18***	1.24	0.36	0.22	7.56	10.71	17.18	***6525***
SHOT	1.93	0.66	1.94	1.30	1.87	0.46	0.19	1.13	***0.18***	***0.09***	***2.89***	***3.95***	***10.56***	13701
HoD	***1.16***	***0.40***	1.28	0.81	***0.91***	0.72	0.55	***0.48***	0.39	0.28	8.83	10.97	17.47	34832
RoPS	2.37	0.81	3.16	1.87	2.82	0.77	1.18	1.56	0.27	0.39	7.91	10.33	18.45	40647
TriSI	1.90	0.65	1.95	1.08	1.68	***0.43***	0.27	0.84	0.29	***0.09***	8.98	11.16	16.80	38306
adaptive *Kalman* recursive filtering														
HoD-S	28.14	9.68	38.48	25.49	2.28	29.11	25.07	9.63	13.67	14.27	127.37	97.55	178.64	28144
FPFH	***14.83***	***5.10***	***14.98***	***11.69***	13.40	***3.27***	***4.59***	9.64	***2.16***	***3.42***	109.41	154.76	136.54	***6518***
SHOT	18.85	6.48	37.37	25.98	10.37	24.73	25.63	10.63	11.36	16.25	105.31	164.46	148.07	13870
HoD	15.88	5.46	29.03	21.60	8.80	11.59	25.11	***4.24***	12.64	14.14	79.02	117.81	125.87	34817
RoPS	20.36	7.00	39.97	28.72	8.13	29.34	23.98	10.17	13.88	18.76	156.53	200.72	202.61	40637
TriSI	17.64	6.07	32.87	21.86	***1.49***	20.78	25.18	6.91	6.92	16.38	***50.72***	***66.26***	***90.25***	38296
adaptive *αβ* recursive filtering														
HoD-S	1.24	0.43	***1.26***	***0.70***	1.02	0.44	0.24	***0.55***	***0.18***	***0.09***	2.91	3.98	***9.23***	27954
FPFH	2.61	0.90	2.63	1.49	2.47	0.89	0.11	1.26	0.34	0.10	2.77	3.88	11.59	***6424***
SHOT	1.93	0.66	1.94	1.30	1.87	0.46	***0.19***	1.13	***0.18***	***0.09***	2.89	3.95	10.56	13701
HoD	***1.22***	***0.42***	1.27	0.81	***1.00***	0.63	0.30	***0.55***	0.38	0.15	2.89	3.92	10.80	34610
RoPS	2.56	0.88	3.25	1.96	2.88	0.76	1.30	1.60	0.27	0.51	2.89	4.02	13.90	40502
TriSI	2.04	0.70	2.08	1.16	2.04	***0.35***	***0.19***	0.90	0.30	0.13	***2.73***	***3.66***	11.10	38160
adaptive *SDRE* recursive filtering														
HoD-S	1.44	0.50	1.46	***0.86***	1.17	***0.27***	0.76	0.63	***0.13***	0.34	8.81	10.88	***16.56***	27990
FPFH	2.79	0.96	2.81	1.65	2.70	0.64	0.48	1.39	0.24	0.35	***7.56***	10.71	17.45	***6425***
SHOT	2.21	0.76	2.21	1.49	1.88	0.73	0.57	1.23	0.32	0.32	7.96	***10.29***	18.06	13702
HoD	***1.40***	***0.48***	***1.49***	0.94	***1.16***	0.46	0.80	***0.62***	0.25	0.43	8.83	10.97	17.72	34725
RoPS	2.63	0.91	3.21	2.02	2.98	0.62	1.02	1.70	0.35	0.30	7.91	10.33	18.65	40502
TriSI	2.28	0.78	2.31	1.31	2.16	0.60	***0.39***	0.97	0.44	***0.18***	8.98	11.16	18.17	38161
competitor schemes														
ICP point	50.32	17.31	62.20	39.09	29.20	1.53	54.76	12.62	3.77	29.92	-164.3	-42.15	249.55	30.50
ICP plane	42.71	14.69	57.96	36.50	28.71	0.08	49.43	10.99	0.45	29.10	-167.5	-30.44	235.93	31.38
ICP x84	2.79	0.96	70.47	45.93	16.94	39.69	52.85	11.13	27.50	21.97	-165.6	-22.33	257.89	49.36
S4PCS	151.14	51.99	151.14	103.35	46.02	79.36	114.71	43.40	45.31	60.85	-155.6	32.60	342.80	1147
S-ICP point	0.04	0.01	20.08	11.44	0.37	0.05	20.06	0.24	0.04	9.05	0.01	0.01	17.49	276.68
S-ICP plane	0.04	0.01	20.08	11.44	0.37	0.05	20.06	0.24	0.04	9.05	0.01	0.01	17.49	327.62

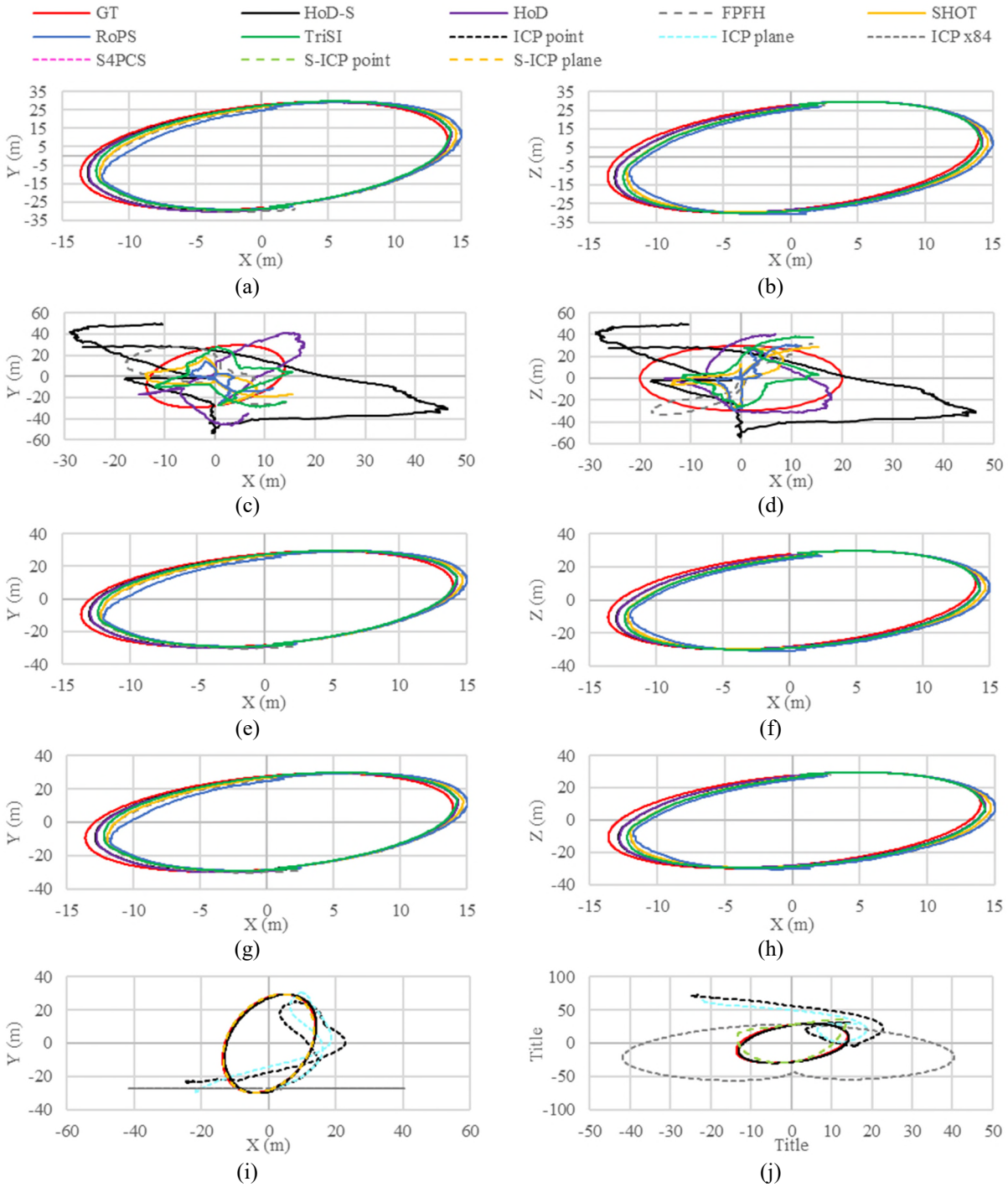


Figure A3: *Sim-EoI* scenario 3 odometry plots of the 3D descriptors combined with the adaptive filter (a)-(b) *H∞*, (c)-(d) *Kalman,* (e)-(f) *αβ*, (g)-(h) *SDRE*, and (i)-(j) competitor methods along with the top performing proposed method HoD-S/ αβ (due to large errors and for better readability we omit from (i)-(j) S4PCS). The *Kalman* filter clearly fails to provide a valid accuracy independently of the descriptor that it is combined. This is due to the Kalman parameters that are sensitive to the $\boldsymbol{P}_k$ cardinality. Competitor methods except from both S-ICP variant fail to attain accurate odometry as the $\boldsymbol{P}_k$ cardinality is below the operating threshold of ICP. Most of the proposed methods present a large error in the relatively sharp turning positions due to the combined *Source-Target* distance presenting less distinctive *Target* features.

Table A4: Performance metrics for the *Sim-Helical* scenario

Descriptor	drift (m)	Terror (%)	$e^{T}_{\max}$	e^{T}_{avg}	max Error (m)			Average error (m)			e^{T}_{avg}	$e^{R}_{\max}$	e^{RT}	t (ms)
					X	Y	Z	X	Y	Z				
adaptive *H*∞ recursive filtering														
HoD-S	***1.85***	***0.90***	***2.09***	***1.61***	1.79	0.42	***1.00***	1.28	0.17	***0.74***	0.08	0.16	***12.55***	***137***
FPFH	2.22	1.08	2.90	2.07	1.77	***0.07***	2.30	***1.04***	0.22	1.65	0.08	0.16	15.81	620
SHOT	5.89	2.86	5.92	2.17	3.63	4.33	1.64	1.67	0.37	***0.74***	0.08	0.15	13.69	1094
HoD	6.27	3.04	6.27	3.29	5.25	1.50	3.08	2.03	1.27	1.04	0.08	0.14	17.55	997
RoPS	2.16	1.05	2.45	1.83	***1.37***	0.33	1.74	1.20	***0.14***	1.28	0.08	0.16	14.88	1164
TriSI	12.74	6.18	12.74	8.26	7.40	7.24	7.24	2.69	4.86	2.82	***0.07***	***0.13***	26.49	1042
adaptive *Kalman* recursive filtering														
HoD-S	***2.03***	***0.98***	***2.47***	***1.59***	***1.79***	***0.12***	1.70	***1.22***	0.23	***0.77***	***0.01***	***0.01***	***13.13***	***134***
FPFH	32.38	15.71	33.74	14.92	10.05	5.48	31.74	2.63	1.12	8.52	32.61	-179.99	58.38	619
SHOT	5.87	2.85	5.90	2.16	3.58	4.38	***1.57***	1.64	0.37	0.79	***0.01***	***0.01***	13.71	1092
HoD	6.37	3.09	6.37	3.29	5.21	1.55	3.31	2.01	1.26	1.10	***0.01***	***0.01***	17.70	995
RoPS	9.36	4.54	10.03	6.12	6.94	0.14	7.24	3.70	***0.19***	3.36	0.20	0.24	23.89	1161
TriSI	12.84	6.23	12.84	8.28	7.42	7.20	7.37	2.70	4.85	2.86	***0.01***	***0.01***	26.52	1041
adaptive *αβ* recursive filtering														
HoD-S	***1.86***	***0.90***	***2.09***	***1.61***	1.79	0.39	***1.01***	1.28	0.17	0.75	***0.01***	***0.01***	***12.53***	***112***
FPFH	2.27	1.10	2.91	2.08	1.77	***0.10***	2.30	***1.04***	0.24	1.66	***0.01***	***0.01***	15.87	607
SHOT	5.92	2.87	5.96	2.17	3.64	4.38	1.61	1.67	0.37	***0.74***	***0.01***	***0.01***	13.61	1074
HoD	6.28	3.05	6.28	3.28	5.26	1.55	3.06	2.03	1.25	1.03	***0.01***	***0.01***	17.45	979
RoPS	2.20	1.07	2.49	1.85	***1.37***	0.35	1.76	1.20	***0.16***	1.29	***0.01***	***0.01***	14.97	1140
TriSI	12.73	6.18	12.73	8.25	7.41	7.20	7.23	2.70	4.85	2.81	***0.01***	***0.01***	26.43	1024
adaptive *SDRE* recursive filtering														
HoD-S	***1.81***	***0.88***	***2.08***	***1.60***	1.74	0.47	***1.04***	1.24	0.19	0.77	0.08	0.16	***12.64***	***116***
FPFH	2.29	1.11	2.91	2.08	1.73	***0.03***	2.34	***1.01***	0.19	1.68	0.08	0.16	15.75	609
SHOT	5.78	2.81	5.82	2.14	3.57	4.27	1.58	1.64	0.38	***0.75***	0.08	0.15	13.69	1077
HoD	6.18	3.00	6.18	3.27	5.19	1.44	3.02	2.00	1.29	1.03	0.08	0.14	17.46	981
RoPS	2.20	1.07	2.48	1.83	***1.15***	0.09	2.18	1.16	***0.13***	1.32	0.08	0.16	14.82	1143
TriSI	12.68	6.15	12.68	8.25	7.35	7.29	7.19	2.66	4.87	2.80	***0.07***	***0.13***	26.40	1027
competitor schemes														
ICP point	2.92	1.42	3.15	1.96	2.07	0.44	2.34	1.39	0.26	1.15	11.71	20.83	26.85	18
ICP plane	22.70	11.02	27.08	11.36	22.61	10.36	4.70	5.99	3.35	2.06	58.66	-96.87	84.62	89
ICP x84	16615	8063	17277	9210	12253	8668	8333	1905.	2878	3333	-132	-15.79	8346	25
S4PCS	147.65	71.65	155.65	84.13	46.40	146.59	24.19	15.94	64.69	10.21	109	-169	207.98	1081
S-ICP point	47.90	23.25	47.90	39.05	28.09	19.00	31.66	19.15	12.80	23.86	0.01	0.01	50.04	107
S-ICP plane	49.32	23.93	49.32	40.37	29.40	19.40	32.86	20.21	12.96	24.82	0.01	0.01	51.00	105

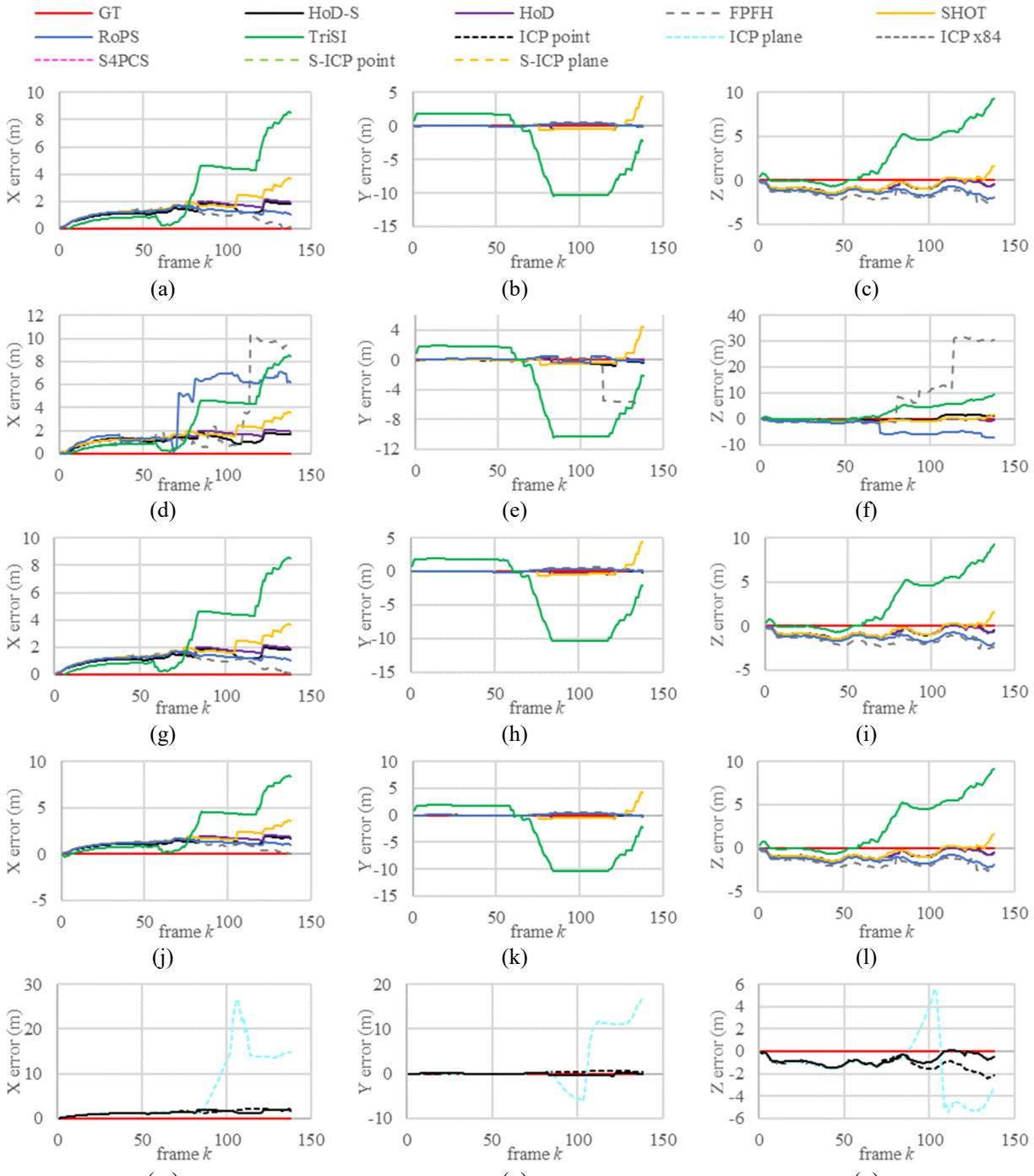


Figure A4: *Sim- Helical* odometry error plots of the 3D descriptors combined with the adaptive filter (a)-(c) $H\infty$, (d)-(f) *Kalman,* (g)-(i) $\alpha\beta$, (j)-(l) *SDRE*, and (m)-(o) competitor methods along with the top performing proposed method HoD-S/ $\alpha\beta$ (due to large errors and for better readability we omit ICP x84, S-ICP point , S-ICP plane and S4PCS from (m)-(o)). The large frame-to-frame motion combined with the highly sparse $\boldsymbol{P}_k$ prevent TriSI from providing an accurate odometry. Despite the sparse nature of $\boldsymbol{P}_k$ ICP point presents an odometry that is more accurate to the rest of the competitor methods.

Table A5: Performance metrics for the *Sim-Voyager* scenario

Descriptor	drift (m)	Terror (%)	$e^{T}_{\max}$	e^{T}_{avg}	max Error (m)			Average error (m)			$e^{R}_{\max}$	e^{T}_{avg}	e^{RT}	t (ms)
					X	Y	Z	X	Y	Z				
adaptive *H*∞ recursive filtering														
HoD-S	***16.59***	***2.10***	28.76	***14.27***	10.68	18.79	18.97	5.12	***7.50***	7.54	0.14	0.27	***40.91***	***66***
FPFH	104	13.24	105	62.86	28.27	50.14	88.19	11.47	29.59	47.60	0.13	0.26	97.43	438
SHOT	33.35	4.23	38.02	18.05	***9.55***	36.17	6.80	***4.84***	12.11	***2.63***	0.13	0.25	42.38	599
HoD	24.31	3.08	***27.71***	15.36	19.58	***18.44***	***6.63***	9.25	9.04	3.18	0.13	0.26	41.68	103
RoPS	21.66	2.75	31.10	16.00	12.20	19.09	14.14	5.68	10.79	5.79	0.14	0.27	43.15	398
TriSI	22.87	2.90	39.56	22.20	12.54	29.45	23.24	7.13	13.09	12.63	***0.08***	***0.16***	46.39	336
adaptive *Kalman* recursive filtering														
HoD-S	***10.34***	***1.31***	55.07	22.22	52.30	***12.06***	***2.33***	13.55	6.65	4.64	95.19	-162.41	140.12	***66***
FPFH	93.16	11.82	93.21	59.57	32.86	35.56	79.65	13.70	24.50	46.74	0.10	0.20	94.85	438
SHOT	34.55	4.38	39.12	18.30	***8.69***	37.61	6.37	***4.43***	12.13	2.46	0.18	0.40	45.20	598
HoD	25.04	3.18	***28.35***	***15.74***	20.29	18.81	6.20	9.67	9.18	***3.03***	***0.01***	***0.02***	***41.79***	102
RoPS	21.99	2.79	28.41	17.71	19.10	12.94	15.49	8.05	***5.87***	11.90	24.21	92.37	71.60	397
TriSI	22.50	2.85	38.94	22.09	12.59	29.27	22.39	7.14	13.07	12.61	1.19	2.98	47.58	335
adaptive *αβ* recursive filtering														
HoD-S	***15.77***	***2.00***	28.03	***13.83***	10.64	18.03	18.63	5.11	***7.05***	7.35	1.88	3.58	41.68	***57***
FPFH	104	13.24	105	62.86	28.27	50.14	88.19	11.47	29.59	47.60	***0.13***	***0.26***	97.43	438
SHOT	33.36	4.23	38.03	17.89	***9.82***	36.00	7.32	***5.00***	11.89	***2.81***	1.53	3.09	***38.19***	595
HoD	24.29	3.08	***27.49***	15.18	20.09	***17.70***	***6.21***	9.44	8.67	3.04	2.23	4.51	43.13	91
RoPS	22.08	2.80	30.72	16.08	12.71	18.64	14.88	5.98	10.57	6.18	1.03	1.97	44.68	390
TriSI	22.93	2.91	39.64	22.22	12.55	29.36	23.50	7.11	13.07	12.67	0.80	1.50	47.06	331
adaptive *SDRE* recursive filtering														
HoD-S	***16.41***	***2.08***	28.67	***14.20***	10.77	18.70	18.88	5.12	***7.45***	7.49	0.14	0.27	***40.78***	***57***
FPFH	104	13.23	105	62.83	28.16	50.26	88.08	11.42	29.65	47.54	0.13	0.26	97.38	435
SHOT	33.21	4.21	37.87	17.98	***9.44***	36.06	6.69	***4.79***	12.06	***2.61***	0.13	0.25	41.23	595
HoD	24.14	3.06	***27.52***	15.26	19.47	***18.33***	***6.52***	9.19	8.99	3.13	0.13	0.26	41.52	91
RoPS	21.46	2.72	30.98	15.91	12.09	18.98	14.03	5.63	10.74	5.74	0.14	0.27	42.99	391
TriSI	22.85	2.90	39.50	22.18	12.60	29.39	23.18	7.16	13.08	12.62	***0.08***	***0.16***	46.42	332
competitor schemes														
ICP point	9.52	1.21	21.03	11.22	4.93	19.40	6.43	4.48	6.37	2.91	19.22	34.10	55.27	15
ICP plane	11.15	1.41	26.18	13.30	3.73	23.66	10.27	3.92	8.38	4.26	41.48	75.56	80.69	16
ICP x84	80.60	10.22	80.60	61.73	53.26	58.43	15.68	43.74	24.99	9.27	147.65	-160	234.32	17
S4PCS	39.98	5.07	50.32	38.59	26.07	30.43	30.39	20.14	18.90	18.91	1.65	1.65	52.48	136
S-ICP point	28.94	3.67	38.45	29.50	21.24	14.80	28.40	14.82	12.27	16.18	0.01	0.01	46.33	171
S-ICP plane	28.94	3.67	38.45	29.50	21.24	14.80	28.40	14.82	12.27	16.18	0.01	0.01	46.33	177

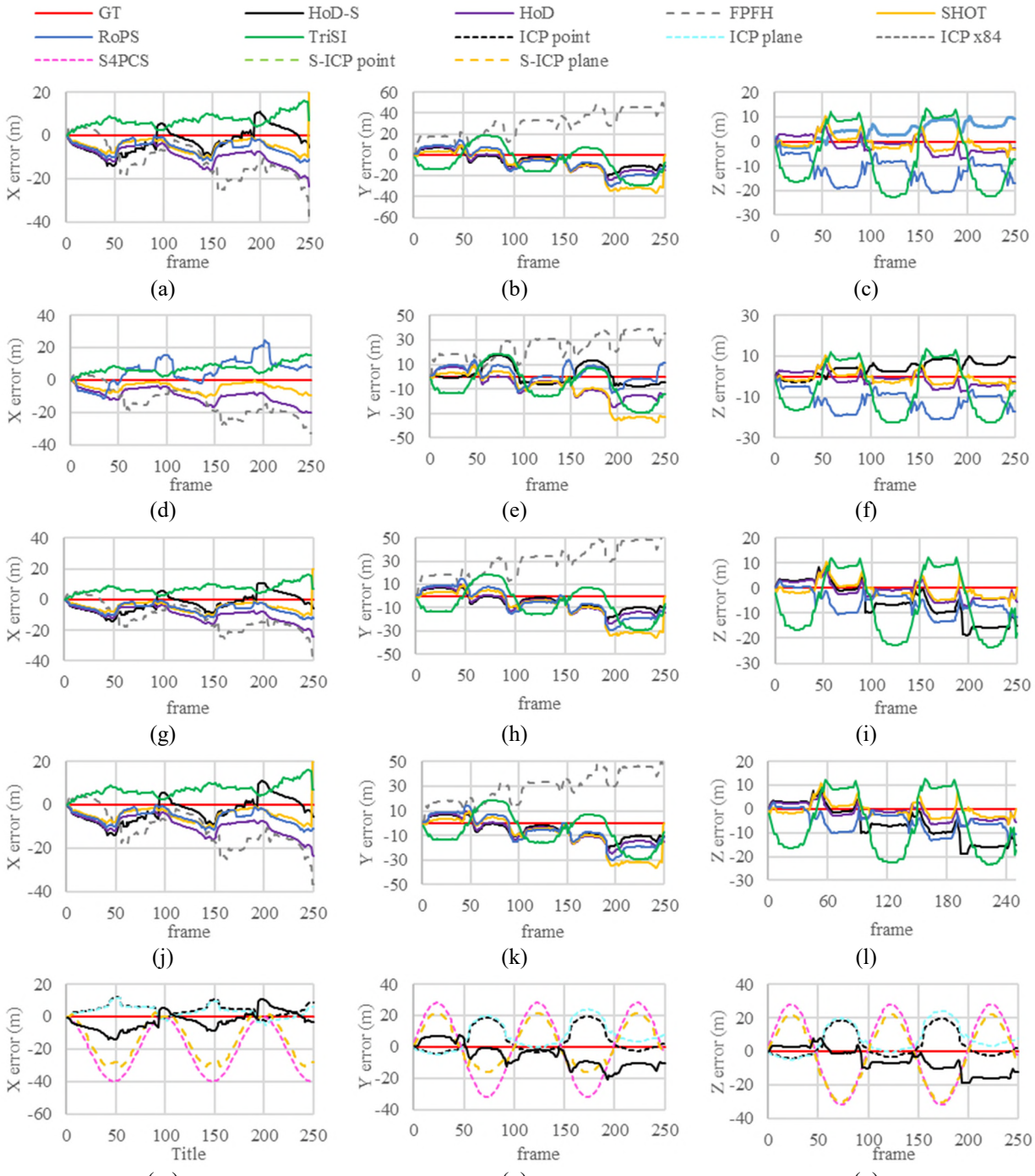


Figure A5: *Sim-Voyager* scenario 5 odometry error plots of the 3D descriptors combined with the adaptive filter (a)-(c) *H*∞, (d)-(f) *Kalman,* (g)-(i) *αβ*, (j)-(l) *SDRE*, and (m)-(o) competitor methods and top performing proposed method HoD-S/ SDRE (due to large errors and for better readability we omit FPFH from Z error plots, HoD-S from (d) and ICP x84 from (m)-(o))

Table A6: Performance metrics for the *Sim-Orion* scenario

Descriptor	drift (m)	Terror (%)	e^{T}_{max}	e^{T}_{avg}	max Error (m)			Average error (m)			e^{R}_{max}	e^{T}_{avg}	e^{RT}	t (ms)
					X	Y	Z	X	Y	Z				
adaptive *H*∞ recursive filtering														
HoD-S	109	13.84	111	64.28	7.72	4.21	109	***3.61***	3.89	54.45	0.14	0.27	70.76	***156***
FPFH	12.22	1.55	14.74	8.72	8.50	***3.27***	9.86	5.06	***1.97***	5.12	0.14	0.27	30.19	720
SHOT	9.43	1.20	11.94	5.91	7.39	7.08	6.16	3.08	2.59	2.74	0.14	0.27	26.93	859
HoD	118	14.97	138	78.47	79.15	81.10	78.99	48.37	33.68	32.72	***0.03***	***0.06***	101	169
RoPS	***7.30***	***0.93***	***11.33***	***6.70***	***7.21***	6.38	***5.56***	5.13	2.03	***2.27***	0.14	0.27	***26.46***	1514
TriSI	77.47	9.83	85.93	50.77	65.86	39.21	38.37	38.10	16.16	16.04	0.06	0.12	67.97	1159
adaptive *Kalman* recursive filtering														
HoD-S	29.37	3.73	37.06	25.17	19.83	6.74	28.52	11.48	5.03	18.05	50.37	-159	94.00	***156***
FPFH	11.88	1.51	14.59	8.70	7.58	***3.41***	10.38	4.63	2.02	5.34	***0.01***	***0.01***	29.85	719
SHOT	***8.46***	***1.07***	***10.76***	***5.32***	5.72	6.80	***6.07***	2.44	2.29	***2.65***	***0.01***	***0.01***	***24.25***	858
HoD	118	14.98	138	78.50	79.24	81.09	78.99	48.42	33.68	32.72	***0.01***	***0.01***	101	169
RoPS	13.40	1.70	16.59	6.44	***0.89***	3.53	16.19	***2.14***	***1.90***	4.33	0.10	0.22	24.69	1513
TriSI	77.59	9.84	86.05	50.82	66.00	39.19	38.42	38.15	16.16	16.05	***0.01***	***0.01***	67.95	1158
adaptive *αβ* recursive filtering														
HoD-S	109	13.91	111	64.50	7.66	***4.07***	110	***3.59***	3.90	54.62	0.05	0.10	70.84	***146***
FPFH	12.29	1.56	14.77	8.75	8.55	3.35	9.85	5.09	***2.00***	5.12	0.07	0.13	30.28	711
SHOT	9.55	1.21	11.98	***5.96***	7.39	7.09	6.21	3.10	2.63	2.79	0.11	0.22	26.99	845
HoD	118	14.97	138	78.50	79.19	81.09	79.04	48.39	33.68	32.74	***0.01***	***0.03***	101	166
RoPS	***7.24***	***0.92***	***11.40***	6.72	***7.23***	6.39	***5.61***	5.15	2.01	***2.28***	0.09	0.19	***26.46***	1503
TriSI	77.54	9.84	86.01	50.80	65.92	39.23	38.41	38.12	16.17	16.05	0.03	0.05	67.95	1154
adaptive *SDRE* recursive filtering														
HoD-S	109	13.83	111	64.22	7.83	4.09	109	3.65	3.90	54.39	0.14	0.27	70.74	***147***
FPFH	12.12	1.54	14.59	8.66	8.39	***3.38***	9.74	5.00	2.02	5.08	0.14	0.27	30.24	712
SHOT	9.53	1.21	11.95	5.94	7.36	7.10	6.18	***3.05***	2.64	2.79	0.14	0.27	26.60	846
HoD	118	14.97	138	78.46	79.18	81.07	78.97	48.39	33.67	32.71	***0.03***	***0.06***	101	168
RoPS	***7.11***	***0.90***	***11.33***	***6.65***	***7.19***	6.40	***5.58***	5.07	***2.01***	***2.27***	0.14	0.27	***26.40***	1504
TriSI	77.50	9.83	85.93	50.78	65.91	39.16	38.32	38.12	16.15	16.03	0.06	0.12	68.01	1155
competitor schemes														
ICP point	13.90	1.76	16.75	9.38	6.71	11.50	8.08	4.63	5.09	3.41	-130	35.91	262.48	13
ICP plane	11.54	1.46	12.59	8.64	8.10	3.02	8.79	4.62	1.80	4.91	-123	65.04	267.53	50
ICP x84	88.92	11.28	127.39	66.46	89.92	1.62	90.21	26.43	21.73	37.32	-163	26.55	277.57	24
S4PCS	54.65	6.93	65.76	43.33	39.43	37.30	37.11	27.31	19.49	19.47	0.54	1.07	60.24	1641
S-ICP point	29.98	3.80	46.69	32.92	20.57	28.57	30.31	15.04	17.15	16.66	0.01	0.01	48.12	125
S-ICP plane	29.98	3.80	46.69	32.92	20.57	28.57	30.31	15.04	17.15	16.66	0.01	0.01	48.12	133

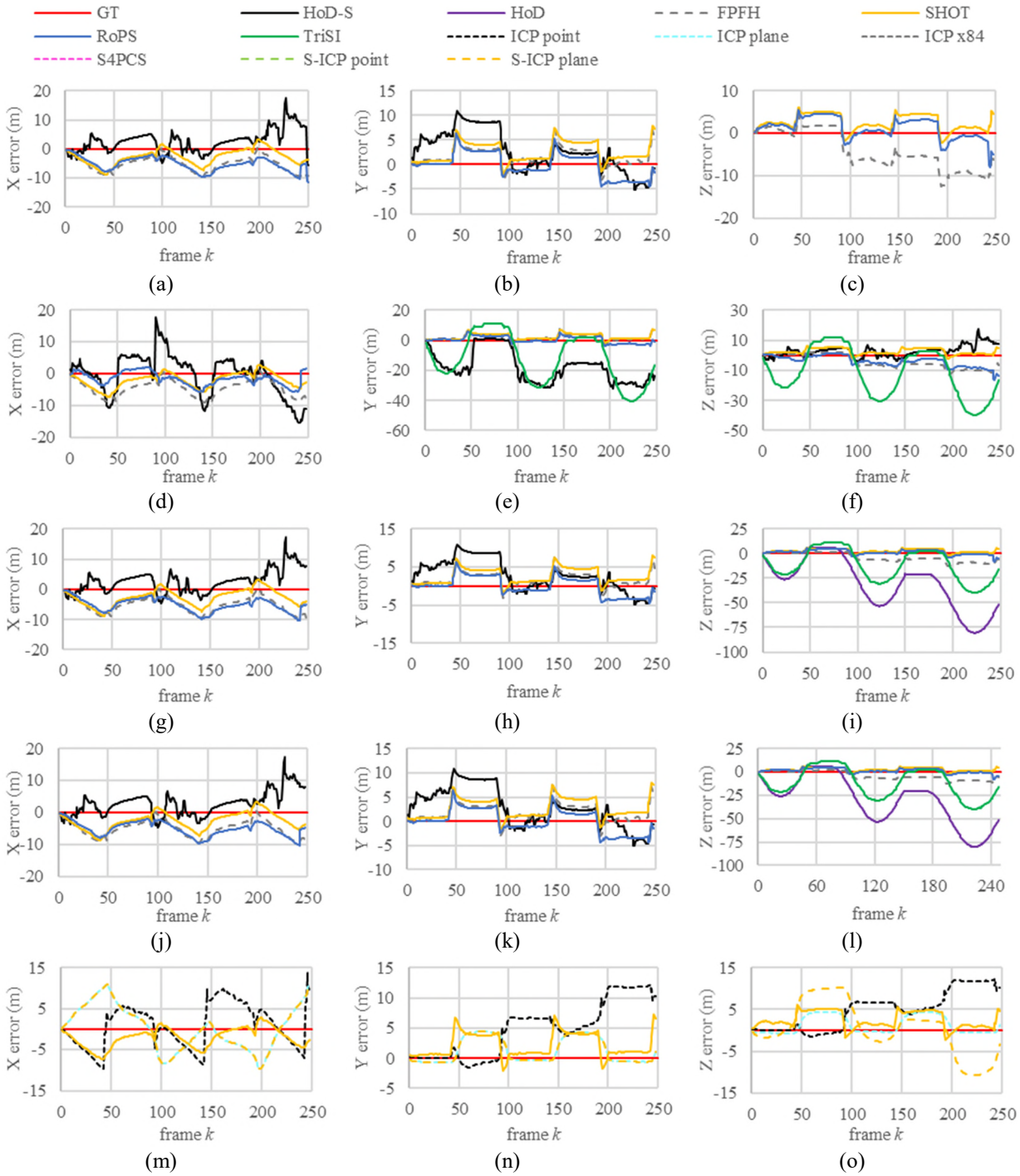


Figure A6: *Sim-Orion* scenario 6 odometry plots of the 3D descriptors combined with the adaptive filter (a)-(b) *H∞*, (c)-(d) *Kalman,* (e)-(f) *αβ*, (g)-(h) *SDRE*, and (i)-(j) competitor methods and top performing proposed method SHOT/ *Kalman* (due to large errors and for better readability we omit HoD from (a), (b), (c), (d), (e), (f), (g), (h), (j), (k), HoD-S from (c), (i), (l), TriSI from (a), (b), (c), (d),(g), (h), (j), (k), and S4PCS, Sparse ICP point2point and ICP x84 from (m)-(o))

Table A7: Performance metrics for the *Sim-Bennu* scenario

Descriptor	drift (m)	Terror (%)	e^{T}_{max}	e^{T}_{avg}	max Error (m)			Average error (m)			e^{R}_{max}	e^{T}_{avg}	e^{RT}	t (ms)
					X	Y	Z	X	Y	Z				
adaptive *H*∞ recursive filtering														
HoD-S	2.02	0.26	***2.04***	***1.15***	***0.21***	***0.64***	***1.93***	***0.15***	***0.38***	***0.92***	0.14	0.28	***10.76***	***47***
FPFH	3.41	0.43	5.59	3.67	***0.21***	4.31	3.43	0.52	1.97	2.22	0.14	0.27	18.12	440
SHOT	***1.22***	***0.15***	7.05	4.54	0.25	4.98	4.99	0.58	2.89	2.92	0.13	0.26	22.72	516
HoD	77.69	9.85	104	56.85	40.82	67.82	67.71	27.89	27.51	27.46	***0.03***	***0.06***	69.37	53
RoPS	11.07	1.40	11.08	6.19	1.73	7.16	8.26	0.76	2.95	3.56	0.14	0.28	20.72	591
TriSI	50.19	6.37	55.81	39.46	36.34	29.85	29.83	25.40	17.93	17.93	***0.03***	***0.06***	59.38	457
adaptive *Kalman* recursive filtering														
HoD-S	8.67	1.10	29.09	16.80	9.38	24.27	12.99	3.68	11.23	6.43	0.02	0.08	40.24	***46***
FPFH	3.40	0.43	***5.57***	***3.67***	***0.21***	***4.32***	***3.42***	***0.52***	***1.96***	***2.22***	***0.01***	***0.01***	***17.96***	439
SHOT	***1.20***	***0.15***	7.06	4.54	0.25	4.98	4.99	0.57	2.89	2.92	***0.01***	***0.01***	22.57	516
HoD	77.69	9.85	104	56.85	40.83	67.82	67.70	27.89	27.51	27.46	***0.01***	***0.01***	69.34	53
RoPS	11.07	1.40	11.08	6.19	1.74	7.15	8.27	0.76	2.95	3.56	***0.01***	***0.01***	20.59	590
TriSI	50.19	6.37	55.82	39.46	36.34	29.86	29.83	25.40	17.93	17.93	***0.01***	***0.01***	59.35	456
adaptive *αβ* recursive filtering														
HoD-S	2.04	0.26	***2.05***	***1.16***	***0.20***	0.64	***1.94***	***0.15***	***0.38***	***0.93***	***0.01***	***0.01***	***10.65***	***41***
FPFH	3.42	0.43	5.59	3.67	0.21	***4.32***	3.42	0.52	1.97	2.23	***0.01***	***0.01***	17.96	437
SHOT	***1.21***	***0.15***	7.06	4.54	0.25	4.98	4.99	0.57	2.90	2.92	***0.01***	***0.01***	22.58	509
HoD	77.71	9.86	104	56.85	40.84	67.83	67.71	27.90	27.52	27.46	***0.01***	***0.01***	69.35	51
RoPS	11.07	1.40	11.09	6.19	1.74	7.15	8.27	0.76	2.95	3.56	***0.01***	***0.01***	20.58	583
TriSI	50.20	6.37	55.83	39.47	36.36	29.86	29.83	25.40	17.93	17.93	***0.01***	***0.01***	59.35	453
adaptive *SDRE* recursive filtering														
HoD-S	1.97	0.25	***1.98***	***1.12***	0.33	***0.76***	***1.80***	***0.21***	***0.44***	***0.86***	0.14	0.28	***11.14***	***42***
FPFH	3.54	0.45	5.69	3.68	***0.30***	4.41	3.52	0.50	1.96	2.22	0.14	0.27	18.09	437
SHOT	***1.40***	***0.18***	6.92	4.51	0.34	4.89	4.89	0.62	2.88	2.91	0.13	0.26	22.83	509
HoD	77.68	9.85	104	56.84	40.84	67.79	67.69	27.90	27.50	27.45	***0.03***	***0.06***	69.37	52
RoPS	10.88	1.38	10.90	6.09	1.61	7.03	8.15	0.71	2.91	3.50	0.14	0.28	20.46	584
TriSI	50.21	6.37	55.80	39.47	36.37	29.83	29.81	25.41	17.93	17.93	***0.03***	***0.06***	59.39	453
competitor schemes														
ICP point	150	19.05	275.76	177.24	147.90	194.21	116.25	85.52	90.65	79.85	-143	38.05	458.55	13
ICP plane	39.90	5.06	52.43	38.27	25.85	32.17	32.21	19.93	18.58	18.62	0.01	0.01	50.07	82
ICP x84	219	27.86	532.12	330.03	281.96	349.77	238.77	160	138.63	152.07	-146	41.38	653.59	35
S4PCS	22.72	2.88	34.53	20.89	14.95	20.87	23.09	9.07	10.72	10.35	81.95	172.80	125.48	1137
S-ICP point	39.90	5.06	52.43	38.27	25.85	32.17	32.21	19.93	18.58	18.62	0.01	0.01	50.07	99
S-ICP plane	39.90	5.06	52.43	38.27	25.85	32.17	32.21	19.93	18.58	18.62	0.01	0.01	50.07	112

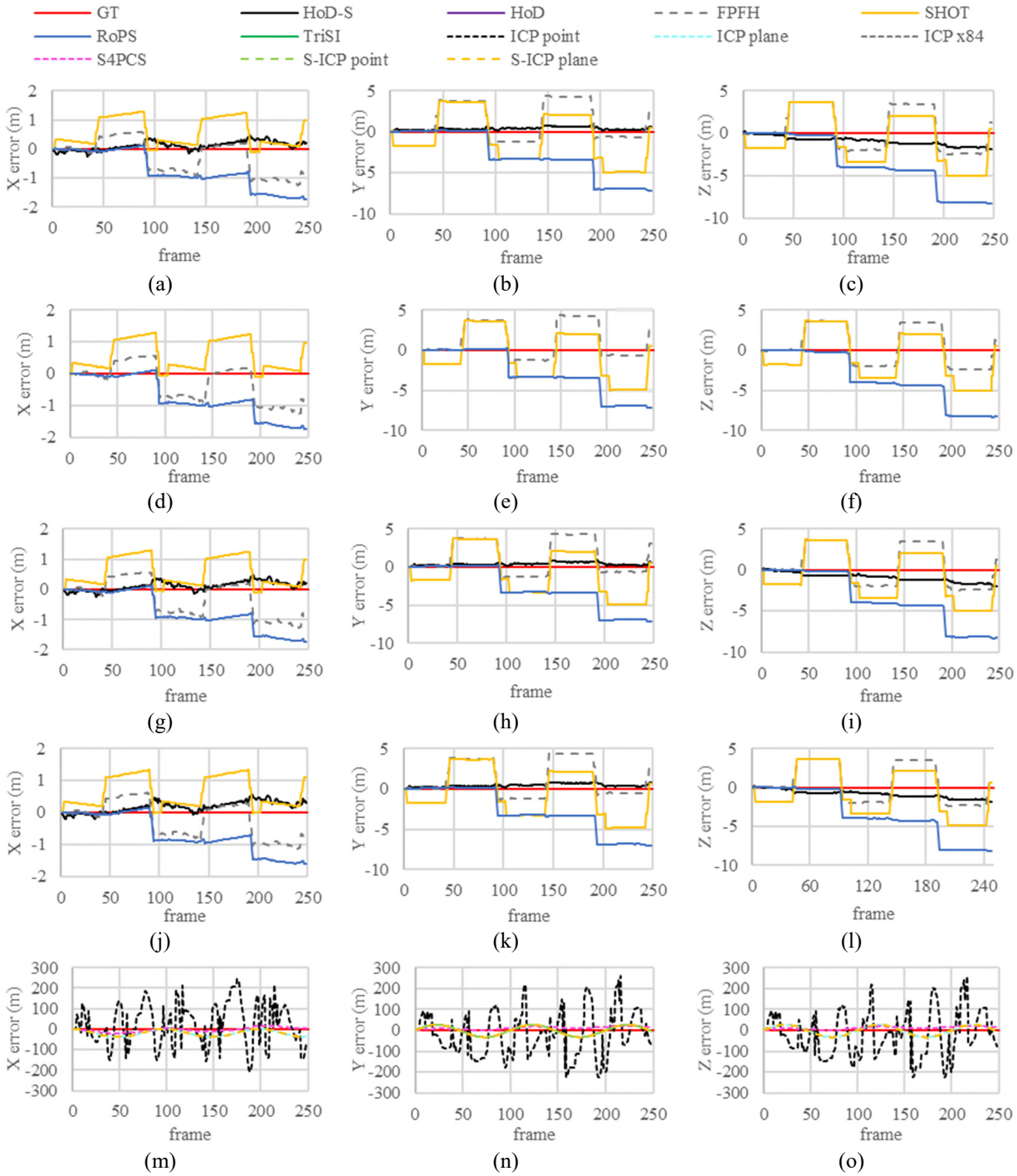


Figure A7: *Sim-Bennu* scenario 7 odometry plots of the 3D descriptors combined with the adaptive filter (a)-(b) *H∞*, (c)-(d) *Kalman,* (e)-(f) *αβ*, (g)-(h) *SDRE*, and (i)-(j) competitor methods (due to large errors and for better readability we omit HoD and TriSI from all plots, HoD-S from (d)-(f) and ICP-x84 from (m)-(o). We also omit the top performing HoD-S/ *αβ* from (m)-(o) as errors compared to competitor methods are at least one order of magnitude lesser)

Appendix B

For better readability this appendix presents the relationship between ICP performance and the number of iterations, for several ICP variants.

Table B1: ICP performance vs. number of iterations

Scenario	ICP variant	normalized e^{T}_{avg}		normalized e^{R}_{avg}		normalized e^{RT}		t (ms)	
		20 iterations	1000 iterations	20 iterations	1000 iterations	20 iterations	1000 iterations	20 iterations	1000 iterations
Real-FB	ICP point	0.086	0.086	11.402	11.402	11.649	11.649	***5***	6
	ICP plane	0.131	0.131	-23.090	-23.090	25.872	25.872	9	9
	ICP x84	***1.291***	6.126	-22.321	***-16.821***	***27.893***	38.699	***41***	983
	S-ICP point	0.266	***0.131***	***0.001***	-23.090	***0.462***	25.872	***145***	191
	S-ICP plane	0.266	***0.105***	***0.001***	0.313	***0.462***	0.602	***150***	305
Real-Curved	ICP point	***0.136***	0.227	***8.447***	12.427	***8.723***	12.745	6	6
	ICP plane	***0.146***	0.177	-13.567	***12.922***	15.792	***13.225***	***10***	28
	ICP x84	***0.343***	11.547	10.971	***-9.621***	***11.341***	31.967	***53***	1058
	S-ICP point	***0.159***	0.176	***0.001***	12.933	***0.276***	13.234	137	***28***
	S-ICP plane	0.159	***0.025***	***0.001***	1.196	***0.276***	1.343	***145***	250
Sim-EoI	ICP point	***0.134***	0.135	-0.565	-0.565	***0.858***	0.859	30	30
	ICP plane	0.126	0.126	-0.576	-0.576	0.811	0.811	***31***	51
	ICP x84	***0.158***	1.004	-0.570	***-0.445***	***0.887***	2.223	***49***	1059
	S-ICP point	***0.039***	0.126	***0.001***	-0.576	***0.060***	0.811	276	***51***
	S-ICP plane	***0.039***	0.088	***0.001***	0.434	***0.060***	0.610	***327***	14702
Sim-Helical	ICP point	0.041	***0.019***	***0.057***	0.339	***0.192***	0.422	***8***	10
	ICP plane	10^{12}	***0.048***	0.578	***0.297***	$3x10^{11}$	***0.461***	***17***	127
	ICP x84	11.933	***3.500***	-0.788	***-0.657***	18.400	***5.940***	***36***	417
	S-ICP point	0.190	***0.055***	***0.001***	0.285	***0.243***	0.411	107	***84***
	S-ICP plane	0.196	***0.008***	0.001	0.001	0.248	***0.064***	***105***	123
Sim-Voyager	ICP point	***0.038***	0.083	***0.209***	0.220	***0.296***	0.344	***7***	10
	ICP plane	$7x10^{5}$	***0.025***	0.174	***0.126***	$6x10^{5}$	***0.189***	***8***	19
	ICP x84	***0.358***	0.362	0.215	***-0.164***	***0.710***	0.749	***16***	284
	S-ICP point	0.037	***0.017***	***0.001***	0.053	***0.059***	0.102	171	***14***
	S-ICP plane	0.037	***0.028***	***0.001***	0.120	***0.059***	0.176	***177***	307
Sim-Orion	ICP point	***0.012***	0.065	-0.165	***0.064***	0.333	***0.185***	***13***	14
	ICP plane	761.035	***0.045***	0.192	***0.062***	1014.713	***0.158***	***12***	14
	ICP x84	***0.084***	0.963	-0.207	***-0.176***	***0.352***	1.701	***24***	407
	S-ICP point	***0.042***	0.069	***0.001***	-0.223	***0.061***	0.354	125	***15***
	S-ICP plane	***0.042***	0.069	***0.001***	0.004	***0.061***	0.122	133	***52***
Sim-Bennu	ICP point	0.371	***0.055***	***-0.199***	0.212	0.731	***0.300***	***11***	16
	ICP plane	$3x10^{14}$	***0.143***	-0.189	***0.184***	$5x10^{14}$	***0.373***	***36***	84
	ICP x84	***0.265***	0.652	***-0.014***	-0.223	***0.604***	1.127	***35***	419
	S-ICP point	0.049	***0.011***	***0.001***	-0.157	***0.064***	0.339	99	***55***
	S-ICP plane	0.049	***0.034***	***0.001***	0.107	***0.064***	0.172	***112***	300