DeepLO: Multi-projection Deep LIDAR Odometry for Space Orbital Robotics Rendezvous Relative Navigation

O. Kechagias-Stamatis [a,b,*], N. Aouf [b] V. Dubanchet[c] and M. A. Richardson [a]

[a] Centre of Electronic Warfare, Cranfield University Defence and Security, Shrivenham, SN6 8LA, UK

[b] Department of Electrical and Electronic Engineering, City University of London, EC1V 0HB, UK

[c] Department of Space Robotics and GNC/AOCS, Thales Alenia Space, Cannes, France

Abstract

This work proposes a new Light Detection and Ranging (LIDAR) based navigation architecture that is appropriate for uncooperative relative robotic space navigation applications. In contrast to current solutions that exploit 3D LIDAR data, our architecture suggests a Deep Recurrent Convolutional Neural Network (DRCNN) that exploits multi-projected imagery of the acquired 3D LIDAR data. Advantages of the proposed DRCNN are; an effective feature representation facilitated by the Convolutional Neural Network module within DRCNN, a robust modelling of the navigation dynamics due to the Recurrent Neural Network incorporated in the DRCNN, and a low processing time. Our trials evaluate several current state-of-the-art space navigation methods on various simulated but credible scenarios that involve a satellite model developed by Thales Alenia Space (France). Additionally, we evaluate real satellite LIDAR data acquired in our lab. Results demonstrate that the proposed architecture, although trained solely on simulated data, is highly adaptable and is more appealing compared to current algorithms on both simulated and real LIDAR data scenarios affording better odometry accuracy at lower computational requirements.

Keywords: Convolutional Neural Networks, Deep learning, LIDAR, Multi-dimensional processing, Recurrent Neural Networks, Relative navigation, Robotics

1. Introduction

Odometry for space robotics applications is an active research area due to the increasing number of robotics platforms deployed into space and the necessity for their autonomous operation. Relative space navigation of a *Source* spacecraft platform in relation to a non-cooperative *Target* platform, i.e. with unknown attitude (pose) is

* Corresponding author
E-mail address: o.kechagiasstamatis@Cranfield .ac.uk

considered a hot research topic. This is because precise relative space navigation will enable a *Source* spacecraft to perform autonomous close-proximity maneuvers and achieve uncooperative rendezvous with a non-cooperative *Target* platform, contributing towards autonomous active space debris removal and satellite inspection and docking operations. In any of these scenarios, the *Target* platform is likely to be non-cooperative and therefore unable to exchange with the *Source* platform its pose actively or passively, e.g. via known markers placed on the *Target*. Therefore, the *Source* spacecraft must estimate its relative position and attitude with respect to the *Target* platform by utilizing only its onboard sensors. Current solutions involve 2D visual data in a monocular [1,2] or a stereo camera configuration [3–6], 2D Infrared (IR) thermal data [7], and 3D Light Detection and Ranging (LIDAR) data [8,9,18–24,10–17]. A comprehensive review of spacecraft pose determination techniques for close-proximity operations is presented in [25].

Even though each modality has its own strengths and weaknesses, LIDAR is preferred due to its proven robustness in the outer space environment [23]. Although visual [26] and IR [7] sensors have already been exploited, with the latter having several advantages over the former such as operating during day and night under harsh illumination conditions. However, the accuracy of IR thermal odometry relies on the temperature of the *Target*'s components that may be affected by internal parameters, e.g. heat dissemination of the platform's components, and by external parameters, e.g. reflection of the sun's radiation. This temperature fluctuation may affect the robustness of the IR feature detection and matching process, which are the core modules in [7]. On the contrary, odometry based on 3D LIDAR data outmatches its 2D counterparts (visual and IR) as it operates during day, night and under poor visibility conditions, is independent of the *Target*'s thermal properties, is capable of revealing the underlying structure of an object [27,28], and can provide both 3D position and intensity data [25].

Spurred by the advantages of 3D LIDAR odometry for space robotics applications, the research community suggests quite a few techniques that use LIDAR data. Specifically, [15] exploits RGB-D data and suggests a sliding window filtering scheme that is combined with a Gauss-Newton method, while initial pose estimation is generated by an OPnP algorithm. Galante *et al.* [13] propose the *Argon* relative navigation system that comprises of a stereo optical camera setup and a flash LIDAR device. On the data acquired, *Argon* applies edge detection and a custom Iterative Closest Point (ICP) scheme to estimate a 6-degrees of freedom pose. Other solutions perform pose initialization via template matching and then apply the typical ICP [29] for frame-to-frame pose estimation [9,16,17,30,31]. Variants of that architecture substitute template matching either with Principle Component Analysis

(PCA) [9,32] or with global 3D feature matching using the Oriented Unique Repeatable Clustered Viewpoint Feature Histogram (OUR-CVFH) [14,33]. Other solutions available in literature fuse pose estimation based on OUR-CVFH or on Spin Images [34] (a 3D local feature descriptor) combined with ICP, with gyroscopic data and then perform *Target* platform tracking using a Multiplicative Extended Kalman Filter (MEKF) [10,19,32]. Volpe *et al.* [11] utilize 2D features from the visual domain combined with LIDAR based distance estimation and Unscented Kalman Filtering (UKF) for performance improvement. Alternatives to pure ICP registration for pose estimation have also been proposed, by substituting ICP with a UKF filter, an iterative least-squares (LS) scheme, or with an Extended Kalman Filter (EKF) [21], [20]. Lately, in [18] the HoD-S 3D local feature matches are recursively filtered via an adaptive H∞ filter affording an accurate space navigation solution. This work is further extended in [35] where several combinations of 3D features and recursive filtering schemes are evaluated on both real and simulated scenarios. Kechagias-Stamatis *et al.* [24] suggest a non-standard space navigation solution where 3D LIDAR data are remapped into multiple 2D planar projections. On the latter ones, 2D keypoint detection, 2D feature description, and cross-plane feature matching are applied, and 2D correspondences are back-projected in the initial 3D space and are then filtered via a Kalman or an H∞ filter. The novelty of this architecture is the combination of advantages of both the 3D and 2D data space and ultimately affording accurate odometry at a low processing time. A comprehensive list of current 3D space odometry methods is presented in Table 1.

It is worth noting that in contrast to space robotics odometry, terrestrial odometry (ground and aerial) is more mature, spreading over several data domains such as 2D visual in a monocular or stereo camera configuration [36,37], 2D IR [38], fusion of visual with IR data [39], fusion of 3D LIDAR with visual data [40], 3D LIDAR fused with inertial data [41], purely on 3D data [42] or exploiting RGB-D data [43–45]. For completeness, the reader is referred to [46] for an extensive review on the odometry methods for terrestrial applications. However, simply extending current algorithms designed for terrestrial robotics odometry into the field of space robotics odometry is not an optimum solution because the space environment is completely different posing additional challenges. Major differences are; first the lack of surrounding objects forcing odometry to rely on a relatively limited number of vertices belonging to a single and usually small-sized object, and second, the low point cloud complexity of space objects reducing the unique and non-degenerated geometries on the *Target*, which are important for the odometry solution to converge to an acceptable estimation.

Table 1.
Current 3D space odometry architectures

Nº	Reference	Year	Point cloud type	Hardware	Method
1	Galante *et al.* [13]	2012	Real	stereo camera and LIDAR	2D edge tracking and custom ICP for pose estimation
2	Sell *et al.* [14]	2014	Real	LIDAR	OUR-CVFH features for pose initialization and ICP for point cloud registration and pose estimation
3	Opromolla *et al.* [16]	2014	Simulated	LIDAR	optimized template matching for pose initialization and ICP for point cloud registration and pose estimation
4	Opromolla *et al.* [17,30]	2015	Simulated	LIDAR	optimized template matching for pose initialization and ICP for point cloud registration and pose estimation
5	Rhodes *et al.* [32]	2016	Simulated	Gyroscope, star tracker, LIDAR	OUR-CVFH or Spin Images features for pose initialization and ICP for point cloud registration and pose estimation that is fused with sensor inputs via a MEKF module
6	Liu, Zhao and Bo [31]	2016	Simulated	LIDAR	template based pose initialization and ICP object tracking
7	Woods and Christian [10]	2016	Simulated	Gyroscope, GPS, star tracker, LIDAR	OUR-CVFH features for pose initialization and ICP for point cloud registration and pose estimation that is fused with sensor inputs via a MEKF module
8	Opromolla *et al.*[9]	2017	Real	LIDAR	optimized template matching or PCA for pose initialization and ICP for point cloud registration and pose estimation
9	Song [15]	2017	Simulated	RGB-D camera	sliding window filter (SWF) smoothing to estimate the structure and pose on SE(3). Gauss-Newton (GN) method is implemented for each window with an initial guess generated by OPnP algorithm
10	Volpe et al. [11]	2017	Simulated	Optical camera and LIDAR	2D feature tracking based odometry combined with LIDAR for distance measurement and UKF
11	Martinez et al. [12]	2017	Simulated	ToF camera	geometrical based orientation estimation for pose initialization, ICP for pose estimation and EKF for kinematic estimation
12	Rhodes, Christian and Evans [19]	2017	Simulated	LIDAR	OUR-CVFH features or OUR-CVFH combined with MEKF for trajectory smoothing
13	Dietrich and McMahon [20]	2017	Simulated	LIDAR	point cloud registration using UKF
14	Dietrich and McMahon [21]	2018	Simulated	LIDAR	point cloud registration using UKF, LS and EKF
15	Jalalabadi and Malaek [23]	2018	Simulated	LIDAR	NN ICP registration using kd-tree, M-UKF trajectory filtering and Upper Bound fusion for multi observer data acquisition
16	Kechagias-Stamatis and Aouf [18]	2019	Real	LIDAR	HoD-S local features with adaptive H∞ recursive filtering
17	Kechagias-Stamatis, Aouf and Richardson [24]	2019	Simulated	LIDAR	3D-to-multi-2D point cloud data remapping followed by multi-2D keypoint detection, description, and cross-plane feature matching. 2D correspondences are then back-projected to the 3D space and filtered via Kalman or H∞.
18	Chen et al. [47]	2019	Real	Optical camera	a deep network predicts the position of the predefined landmark points in the input image and pose estimation is performed by establishing the 2D-3D correspondences between the input image and the created 3D Target model
19	Kechagias-Stamatis, Aouf and Dubanchet [35]	2019	Real and Simulated	LIDAR	Evaluation of multiple combinations of 3D features and recursive filtering schemes

Spurred by the advantages of 3D LIDAR odometry and the appealing performance of deep learning for terrestrial odometry applications [46], we present a Deep Recurrent Convolutional Neural Network (DRCNN) that is appropriate for space relative navigation. Our deep LIDAR odometry solution incorporates a Convolutional Neural

Network (CNN) module and a Recurrent Neural Network (RNN) module. Despite the CNN and RNN modules are applied on 2D data, we exploit 3D LIDAR point cloud data by properly remapping the 3D vertices to the 2D data domain. Advantages of the proposed DRCNN architecture are:

a. Effective feature representation, via adopting a CNN module, where features can be generalized and ultimately used in untrained environments.

b. Robust and automatic modeling of the navigation dynamics due to the RNN module.

c. Transforming the acquired 3D LIDAR data into multiple 2D depth maps to reduce the processing burden but simultaneously minimize information loss.

d. DRCNN is highly adaptive as it can be exclusively trained on simulated data and tested on both simulated as well as real LIDAR data affording better odometry accuracy than current algorithms even if the data modality changes. This feature is very important because DRCNN is capable of offline training on simulated data but still capable of accurate odometry on untrained data of a different modality, increasing the flexibility of the robotic platform to perform odometry on a broad spectrum of scenarios.

The remainder of the article is organized as follows: Section 2 introduces the proposed DRCNN odometry architecture and Section 3 evaluates the suggested technique against current ones on realistic simulated and real LIDAR data scenarios. Our conclusions are presented in Section 4.

2. DRCNN Odometry

The problem addressed in this work is LIDAR odometry for relative space robotics navigation of a *Source* platform equipped with a 3D LIDAR sensor that moves in the 3D space relative to a *Target* platform with an unknown pose. Therefore, given two consecutive *Target* point clouds $\boldsymbol{P}_k = \{p_k^1, ..., p_k^a\}$ and $\boldsymbol{P}_{k+1} = \{p_{k+1}^1, ..., p_{k+1}^b\}$ captured by the *Source*'s LIDAR sensor, with each vertex being in the form $p_k=(x_k,y_k,z_k)$ and $p_{k+1}=(x_{k+1},y_{k+1},z_{k+1})$, the generic odometry process aims to calculate a rigid body transformation,

$$R^* = \begin{bmatrix} R & |T \\ 0 & |1 \end{bmatrix} \quad (1)$$

where R is the rotation and T the translation component, that remap $\boldsymbol{P}_k$ to $\boldsymbol{P}_{k+1}$:

$$p_{k+1} = Rp_k + T \quad (2)$$

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

Despite that attitude *R* may originate from Euler or Quaternion angle encodings, we express *R* via a rotation matrix [2,48–50] but we ensure it conforms to the SO(3) constraints both during the training and the evaluation phases presented in Section 3. We chose this angular representation as it is suitable for the regression we develop through the last part of our network and it is also meeting the motion estimation manner used in the competitor filtering approaches presented in Section 3. Hence, maintaining the same way of regressing the motion quantities is useful for the comparison of the results, because inevitably the performance difference between the proposed technique and the competitor methods is not due to the angular representation but purely to the odometry pipeline. Regarding the SO(3) constraints of the rotation matrix elements, our training scheme also considers learning these constraints from the known ground truth *R* that is used. In fact, the DRCNN during training is naturally imposed to learn the SO(3) constraints because the ground truth *R* is free of these constraints. Thus, implicitly the *R* estimation is practically optimized during training to meet the SO(3) constraints and this embedded knowledge of DRCNN is then used during testing, providing rotation matrices respecting these constraints. However, if during testing DRCNN delivers a rotation matrix that does not meet the SO(3) constraints, then the solution is rejected and the odometry solution is recalculated with additional randomness to overcome the previous local minimum. In case the new odometry solution still does not conform to the constraints, then it is discarded and $R_{k+1}^{*} = R_k^{*}$. However, due to optimizing the *R* estimation to meet the SO(3) constraints, in the vast majority of pose instances the initial rotation matrix delivered met these constraints. It is worth noting that in contrast to [2,48–50] that express R via a rotation matrix without explicitly dealing theoretically with the SO(3) constraints, in our work we practically tackle the theoretical aspect of these constraints.

Additionally, it should be noted that Eq. (1) – (3) should ideally involve the true point correspondences between the $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ point clouds that exactly link the same physical points. Typical methods for establishing point correspondences between $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ involve 3D local or global feature matching of a subset of the vertices belonging to $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$. However, in real-world scenarios, point correspondences are unknown, and depending on the relative motion between $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ there might not be an actual one-to-one point correspondence at all. In that

case, the odometry method seeks the optimum point correspondences based on a minimum feature matching error, where each feature is the representative description of the surroundings of each point. The optimum point correspondences are then input to Eq. (1) – (3) aiming at ultimately presenting the minimum error between the calculated and the ground truth *R* and *T* values, respectively. Though the point correspondence quality defines the odometry accuracy and any deviation of the estimated *R* and *T* compared to the corresponding ground truth values can only be presented during controlled experimental setups and not during real-world scenarios.

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

$$R_u^* = \prod_{\mu=1}^{u} R_\mu^* \qquad (4)$$

In the space-related literature, R^* is typically estimated by a two-stage process, i.e. coarse *Target* pose initialization via template matching or 3D feature matching (global or local features), and then fine *Target* pose estimation via an iterative process. However, as presented in Section 3, current solutions suffer from; a high processing burden, template/feature mismatching, and ICP not always converging to an optimum odometry solution.

2.1. Pre-processing LIDAR data

Despite 3D data offering quite a few advantages over their 2D counterpart (see Section 1), the 3D data modality imposes a higher computational burden [28]. Therefore, in our proposed space robotics navigation architecture we take advantage of both data modalities (3D and 2D) by remapping $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ from the 3D to the 2D domain where we create three 2D depth images. Specifically, for $\boldsymbol{P}_k$ and accordingly for $\boldsymbol{P}_{k+1}$, we transfer the XYZ_{LIDAR} reference frame that is aligned and centered at the LIDAR sensor coordinate frame onboard the *Source* platform to the geometric center of the target point cloud $\boldsymbol{P}_k$ and create the XYZ_{Target} reference frame. Then, we quantize the floating-point vertex coordinates $\boldsymbol{P}_k = \{p_k^1, ..., p_k^a\}$ into $\boldsymbol{P}_{Q\_k} = \{p_{Q\_k}^1, ..., p_{Q\_k}^a\}$ with,

$$p_{Q\_k}\left(x_{Q\_k}, y_{Q\_k}, z_{Q\_k}\right) = \left\lfloor q_f \cdot p_k\left(x, y, z\right) \right\rfloor \qquad (5)$$

where q_f is a quantization factor, $\lfloor \cdot \rfloor$ the bottom-round process, and α the point cloud cardinality. Next, we project $\boldsymbol{P}_{Q\_k}$ on each plane of the XYZ_{Target} reference frame by utilizing an orthographic projection process P_{ortho}:

$$\tilde{p}_{Q\_k} = \begin{bmatrix} \tilde{x}_q \\ \tilde{y}_q \\ \tilde{z}_q \\ 1 \end{bmatrix} = P_{ortho} \cdot p_{Q\_k} = \begin{bmatrix} c_1 & 0 & 0 & 0 \\ 0 & c_2 & 0 & 0 \\ 0 & 0 & c_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x_{Q\_k} \\ y_{Q\_k} \\ z_{Q\_k} \\ 1 \end{bmatrix} \quad (6)$$

Depending on the projection plane, we substitute with zero the appropriate binary remapping coefficient $c_1, c_2, c_3 \in \{0,1\}$ of P_{ortho}, e.g. for c_1=c_2=1 and c_3=0, the XY depth image $\tilde{p}_{Q\_k}^{XY}$ is created. The three orthographic projections $\tilde{p}_{Q\_k}^{XY}, \tilde{p}_{Q\_k}^{XZ}, \tilde{p}_{Q\_k}^{YZ}$ are depth images created in a form of parallel projection of the point cloud onto the corresponding planes of the XYZ_{Target} reference frame, which are simplified versions of $\boldsymbol{P}_{Q\_k}$. The depth value of each pixel within each $\tilde{p}_{Q\_k}$ is unique and represents the distance between the *Target* and the XYZ_{Target} reference frame. An advantage of this projection scheme is its fast execution time. The reason for translating the XYZ_{LIDAR} to the *Target* platform and create the XYZ_{Target} reference frame is to preserve the *Target's* details during the quantization process of Eq. (5), while in parallel keep the projection images $\tilde{p}_{Q\_k}^{XY}, \tilde{p}_{Q\_k}^{XZ}, \tilde{p}_{Q\_k}^{YZ}$ small to afford low memory and processing requirements. Given that the x-axis is towards the *Target* platform, the z-axis is upwards with respect to the LIDAR sensor and the y-axis is at right angles, by projecting the *Target* point cloud on the XYZ_{LIDAR} frame the $\tilde{p}_{Q\_k}^{XZ}$ and the $\tilde{p}_{Q\_k}^{XY}$ would contain no depth information for the distance between the *Source* and the *Target*, but would inevitably increase the $\tilde{p}_{Q\_k}^{XZ}$ and the $\tilde{p}_{Q\_k}^{XY}$ images. It is worth noting that we also investigated preserving the original XYZ_{LIDAR} frame and applying range gating to create projections that contain only the *Target*. Though, the processing time for range gating was slightly more processing deficient than translating the reference frame.

Selecting q_f is not trivial as it highly affects the size of the multi-2D projections and the number of details contained in each projection. In fact, high q_f values create large non-uniform 2D projections that impose a greater computational burden and memory requirement of the processing platform. On the contrary, small q_f values discard the point cloud topology information during the 3D to 2D remapping process by subsampling the projected data. For our trials we set $q_f = 20$ such as to fully exploit the memory capability of our computer platform, while still preserving the *Target*'s point cloud topology. For completeness, Fig. 1 presents the $\tilde{p}_{Q\_k}^{XY}$ for several q_f values.

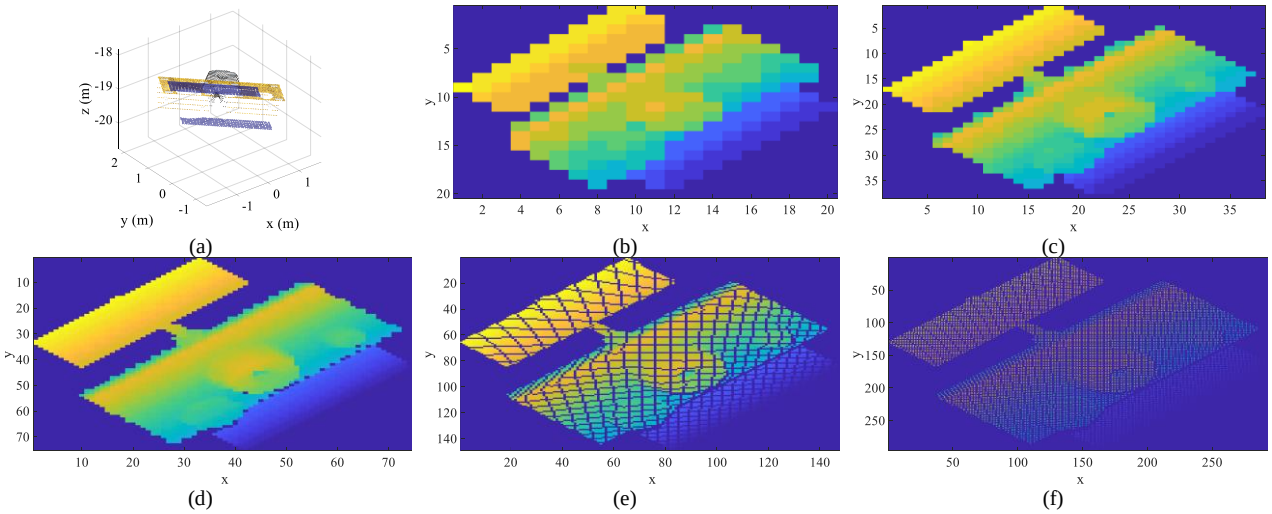


Fig. 1 Interplay between the $\tilde{p}_{Q\_k}^{XY}$ projection and q_f (a) 3D point cloud model and $\tilde{p}_{Q\_k}^{XY}$ projection with a q_f value of (b) 5 (c) 10 (d) 20 (e) 40 (f) 80 (closer pixels are hotter, best seen in color)

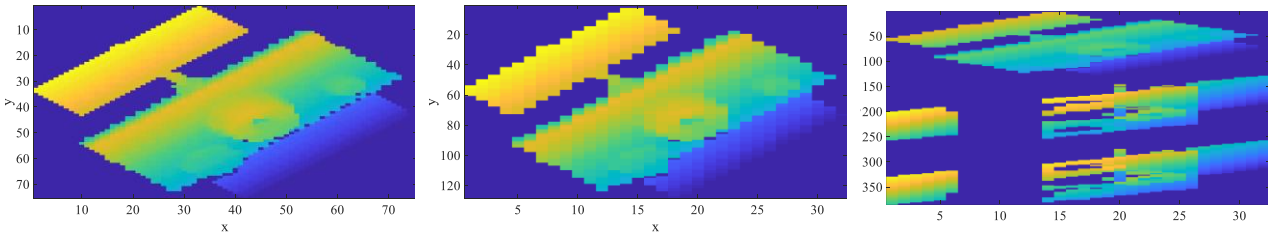
2.2. Deep Recurrent Convolutional Neural Network

Given the three depth images of $\boldsymbol{P}_{Q\_k}$, i.e. $\tilde{p}_{Q\_k}^{XY}, \tilde{p}_{Q\_k}^{XZ}, \tilde{p}_{Q\_k}^{YZ}$, we create a stacked image,

$$I_k = \tilde{p}_{Q\_k}^{XY} \left\| \tilde{p}_{Q\_k}^{XZ} \right\| \tilde{p}_{Q\_k}^{YZ} \qquad (7)$$

where $\|(\cdot)$ is a 1D vertical concatenation process. Then we create $I_{k,k+1} = I_k \| I_{k+1}$ that is input to our proposed DRCNN network, which comprises of a CNN module that is followed by an RNN module.

However, the image size of each projection $\tilde{p}_{Q\_k}^{XY}, \tilde{p}_{Q\_k}^{XZ}, \tilde{p}_{Q\_k}^{YZ}$ is not consistent and thus all three projections must be resized to apply Eq. (7). Hence, prior to creating I_k we resize each projection to 128x32 pixels by applying to it a Nearest-neighbor interpolation scheme. For this work, the advantage of the Nearest-neighbor interpolation over the bilinear and bicubic interpolations is not smoothing the topology information of the projections and thus attaining lower odometry errors. It is worth noting that during our preliminary trials we evaluated several image size combinations but concluded that the 128x32 image size per projection was the optimum. Hence, the depth image input images I_k and I_{k+1} have a size of 384 pixels x 32 pixels. A study on several depth image input sizes and interpolations schemes is presented in Section 3.4. An example of the I_k processing stages is presented in Fig. 2.



(a) (b) (c)

Fig. 2 (a) $\tilde{p}_{Q\_k}^{XY}$ projection with $q_f = 20$ (b) Nearest-neighbor interpolated of $\tilde{p}_{Q\_k}^{XY}$ (c) stacked image containing the interpolated $\tilde{p}_{Q\_k}^{XY}, \tilde{p}_{Q\_k}^{XZ}, \tilde{p}_{Q\_k}^{YZ}$ (closer pixels are hotter, best seen in color)

The CNN module performs feature extraction on $I_{k,k+1}$ and since both components of $I_{k,k+1}$ involve 2D depth images, the associated features are geometric rather than texture-based. This is important as the features can be generalized increasing the overall robustness. The configuration of the CNN module is presented in Table 2 (layers 1-8). Initially, the 2D depth image $I_{k,k+1}$ of size 768 pixels x 32 pixels is input to layer 1. The size of $I_{k,k+1}$ is experimentally defined during training and is the largest possible such as to preserve the details of $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ and ultimately afford an appealing odometry solution. However, the size of $I_{k,k+1}$ is limited by the available GPU memory. The following six layers are three convolutional layers, each of which is followed by a Rectified Linear Unit (ReLU) activation layer. The reasoning for exploiting a shallow CNN architecture rather than a deep one is linked to the $I_{k,k+1}$ image size and the selection of the q_f, both of which determine the level of details within $I_{k,k+1}$. Specifically, from Fig. 2 (c) it is obvious that the level of details in $I_{k,k+1}$ is not high and thus the high-level domain-based features produced by the deep convolutional layers regularly used in RGB based odometry and classification applications are not optimum for our work. Instead, low-level features extracted from shallow CNNs are more appropriate for our $I_{k,k+1}$ projections. The receptive filter size of each convolutional layer is fixed to 5x5, which for the quantization factor q_f used during the multi-projection process, enhances capturing large features. Regarding the number of channels in each convolutional layer, we intentionally double them in any successive convolutional layer to afford the CNN module to learn more features. Finally, the CNN module concludes with a Fully Connected layer to bridge the CNN and RNN modules. This is because the cardinality of the output tensor *ReLU_3* exceeds the GPU memory limitations of our testing platform and thus we use the *Fully Connected_1* layer to reduce the tensor size of the CNN output. The size of the latter layer is experimentally defined and is the largest possible based on the available GPU memory to minimize information loss. An example of the activated responses from the 256 channels

at layer *Convolution_3* is presented in Fig. 3, confirming that the extracted features are quite generic without being stable on any particular characteristics of the *Target* due to the minor depth variations of the $I_{k,k+1}$. A study on several CNN depths, number of channels, receptive filter sizes, and CNN-RNN bridge-layer configuration is presented in Section 3.4.

Table 2.
DRCNN configuration

Layer Nº	Network type	Layer type	Variables
1	CNN	Input	768x32 depth image
2		Convolution_1	filter size 5x5, padding 3, stride 2, channels 64
3		ReLU_1	-
4		Convolution_2	filter size 5x5, padding 2, stride 2, channels 128
5		ReLU_2	-
6		Convolution_3	filter size 5x5, padding 2, stride 2, channels 256
7		ReLU_3	-
8		Fully Connected_1	1024x1 matrix
9	RNN	LSTM_1	per LSTM layer: hidden values 1000, *tanh* activation function for cell and hidden states, *sigmoid* activation function for gates, mini Batch Size 10
10		LSTM_2	
11		LSTM_3	
12		Fully Connected_2	1024x1 matrix
13		Regression	1024x1 matrix converted to matrix R^*

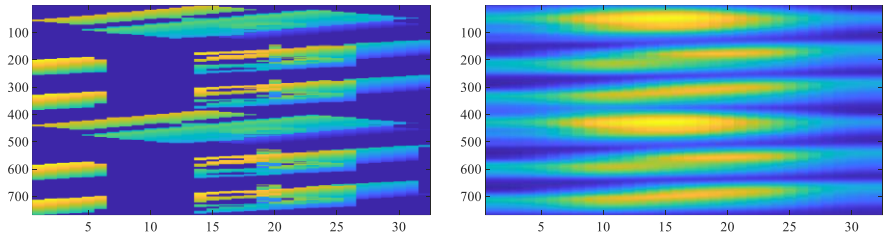


Fig. 3 CNN activation response (a) $I_{k,k+1}$ (b) response example at layer *Convolution_3*
(closer pixels/highest response respectively are hotter, best seen in color)

The RNN module aims at automatically modeling the *Source* – *Target* platform dynamics and the relations between the $I_{k,k+1}$ features extracted by the CNN module. It is worth noting that compared to handcrafted models used to describe motion and geometry, the RNN module is more flexible to learn the motion model. This is because RNN and specifically the Long Short-Term Memory (LSTM) layers used in our DRCNN architecture can learn long-term dependencies between image projections that exceed two sequential depth image projections, i.e. frame *k* and *k+1*. For an analysis on the operating principles of LSTM layers, the reader is referred to [51]. The proposed RNN module consists of three LSTM layers to enhance the RNN capability in learning a high-level representation and model complex dynamics. For completeness, it is worth mentioning that in Section 3.4 we evaluate several

LSTM layer depths, number of hidden values, and activation functions. However, due to GPU memory limitations, the maximum number of hidden values is limited to 1000 for three LSTM layers.

The three LSTM layers are then followed by the *Fully Connected_2* layer that has the same tensor cardinality as *Fully Connected_1*. Finally, the last layer of the RNN module is a *Regression* layer the output of which is converted into R^* by exploiting elements 1-12 so that,

$$regression\ layer = [r_{11}\ r_{12}\ r_{13}\ r_{21}\ r_{22}\ r_{23}\ r_{31}\ r_{32}\ r_{33}\ t_x\ t_y\ t_z]^T \quad (8)$$

while the remaining tensor elements 13-1024 are discarded. This is because the CNN output layer *Fully Connected_1*, along with the input and the output layers of the RNN module need to be of the same size, i.e. 1024x1, while the R^* is a 4x4 matrix containing the 12 rotational and translational values. As a reminder, the size of the *Fully Connected_1* layer is the largest possible to balance information loss and the memory limitations of our GPU. Finally, the configuration of the RNN module along with the parameters per layer are presented in Table 2 (layers 9-13), while the proposed DRCNN architecture for space robotics navigation is presented in Fig. 4. A study on several RNN depths, LSTM activation functions, and the number of LSTM hidden values is presented in Section 3.4.

The suggested DRCNN architecture is appealing for the following reasons:

a. Classic odometry involves local feature detection, matching, and then motion estimation based on the matched frame-to-frame features. The data domain used is usually 3D LIDAR, 2D visual, 2D IR, or a mixture of these. In DRCNN, we rely on a fully autonomous system that exploits the CNN module features and the robust navigation dynamics modeling of the RNN module, without involving any process of the typical odometry pipeline, i.e. feature detection and matching.

b. Compared to current deep learning odometry solutions for space robotics odometry [22], DRCNN exploits 3D LIDAR data rather than 2D visual imagery, affording the advantages of LIDAR over visual data as presented in Section 1. In contrast to the DRCNN proposed in this paper, space odometry literature in [22] purely relies on CNN and does not involve any type of RNN module. Additionally, the solutions presented in [52,53] propose deep learning odometry methods for terrestrial applications that combine CNN with RNN layers. However, these techniques exploit 2D visual data rather than 3D LIDAR data as done in this work and also suffer from the limitations of visual imagery, e.g. day operating conditions. An additional advantage of the proposed DRCNN over

the networks of [52,53] is that due to the larger information content per pixel of the visual imagery compared to the LIDAR projections exploited here, our CNN module is shallower empowering its generalization capability and ultimately affording DRCNN to be trained on simulated data and achieving appealing accuracy on real LIDAR data.

It should be noted that despite DRCNN attains an appealing odometry accuracy, it also has a few constraints that are mainly linked to the available GPU memory of our computer platform, which in turn impose limitations to the entire architecture and performance of DRCNN. Specifically, the weaknesses and sensitivities of DRCNN are:

a. the $I_{k,k+1}$ depth image size, which along with the quantization factor q_f determine the level of details within each point cloud projection. From our preliminary tests, we observed that the higher the level of details, the better the odometry accuracy is, but also a deeper CNN architecture is required to exploit the fine details. However, both these features impose larger GPU memory requirements.

b. The *FC_1* size, which links the CNN and the RNN modules. Inevitably due to memory constraints, we exploited the largest possible *FC_1* size by trading off accuracy because the layer's size is linked with the amount of information input to the LSTM module.

c. The hidden values within each LSTM layer are constrained to 1000, while based on our preliminary trials, increasing the number of hidden values may benefit the odometry accuracy.

d. The memory limitations presented above inevitably limit the DRCNN application to *Targets* with a known satellite model. This is because the selection of q_f is linked to the distinct topology information and the size of the *Target* model and thus cannot be generalized to facilitate a broad spectrum of satellites that contain different levels of details.

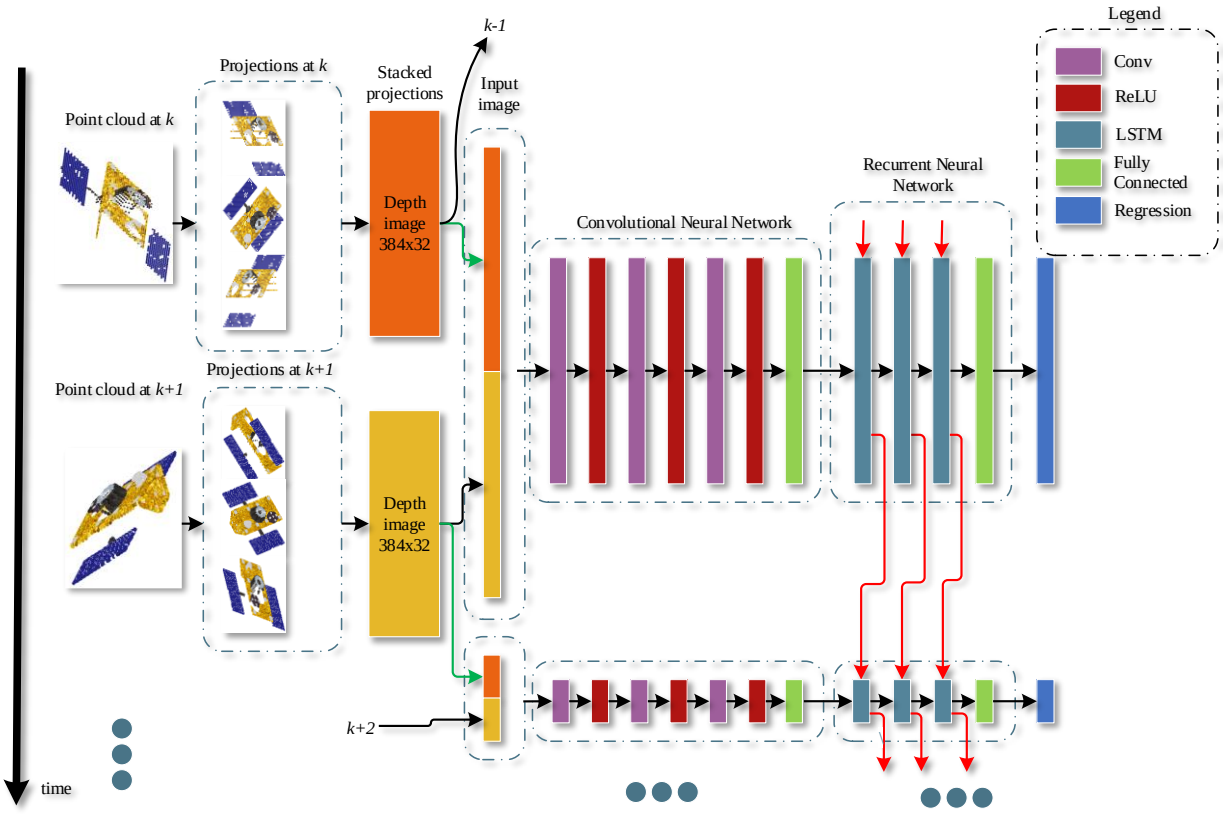


Fig. 4 Suggested DRCNN architecture

3. Experiments

Trials involve both simulated and real 3D LIDAR data of a non-cooperative but known *Target* satellite. Real LIDAR data (point cloud) of a satellite mockup are acquired in our lab by a LIDAR sensor placed on a *Source* platform that is moving within the lab.

3.1. Experimental setup

Prior to evaluating DRCNN, we train it using the Elite satellite *Target* platform developed by Thales Alenia Space (France) that is inspired by the Globalstar-2 and Iridium constellations. The Elite *Target* satellite is a complete 3D point cloud model $\boldsymbol{P_{3D}}$ from which we create self-occluded point cloud views emulating realistic views of the *Target* platform as observed by the virtual motion of the *Source* platform. These views are created by applying the Hidden Point Removal (HPR) [54] algorithm on the *Target* platform.

HPR initially remaps the coordinates of each vertex p_{3D} belonging to $\boldsymbol{P_{3D}}$ by exploiting an imaginary ray that connects each p_{3D} vertex with the viewpoint set on the LIDAR sensor onboard the *Source*. The output of this stage is a mirror image of $\boldsymbol{P_{3D}}$ as observed from the *Source*, which is then projected onto a sphere of radius *R* that is centered at the LIDAR sensor. The latter process is named "spherical flipping" and the resulting point cloud consists of the P_{sfq} vertices:

$$p_{sfq} = p_{3D} + 2(R - \|p_{3D}\|)\frac{p_{3D}}{\|p_{3D}\|} \qquad (9)$$

In out trials *R* is automatically calculated as suggested in [55]. Finally, the convex hull of the resulting point cloud, associated with a weight factor a_q for each point of the cloud is given by:

$$\left\{\sum_{q=1}^{|P_{sfq}|} a_q p_{sfq} \,|\, (\forall q : a_q \geq 0) \wedge \sum_{q=1}^{|P_{sfq}|} a_q = 1)\right\} \qquad (10)$$

Summarizing, a vertex p_{3D} of the raw *Target* point cloud $\boldsymbol{P_{3D}}$ is considered as visible and belongs to $\boldsymbol{P_k}$, only if its spherical flipped form P_{sfq} is on the convex hull. In our trials, depending on the *Source* – *Target* relative position and distance, the cardinality of $\boldsymbol{P_k}$ varies from 8000 vertices down to only 30 vertices.

Once $\boldsymbol{P_k}$ is created, we train the proposed DRCNN to link the response of the deepest CNN layer with the relative *Source* – *Target* platform position, by utilizing three LSTM layers. For that purpose, the Elite satellite *Target* platform performs a simulated sinusoidal trajectory with $y = 2\sin(x), z = x$ with $\{x = 0.2g \,|\, g \in \{1,...,25000\}\}$. This trajectory has the advantage of repeatedly training our network on combined curved and straight trajectories. For the training process we use an initial learning rate of 0.01 with a gradient threshold of 1. As a reminder, the size of $I_{k,k+1}$ and the cardinality of the fully connected layers in the DRCNN are governed by the GPU memory limitations of our computer platform and are experimentally defined during the training stage. Finally, it should be noted that once DRCNN is trained on the sinusoidal trajectory it is then evaluated on both simulated and real LIDAR data scenarios. As a reminder DRCNN is trained to meet the SO(3) constraints because the ground truth *R* is free of these constraints (see Section 2).

3.1.1 Trials on simulated data

The first batch of trials considers three trajectories of the Elite satellite *Target* platform that we also use during the DRCNN's training stage. For our simulated data trials, we consider three scenarios, namely a straight-line approach

(*SLA*), an ellipse of inspection (*EoI*), and a helical (*Helix*). Similar to the training process, for all three scenarios we create self-occluded point cloud views emulating realistic views of the *Target* platform by applying the HPR algorithm. An example of the Elite *Target* platform along with the simulated ground truth trajectories are presented in Fig. 5 (a)-(d). It should be noted that for the simulated data scenarios we intentionally extend the *Source* – *Target* ranges beyond the standard ones to push the limits of odometry and investigate the performance of the evaluated methods. The odometry accuracy is compared against the fictitious ground truth position of the *Source* platform utilizing the metrics presented in Section 3.1.3.

3.1.2 Trials on real LIDAR data

The next set of trials considers real data acquired by a Velodyne VLP-16 Puck Lite LIDAR sensor. Trials evaluate the proposed space robotics odometry architecture on several scenarios where the LIDAR sensor is placed on a moving *Source* platform, in relation to a scaled *Target* EnviSat satellite model. The odometry accuracy is evaluated against the ground truth position of the *Source* that is determined from an Optitrack setup [56] that tracks the *Source* platform within our lab. Optitrack provides the position of objects that are within its field of view and are visible in the Near Infrared (NIR) bandwidth in sub-millimeter accuracy. Thus, we place highly NIR reflective markers on the VLP-16. The accuracy of the pose solution used as ground truth is 10^{-3} m.

In our trials we consider three trajectories, namely a Forward-backward (*FB*), a *FB curved*, and a single leg curved (*Curved*). On average, the point cloud cardinality of $\boldsymbol{P}_k$ is 190 vertices. Fig. 2 (e)-(i) present the *Target* platform, an example of the acquired $\boldsymbol{P}_k$ and the ground truth trajectories of the real data scenarios. As a reminder, despite this set of scenarios involves real LIDAR data, the proposed DRCNN architecture is still trained on the simulated sinusoidal trajectory presented in Section 3.1.1.

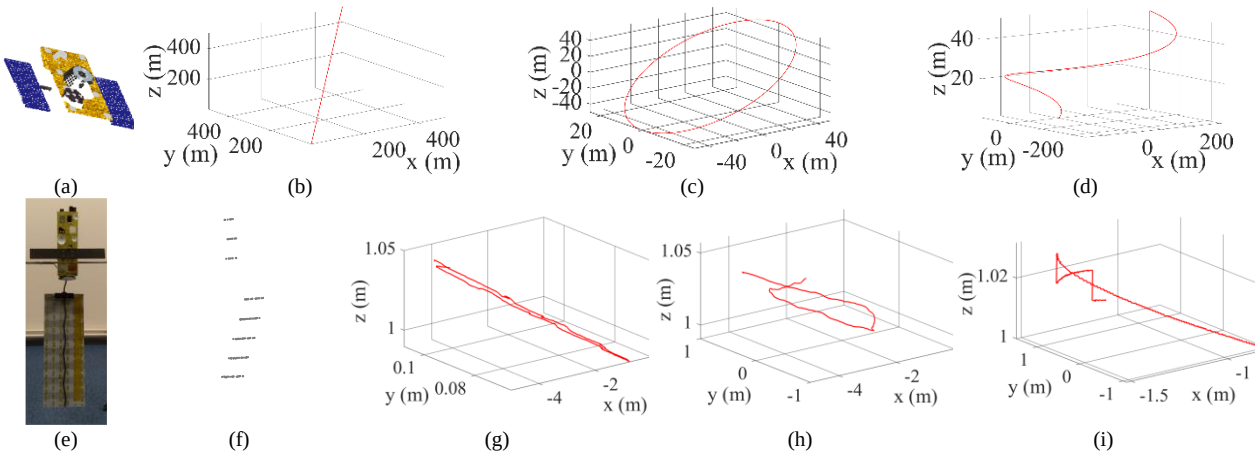


Fig. 5 (a) *Target* point cloud and evaluated simulated trajectories (b) *SLA* (c) *Eo*I (d) *Helix* where motion is from low towards high z-axis values – (e) Target mockup (f) Target point cloud and evaluated real trajectories (g) *FB* (h) *FB curved* (i) *Curved*

3.1.3 Evaluation criteria

We challenge the suggested DRCNN architecture against current space navigation techniques and specifically against ICP [9,16,17,30,31], HoD-S combined with adaptive H∞ recursive filtering [18], OUR-CVFH combined with ICP [10,14,32], and Spin Images combined with ICP [32]. For the latter two cases, we apply the OUR-CVFH/ICP and Spin Images/ICP on a frame-to-frame basis. Similarly, it should be noted that in our trials the ICP algorithm is used to perform odometry rather than *Target* tracking, and thus at instance $k+1$ ICP does not initialize with the pose estimated at instance k. Additionally, the parameters of the proposed architecture and the competitor methods are tuned for optimum odometry based on the sinusoidal trajectory used to train DRCNN. Table 3 presents the tuned parameters, while the parameters not tuned are fixed either to the ones originally proposed by their authors or to their PCL implementation [57–59]. In our trials we consider that the initial position and pose R^* of the *Source* is known and that all the evaluated 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 [21]. Despite several more space odometry architectures exist (see Table 1), in most cases their implementation is not available and re-implementing these methods might lead to a non-optimal performing solution. Additionally, simply utilizing odometry methods that were initially designed for terrestrial applications is also a non-optimum solution as the projected LIDAR imagery and the RGB imagery have a large domain gap. Hence, for fairness, we constrain our evaluation to the competitor methods with an available code, which are presented above. For completeness, it is also

worth mentioning that the design of DRCNN is constraint to the data domain and level of details presented here, i.e. LIDAR imagery projected on the planes of a local reference frame. Thus, simply applying DRCNN on the LIDAR data of the popular Kitti dataset [60], would lead to a non-optimum solution as the finer features of that dataset require high-resolution depth images $I_{k,k+1}$ and deeper CNNs exceeding the three convolutional layers of our DRCNN. However, redesigning and tuning the DRCNN to meet these requirements is another research topic that is beyond the scope of this paper.

In our trials performance is evaluated in terms of drift, i.e. RMSE between the estimated endpoint and the ground truth (GT) endpoint, T_{error} presenting the overall translational error as a percentage over the GT distance traveled, e^T representing RMSE of the true and the estimated position, and finally t the processing time required per frame. All trials are implemented on a desktop with an Intel i7, an NVIDIA Quadro K2200 GPU, and 16GB of RAM, running Windows 10 and MATLAB 2019a. For completeness, it is worth mentioning that real space platforms use space-grade field-programmable gate arrays (FPGA), however in the context of evaluating the conceptual validity and performance of DRCNN against current space robotics odometry methods, we believe that the computer platform used during trials is acceptable.

Table 3.
Tuned parameters

Category	Tuned parameters
OUR-CVFH	5° angular threshold, curvature threshold 1, axis ratio 0.8
Spin Images	description radius 0.02, 8 resolution bins
ICP	point-to-point variant, 1% translational tolerance in consecutive iterations, 1000 iterations
HoD-S	description radius 20 x average $\boldsymbol{P}_{k+1}$ resolution, encoding quality 10 bins
adaptive H∞ recursive filtering	$dt = 10^{-5}$ and $g = 0.1$ parameters of H∞, iterations equal to the HoD-S matches cardinality

3.2. Simulated data odometry trials

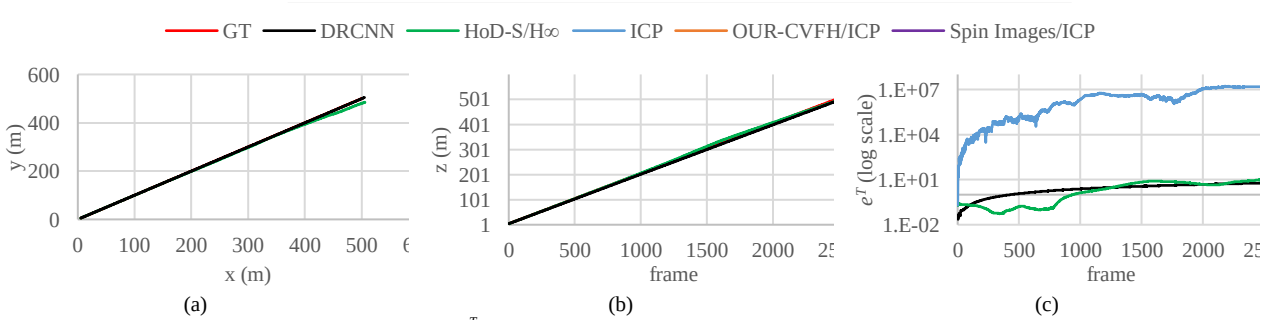
3.2.1 SLA scenario

This is a constant *Target* pose scenario where the *Source –Target* range is increasing. From Table 4 it is evident that the suggested architecture is overall more accurate than the competitor techniques challenged, while the processing burden is only 60ms. Next to follow is HoD-S/H∞, while ICP does not present an appealing odometry

solution despite the frame-to-frame motion being relatively small. This is mainly due to the sparse nature of $\boldsymbol{P_k}$ and $\boldsymbol{P_{k+1}}$ prohibiting ICP to settle to a globally optimum solution and due to implementing ICP on a frame-to-frame basis without initializing ICP at frame $k+1$ with the pose estimated at instance k. Regarding the Spin Image/ICP and OUR-CVFH/ICP solutions, both fail to present a valid odometry solution because Spin Images and OUR CVFH do not afford any correct feature matches between $\boldsymbol{P_k}$ and $\boldsymbol{P_{k+1}}$, mainly due to the sparse nature of the *Target* point cloud. For OUR-CVFH specifically, the lack of distinctive vertex clusters on the *Target* point cloud automatically degrades the high performing OUR-CVFH feature estimation to the VFH features that are typically less robust. Hence, Spin Image/ICP and OUR-CVFH/ICP preserve during each frame-to-frame motion the R^* initialization value, i.e. unity rotation matrix and zero translation matrix. Therefore, the performance metrics of these two methods are omitted from Table 4.

Considering the processing burden imposed by each method, DRCNN is the fastest to execute despite being a two-phase method, i.e. 3D to multi-2D point remapping and activating the DRCNN. This is due to the space environment's lack of background and the small-sized *Targets*, making the remapping process highly efficient requiring less than a millisecond. The activation of the DRCNN is also fast executing within a few milliseconds. Among the methods offering a valid odometry solution, HoD-S/H∞ is the least processing efficient as it involves 3D data manipulation, which is well known for imposing a higher processing burden.

Fig. 6 presents the trajectory plots of DRCNN against HoD-S/H∞ and the ground truth (GT). For better readability and due to their large error, we eliminate the presentation of the trajectory plots of ICP, Spin Images/ICP, and OUR-CVFH/ICP.



(a) (b) (c)

Fig. 6 *SLA* scenario (a) XY (b) z and error plots (c) e^T. Due to large errors and for better readability, Spin Image/ICP, OUR CVFH/ICP and ICP are neglected from (a)-(c)

Table 4.
Performance Metrics for the *SLA* Scenario

	drift (m)	T_{error} (%)	e^T	t (s)
DRCNN	10.29	0.69	2.97	0.06
HoD-S/H∞	17.66	1.19	3.44	0.20
ICP	10^7	10^6	4×10^6	0.13

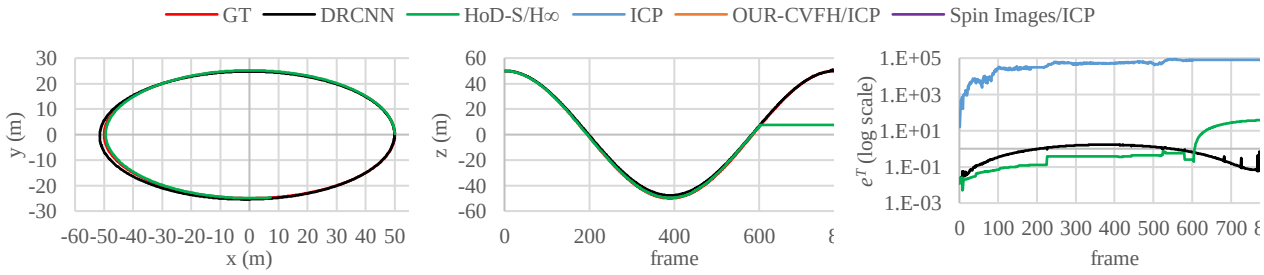
3.2.2 EoI scenario

This scenario considers a simultaneously frame-to-frame varying *Target* pose and *Source* – *Target* distance, emulating the *Source* platform orbiting around the *Target* platform. Table 5 presents the performance metrics of the evaluated methods and Fig. 7 shows the trajectory plots of DRCNN against HoD-S/H∞ and GT.

This scenario is more challenging because in addition to the sparse *Target* point cloud, the trajectory is curved and altitude (z-axis) also varies. Despite that, the accuracy of the proposed architecture affords overall low errors and a low processing time. It should be noted that despite this scenario is more challenging compared to the *SLA* scenario, the performance of DRCNN is better because the trajectory is shorter and thus errors do not build up. However, despite the relatively short trajectory, ICP still presents large errors, while Spin Images/ICP and OUR CVFH/ICP also fail to offer correct feature matches for relative motion estimation. An analysis for these methods' failure to offer a valid space odometry solution is presented in the *SLA* scenario. Despite HoD-S/H∞ and ICP being the fastest to execute, DRCNN is overall more appealing due to the small e^T error it attains and its low computational requirements.

Table 5.
Performance Metrics for the *EoI* Scenario

	drift (m)	T_{error} (%)	e^T	t (s)
DRCNN	0.14	0.03	0.84	0.06
HoD-S/H∞	65.13	21.24	5.42	0.01
ICP	1.5×10^4	3×10^4	5×10^4	0.01



(a) (b) (c)

Fig. 7 *EoI* scenario (a) XY (b) z and error plots (c) e^T. Due to large errors and for better readability, Spin Image/ICP, OUR CVFH/ICP and ICP are neglected from (a)-(c)

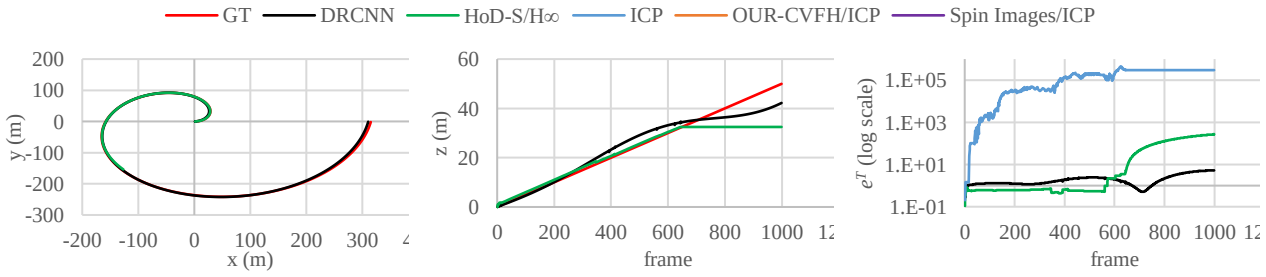
3.2.3 Helix scenario

This is the most challenging scenario evaluated in this work 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 cardinality. Despite that, we intentionally push the limits of odometry requirements and investigate the performance of the evaluated methods. Table 6 presents the performance metrics, while Fig. 8 shows the corresponding trajectories. From Table 6 it is evident that the performance hierarchy of all methods is preserved with the suggested DRCNN method still offering an appealing solution.

Interestingly, from Fig. 8 we observe that HoD-S/H∞ from frame 651 onwards presents an unexpected behavior with large errors. This is because from that frame till the end of this trajectory, the relative *Source* – *Target* position and distance are such that they prohibit HoD-S from achieving feature matches and thus, the R^* remains at its initialization value, i.e. unity rotation matrix and zero translation matrix.

Table 6.
Performance Metrics for the *Helix* Scenario

	drift (m)	T_{error} (%)	e^T	t (s)
DRCNN	9.04	0.64	2.03	0.58
HoD-S/H∞	468.26	33.00	51.35	0.78
ICP	$5x10^5$	$3x10^4$	$1.7x10^5$	0.11



(a) (b) (c)

Fig. 8 *Helix* scenario (a) XY (b) z and error plots (c) e^T. Due to large errors and for better readability, Spin Image/ICP, OUR CVFH/ICP and ICP are neglected from (a)-(c)

3.3. Real data odometry trials

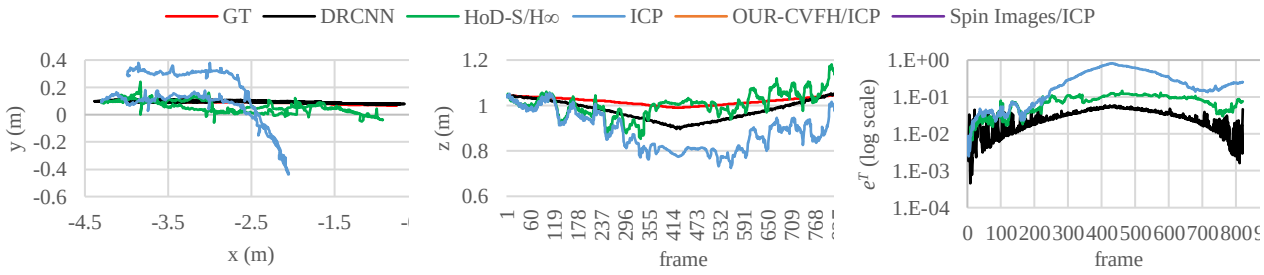
Compared to the simulated scenarios of Section 3.2, the three real data scenarios evaluated in this section are more challenging due to the highly sparse $\boldsymbol{P_k}$, the limited structure of the EnviSat *Target* point cloud (Fig. 5 (f)), and

most importantly, due to the different data domain of the training and testing data, i.e. simulated vs. real LIDAR data.

3.3.1 FB scenario

This scenario considers a straight-line Forward – Backward motion of the *Source* with respect to the *Target*, where point clouds are acquired by a LIDAR sensor that is placed on the *Source* platform. The performance metrics attained by each method are presented in Table 7, while the corresponding trajectory and error plots in Fig. 9. From the results presented it is evident that DRCNN affords the lowest e^T errors and a low processing burden making it a very appealing solution, despite it is trained solely on simulated data. It should be noted that similar to the trials relying on simulated data (Section 3.2), Spin Images/ICP and OUR-CVFH/ICP do not attain valid feature matches forcing R^* to preserve its initialization values. Hence, in the performance metrics of Table 7 and Fig. 9 we neglect presenting these two methods.

In contrast to the simulated data scenarios, on the FB trajectory DRCNN, HoD-S/H∞, and ICP have notably better accuracy. This is because the *Source* platform is moving slow, and thus $\boldsymbol{P_k}$ and $\boldsymbol{P_{k+1}}$ present a smaller frame-to-frame variation. Regarding ICP, despite being more accurate compared to the simulated scenarios, it still attains larger translational errors compared to DRCNN and HoD-S/H∞. It is worth noting that the ICP's performance on the *FB* trajectory is in line with [18] because in this work we tune ICP according to Table 3.



(a) (b) (c)

Fig. 9 *FB* scenario (a) XY (b) z and error plots (c) e^T. Due to large errors and for better readability, Spin Image/ICP, OUR CVFH/ICP and ICP are neglected from (a)-(c)

Table 7.
Performance Metrics for the *FB* Scenario

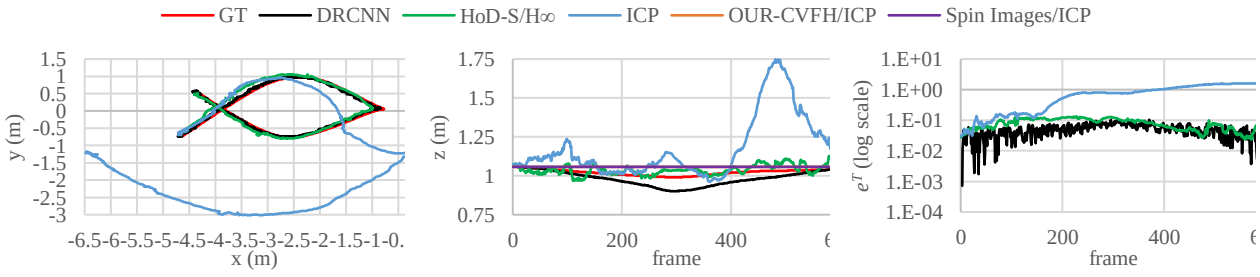
	drift (m)	T_{error} (%)	e^T	t (s)
DRCNN	0.08	1.08	0.03	0.06
HoD-S/H∞	0.12	1.65	0.08	0.24
ICP	0.44	5.91	0.29	0.05

3.3.2 FB curved scenario

This is a highly curved real data trajectory that is more challenging compared to the *FB* scenario. Despite that, DRCNN and HoD-S/H∞ achieve low e^T errors with DRCNN requiring less processing time. Similarly to the *FB* trajectory, Spin Images/ICP and OUR-CVFH/ICP fail to present feature matches and thus provide a poor odometry performance. Table 8 presents the performance metrics and Fig. 10 the corresponding trajectories of DRCNN, HoD-S/H∞, and ICP. Comparing Table 8 with the corresponding tables of the previous scenarios we observe that the performance hierarchy of the evaluated methods is still preserved and that the suggested DRCNN method remains an appealing solution.

3.3.3 Curved scenario

Similar to the previous scenarios exploiting real LIDAR data, in this trial DRCNN manages a low translational error with a highly appealing computational efficiency. Despite the e^T error of DRCNN not being optimum, it is still close to that of the competitor methods. Performance metrics are presented in Table 9, while Fig. 11 presents the corresponding trajectory plots.

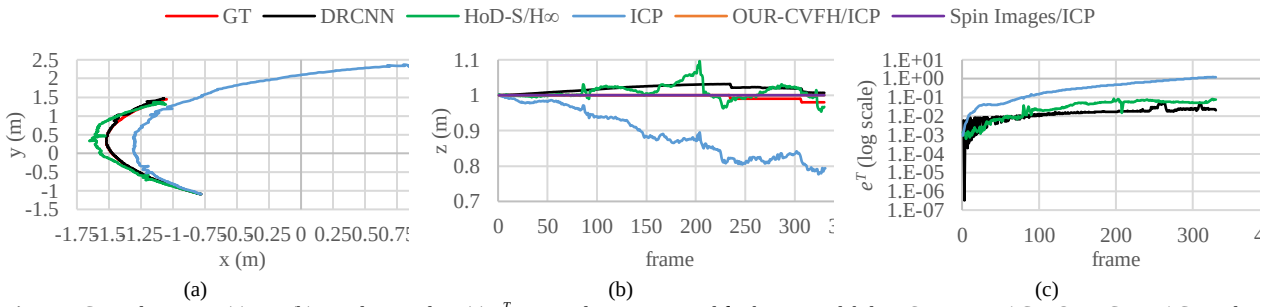


(a) (b) (c)

Fig. 10 *FB Curved* scenario (a) XY (b) z and error plots (c) e^T. Due to large errors and for better readability, Spin Image/ICP, OUR CVFH/ICP and ICP are neglected from (a)-(c)

Table 8.
Performance Metrics for the *FB Curved* Scenario

	drift (m)	T_{error} (%)	e^T	t (s)
DRCNN	0.03	0.24	0.05	0.06
HoD-S/H∞	0.13	1.05	0.08	0.23
ICP	2.70	21.79	0.80	0.06



(a) (b) (c)

Fig. 11 *Curved* scenario (a) XY (b) z and error plots (c) e^T. Due to large errors and for better readability, Spin Image/ICP, OUR CVFH/ICP and ICP are neglected from (a)-(c)

Table 9.
Performance Metrics for the *Curved* Scenario

	drift (m)	T_{error} (%)	e^T	t (s)
DRCNN	0.04	0.96	0.02	0.06
HoD-S/H∞	0.13	3.48	0.04	0.36
ICP	2.09	55.57	0.43	0.06

3.4. DRCNN variations

For completeness, we also evaluate various DRCNN configurations to highlight the contribution of each of the DRCNN's parameters to the odometry solution. Specifically, given the core DRCNN architecture presented in Table 2, in the following trials we modify one of DRCNN's parameters while the remaining ones are preserved. The evaluated parameters are the $I_{k,k+1}$ input size and interpolation scheme, CNN depth, number of channels and filter size per convolutional layer, RNN depth, LSTM number of hidden values and activation function, and finally several CNN-RNN bridge layer configurations. To keep the paper in a reasonable length, evaluation is performed on the *SLA* scenario using the T_{error} metric.

The first trial evaluates the performance of DRCNN under several fixed 1:1 and variable height/width ratio input image sizes of I_k. Evaluation involves sizes up to 128pixels per dimension, which is a limiting factor of our GPU memory, with Fig. 12 presenting the corresponding performance. The latter figure shows that the lowest T_{error} attained for the 1:1 case is at 32x32 pixels I_k image size (4.53%), while for the variable height/width ratio case at 128x32 pixels (0.69%). For the fixed 1:1 ratio case, despite increasing the I_k size the odometry performance does not improve mainly due to the quantization factor ($q_f = 20$), and the receptive filter size within the CNN module (5x5 pixels), which all three parameters have a strong interaction on the DRCNN's performance. For the variable height/width ratio case, the optimum image size is the 128x32 pixels, revealing that indeed the 32pixels width is

well linked to the q_f and receptive filter size selected. This is also evident from the 64x32pixels size, which attains a T_{error} value close to the average of the 32x32 and 64x64pixels sizes. On that basis, it would be interesting to evaluate the 128x128pixels case to reveal if indeed the 128pixels would further benefit the DRCNN's odometry. Though our GPU memory limits our trials up to 64x64pixels and 128x32pixels for the 1:1 and the variable height/width ratio cases, respectively.

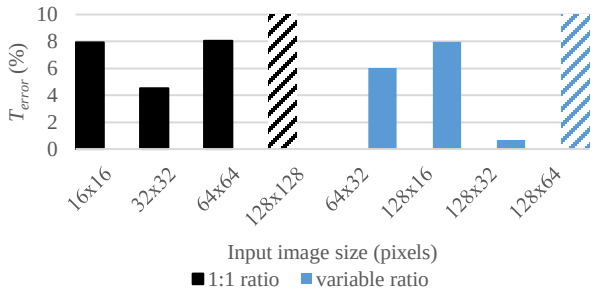


Fig. 12 DRCNN performance for various input image sizes and height/width ratio
(dashed column height is for illustration purposes showing GPU memory-limited cases)

One of the important parameters affecting the details in I_k is the interpolation method applied to the quantized depth images while resizing them to obtain I_k. Thus, we evaluate three interpolation schemes, namely the nearest neighbor, the bilinear, and the bicubic. The first one is the simplest technique where the output pixel in the interpolated image is assigned the value of the pixel that the point falls within. For the bilinear interpolation, the output pixel value is a weighted average of pixels in the nearest 2x2 neighborhood, while the bicubic interpolation is an extension of the bilinear involving the nearest 4x4 neighborhood. Depending on the application, each method has its own strengths. For the odometry scenarios examined in this work, we observed that preserving the depth variations in the I_k is crucial as these trigger the responses within each DRCNN layer. Indeed, from Table 10 it is evident that as I_k becomes smoother the T_{error} metric is increasing. In terms of processing time, all methods are of the same order.

Table 10.
DRCNN performance for various interpolation schemes applied on I_k

	T_{error} (%)	t (s)
nearest	0.69	2
bilinear	1.54	3
bicubic	3.64	3

A major contributor in DRCNN is the CNN module. The main parameters affecting the CNN's performance are its depth, the number of channels in each convolutional layer, and the receptive filter size used for each convolution. The first batch of trials considers the DRCNN configurations presented in Table 11, where the CNN depth ranges from three layers up to 11 layers with the convolution channels altering from 64 up to 1024. Fig. 13 highlights that extending the CNN's depth does not improve the odometry accuracy of the DRCNN (network configurations *A*-*E* of Table 11). This is due to the coarse level of details in I_k, which are capable of providing generic features and thus the fine details extracted by the deeper layers (exceeding the three convolutional layers) are reducing the overall DRCNN's performance. This does not contrast the mainstream CNN approach for RGB imagery where commonly deeper architectures are more robust because the information content per pixel in the visual imagery is larger compared to the LIDAR projections exploited in this work. Considering the number of channels per convolutional layer, from Fig. 13 it is obvious that preserving the same number of channels for all convolution layers is not an optimum choice (network configurations *F*-*I* of Table 11). This is because as the CNN depth increases, the features extracted by each convolutional layer become finer, and thus more channels are required to learn these features [52]. Accordingly, setting a large channel size for the entire CNN structure is also not optimum as the channel size is not matched to the level of details of the extracted features. Overall, the optimum CNN depth and channel size combination is found at the network configuration *C* of Table 11. However, given that the level of details in I_k is low, evaluating the network configuration *B* could potentially provide an appealing T_{error} metric. Though, this shall pose future work as currently for our computer platform this is a GPU memory-limited case.

Table 11.
Various DRCNN configurations by altering the CNN depth and channels, one per column, presented as; Fully Connected layers "FC-(number of units)", Convolutional layers "Conv-(number of feature maps)@ filter size", LSTM layers "LSTM-(number of hidden values)"

	Network configuration								
	A	B	C	D	E	F	G	H	I
CNN module	Conv-64 @5x5 +ReLU	Conv-64 @5x5 +ReLU Conv-128 @5x5 +ReLU	Conv-64 @5x5 +ReLU Conv-128 @5x5 +ReLU Conv-256 @5x5 +ReLU	Conv-64 @5x5 +ReLU Conv-128 @5x5 +ReLU Conv-256 @5x5 +ReLU Conv-512 @5x5 +ReLU	Conv-64 @5x5 +ReLU Conv-128 @5x5 +ReLU Conv-256 @5x5 +ReLU Conv-512 @5x5 +ReLU Conv-1024 @5x5 +ReLU	Conv-64 @5x5 +ReLU Conv-64 @5x5 +ReLU Conv-64 @5x5 +ReLU	Conv-128 @5x5 +ReLU Conv-128 @5x5 +ReLU Conv-128 @5x5 +ReLU	Conv-256 @5x5 +ReLU Conv-256 @5x5 +ReLU Conv-256 @5x5 +ReLU	Conv-512 @5x5 +ReLU Conv-512 @5x5 +ReLU Conv-512 @5x5 +ReLU
	FC-1024	FC-1024	FC-1024	FC-1024	FC-1024	FC-1024	FC-1024	FC-1024	FC-1024
RNN module	LSTM-1000 LSTM-1000 LSTM-1000	LSTM-1000 LSTM-1000 LSTM-1000	LSTM-1000 LSTM-1000 LSTM-1000	LSTM-1000 LSTM-1000 LSTM-1000	LSTM-1000 LSTM-1000 LSTM-1000	LSTM-1000 LSTM-1000 LSTM-1000	LSTM-1000 LSTM-1000 LSTM-1000	LSTM-1000 LSTM-1000 LSTM-1000	LSTM-1000 LSTM-1000 LSTM-1000

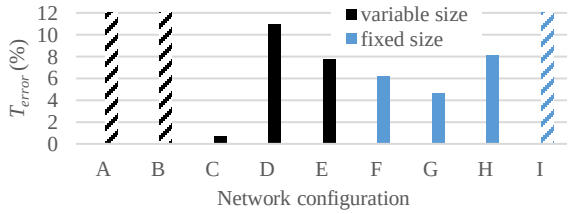


Fig. 13 DRCNN performance for various CNN configurations (Table 11) altering layer depths and channel size per convolutional layer (dashed column height is for illustration purposes showing GPU memory-limited cases)

The next trial investigates the interplay between varying the receptive filter size and the overall DRCNN performance. The network configurations evaluated are presented in Table 12, with the corresponding DRCNN T_{error} presented in Fig. 14. The latter figure highlights that, as expected, the filter size has to be matched with the size of the extracted features. Hence, a fixed size of 5x5 or 7x7 is appealing attaining 0.69% and 2.88% T_{error}, respectively. We also evaluate variable sized-filters that either gradually increase or reduce size. Results in Fig. 14 confirm that indeed the extracted features are large and thus the 3x3/5x5/7x7 filter pattern is more appealing than its reversed variant.

Table 12.
Various DRCNN configurations by altering the receptive filter size, one per column, presented as; Fully Connected layers "FC-(number of units)", Convolutional layers "Conv-(number of feature maps)@ filter size", LSTM layers "LSTM-(number of hidden values)"

	Network configuration					
CNN module	Conv-64 @3x3+ReLU Conv-128 @3x3+ReLU Conv-256 @3x3+ReLU FC-1024	Conv-64 @5x5+ReLU Conv-128 @5x5+ReLU Conv-256 @5x5+ReLU FC-1024	Conv-64 @7x7+ReLU Conv-128 @7x7+ReLU Conv-256 @7x7+ReLU FC-1024	Conv-64 @9x9+ReLU Conv-128 @9x9+ReLU Conv-256 @9x9+ReLU FC-1024	Conv-64 @3x3+ReLU Conv-128 @5x5+ReLU Conv-256 @7x7+ReLU FC-1024	Conv-64 @7x7+ReLU Conv-128 @5x5+ReLU Conv-256 @3x3+ReLU FC-1024
RNN module	LSTM-1000 LSTM-1000 LSTM-1000 FC-1024	LSTM-1000 LSTM-1000 LSTM-1000 FC-1024	LSTM-1000 LSTM-1000 LSTM-1000 FC-1024	LSTM-1000 LSTM-1000 LSTM-1000 FC-1024	LSTM-1000 LSTM-1000 LSTM-1000 FC-1024	LSTM-1000 LSTM-1000 LSTM-1000 FC-1024

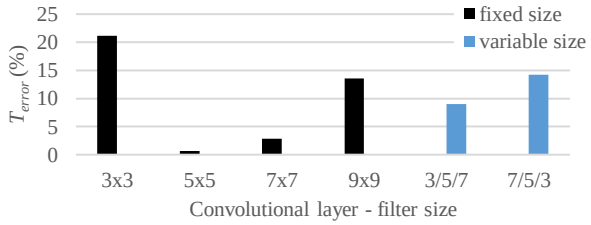


Fig. 14 DRCNN performance for various receptive filter sizes
(3/5/7 and 7/5/3 refer to a changing receptive filter size per convolutional layer with a pattern of 3x3/5x5/7x7 and 7x7/5x5/3x3 respectively)

Another important contributor in DRCNN is the layer that bridges the CNN and the RNN modules. As a reminder, Eq. (1) and Eq. (3) require a 4x4 R^* transformation matrix, and thus a Fully Connected (FC) layer in the form of a 16x1 matrix is the minimum bridge-layer size. In the following trials we evaluate several FC configurations including various FC matrix sizes (in the form of a column matrix) and multiple FC layers (Table 13). It should be noted that altering the FC size is affecting the RNN's input and output, with the excessive FC elements being discarded (elements 17 up to the matrix cardinality). The first batch of trials involves a single FC layer of various sizes. Fig. 15 shows that indeed selecting an FC-16 layer, i.e. 16x1 matrix, is appealing, but still this is not the optimum choice (5.67% T_{error}). Increasing the FC layer size elements and exploiting the first 16 entries can be beneficial for an FC layer size of at least1024. Further increasing the FC size is not possible due to GPU memory limitations, but following the trend in Fig.15, the FC-2048 case can be appealing. We also evaluate the multi-layer FC case by investigating the effectiveness of the FC-1024/16 and FC-1024/512/16 cases, where each number refers to the matrix size per FC layer. Despite the FC-1024/16 not being the optimum configuration, it still stands between the performance of the two individual layer cases with 4.10% T_{error}, i.e. FC-1024 attains 0.69 T_{error} and FC-16 5.67% T_{error}, indicating that indeed the FC-1024 setup case encodes well the

R^* of Eq. (1). Unexpectedly, further increasing the bridge-layer size and combining the top-3 performing FC layers, i.e. FC-1024/512/216, does not provide performance close to the average value of the involved FC layers. For completeness, the discarded excessive FC elements from 17 up to the matrix cardinality were close to zero and did not provide any useful information. Overall, a deeper explanation of the influence of the bridge-layer to the DRCNN performance is not obvious, enhancing the necessity of further progress in the Explainable Artificial Intelligence (XAI) domain.

Table 13.
Various DRCNN configurations by altering the Fully Connected bridge-layer setup between the CNN and the RNN modules, one per column, presented as; Fully Connected layers "FC-(number of units)", Convolutional layers "Conv-(number of feature maps)@ filter size", LSTM layers "LSTM-(number of hidden values)"

	Network configuration							
CNN module	Conv-64 @5x5+ReLU Conv-128 @5x5+ReLU Conv-256 @5x5+ReLU FC-16	Conv-64 @5x5+ReLU Conv-128 @5x5+ReLU Conv-256 @5x5+ReLU FC-64	Conv-64 @5x5+ReLU Conv-128 @5x5+ReLU Conv-256 @5x5+ReLU FC-256	Conv-64 @5x5+ReLU Conv-128 @5x5+ReLU Conv-256 @5x5+ReLU FC-512	Conv-64 @5x5+ReLU Conv-128 @5x5+ReLU Conv-256 @5x5+ReLU FC-1024	Conv-64 @5x5+ReLU Conv-128 @5x5+ReLU Conv-256 @5x5+ReLU FC-2048	Conv-64 @5x5+ReLU Conv-128 @5x5+ReLU Conv-256 @5x5+ReLU FC-1024 FC-16	Conv-64 @5x5+ReLU Conv-128 @5x5+ReLU Conv-256 @5x5+ReLU FC-1024 FC-512 FC-16
RNN module	LSTM-1000 LSTM-1000 LSTM-1000 FC-16	LSTM-1000 LSTM-1000 LSTM-1000 FC-64	LSTM-1000 LSTM-1000 LSTM-1000 FC-256	LSTM-1000 LSTM-1000 LSTM-1000 FC-512	LSTM-1000 LSTM-1000 LSTM-1000 FC-1024	LSTM-1000 LSTM-1000 LSTM-1000 FC-2048	LSTM-1000 LSTM-1000 LSTM-1000 FC-16	LSTM-1000 LSTM-1000 LSTM-1000 FC-16

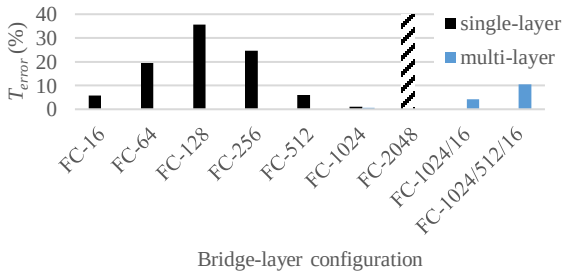


Fig. 15 DRCNN performance for various CNN-RNN bridge-layer configurations, FC-1024/16 and FC-1024/512/16 refer to multi-layered configurations with a pattern of FC-1024 layer succeeded by an FC-16 layer and succeeded by an FC-512 followed by an FC-16 layer respectively. (dashed column height is for illustration purposes showing GPU memory-limited cases)

The final batch of our trials involves modifying the RNN module parameters. Hence, we alter the RNN depth by evaluating several LSTM layer depths ranging from one up to eight, and for each LSTM layer we also vary the number of hidden values ranging from 100 up to 1000. The performance of the RNN variants is presented in Fig. 16. A common feature is that all RNN configurations attain a global minimum T_{error} and thereafter present a minor fluctuation at higher T_{error} values. Overall, increasing the LSTM layers improves performance, while for more than six LSTM layers the T_{error} metric stabilizes indicating that for our image projections $I_{k,k+1}$ a shallow to medium

depth is sufficient. Considering the number of hidden values, these do not considerably contribute to the DRCNN performance, especially for up to two or more than six LSTM layers. For the intermediate two-to-five LSTM layers case, a minor performance variation is observable with the three LSTM-1000 configuration network being the optimum choice and the five LSTM-100 structure closely following. Again, a clear explanation of the interplay between the number of hidden values in each LSTM layer and the DRCNN performance is not obvious and further study in the XAI domain can be enlightening.

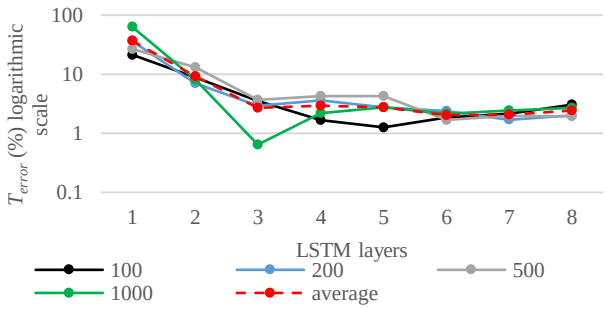


Fig. 16 Various RNN depth and LSTM hidden values configurations

Finally, we evaluate the interplay between the T_{error} metric and the LSTM's activation function for the cells and hidden states, and the gates. As expected, from Table 14 it is obvious that setting the *sigmoid* as the activation function for the gates presents on average lower odometry errors. Additionally, choosing the *softsign* activation function for the cells and hidden states also affords lower T_{error} values. However, from Table 14 it is clear that the combination of the latter two activation functions is not optimum highlighting that the internal operations within the LSTM are still an open case for further study.

Table 14.
DRCNN performance T_{error} (%) for various LSTM activation functions

		Cell and hidden states		
		tanh	softsign	average
Gates	sigmoid	0.69	3.12	1.90
	hard-sigmoid	13.08	1.65	7.36
	average	6.88	2.38	

3.5. Discussion

From the trials performed in Sections 3.2 and 3.3, it is evident that DRCNN is capable of low error space robotics odometry with an appealing processing time. Indeed, for the scenarios involving simulated data (Section 3.2), the

proposed deep learning network attains smaller errors compared to the competitor methods, without compromising computational efficiency. Regarding HoD-S/H∞ in the *EoI* and the *Helix* scenarios, it fails to correctly estimate the z-axis translation. Since this method and the DRCNN share the same odometry estimation scheme (Eq. (2) and Eq. (3)), the performance difference is purely due to their architecture. Specifically, the odometry accuracy of HoD-S/H∞ depends on the number of the true positive correspondences, forcing this technique to fail if the point correspondences between the $\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$ point clouds are only a few because the H∞ filtering process performs a limited number of iterations that are equal to the cardinality of the correspondences. Considering the Spin Image/ICP and OUR-CVFH/ICP solutions, these fail to present a valid odometry solution because both feature matching methods, i.e. Spin Images and OUR CVFH, do not attain true positive correspondences due to the sparse nature of the *Target* point cloud. Finally, ICP presents a low odometry accuracy mainly due to implementing it individually on a frame-to-frame basis rather than initializing it with the pose estimated at the previous instance. However, we follow this strategy to preserve a uniform frame-to-frame pose estimation scheme for all techniques evaluated here. For the real data scenarios (Section 3.3), DRCNN is still an appealing space robotics navigation solution with HoD-S/H∞ closely following. For the latter method, it presents more accurate odometry compared to the simulated data scenarios due to the short trajectory path of this scenario and the fact that the HoD-S feature descriptor is specifically designed for sparse point clouds. Likewise, ICP is more accurate compared to the simulated data scenarios due to the short trajectory path. Finally, Spin Image/ICP and OUR-CVFH/ICP are still prone to the highly sparse *Target* point clouds and thus fail to establish true positive correspondences and ultimately a valid odometry solution.

The most important attribute of DRCNN is its adaptive capability that permits DRCNN to be trained on simulated data and evaluated on real 3D LIDAR data, and still attain a space robotics odometry solution with low errors. This feature is quite important as it enables the capability of offline training on a wide variety of low-cost simulated data scenarios neglecting the current requirement of setting up facilities to acquire real 3D LIDAR data to train a deep learning network. The adaptability of our DRCNN architecture is due to the following reasons:

a. Multi-projection of the frame-to-frame point clouds $\boldsymbol{P_k}$ and $\boldsymbol{P_{k+1}}$ enabling the CNN module to extract 2D depth image features from a large portion of the *Target* point cloud that depends on the *Target* pose relative to the LIDAR sensor onboard the *Source*. Within our multi-projection process (Section 2.1) we intentionally quantize the projected point clouds (Eq. (5)) assisting in the establishment of feature correspondences between $\boldsymbol{P_k}$ and $\boldsymbol{P_{k+1}}$

despite minor changes due to the relative *Source* – *Target* motion.

b. The CNN module involves a shallow network, i.e. only three convolutional layers, forcing the extracted features to be quite generic, e.g. corners, blobs and edges [61,62]. The latter enables the trained-on-simulated-data CNN module to generalize to such an extent that it is still capable of real data processing.

c. The RNN module exploits feature maps that rely on generic features and thus regardless of the domain, navigation dynamics are still accurately modeled.

4. Conclusion

In this work, we present a Deep Recurrent Convolutional Neural Network that poses a low error and computationally efficient space robotics odometry solution. Our methodology combines the advantages of 3D implementation, the processing efficiency of manipulating 2D data, and finally the advantages of CNN and RNN architectures. Specifically, our method relies on LIDAR data that are remapped into multiple 2D depth image projections affording a lower computational burden compared to directly manipulating 3D data. Our suggested DRCNN comprises of a CNN module and an RNN module to combine the advantages of both these networks, i.e. feature extraction and learning by the CNN and robust complex dynamics modeling by the RNN.

To evaluate the efficiency and robustness of DRCNN, we initially train our network on a simulated space navigation scenario and then challenge it on realistic simulated space navigation scenarios and real 3D LIDAR data. Our trials demonstrate that the proposed DRCNN architecture is more accurate than current methods while imposing a very low processing burden. Additionally, one of the highlights of our architecture is its adaptive capability, as it is trained on simulated data and is still capable of providing accurate odometry on data of a different modality (real data). This adaptive nature of our proposed architecture affords extending the odometry capabilities of future space robotic platforms.

Despite the promising performance of DRCNN, it is limited by the capabilities of our computer platform. Hence, future work involves implementing DRCNN on a computer platform with a larger GPU memory so that the full potential of the DRCNN can be revealed without the constraints presented in this paper. Furthermore, future scope shall also evaluate DRCNN on more complex real data scenarios with a longer trajectory path.

Conflict of interest statement

None declared

Funding Sources

This research was supported by the European Union's Horizon 2020 research and innovation program, under the project "Integrated 3D Sensors (I3DS)" with grant agreement No 730118.

Acknowledgments

The authors would like to thank Thales Alenia Space (France) for providing the simulated space platform model and the anonymous reviewers for their constructive comments.

References

[1] M.S. Krämer, S. Hardt, K. Kuhnert, Image Features in Space - Evaluation of Feature Algorithms for Motion Estimation in Space Scenarios, in: Proc. 7th Int. Conf. Pattern Recognit. Appl. Methods, SCITEPRESS - Science and Technology Publications, Funchal, Madeira, Portugal, 2018: pp. 300–308. doi:10.5220/0006555303000308.

[2] D. Rondao, N. Aouf, Multi-View Monocular Pose Estimation for Spacecraft Relative Navigation, in: 2018 AIAA Guid. Navig. Control Conf., American Institute of Aeronautics and Astronautics, Reston, Virginia, 2018. doi:10.2514/6.2018-2100.

[3] L. Li, J. Lian, L. Guo, R. Wang, Visual odometry for planetary exploration rovers in sandy terrains, Int. J. Adv. Robot. Syst. 10 (2013) 1–7. doi:10.5772/56342.

[4] T. Tykkala, A.I. Comport, A dense structure model for image based stereo SLAM, in: Robot. Autom. (ICRA), 2011 IEEE Int. Conf., Shangai, China, 2011: pp. 1758–1763. doi:10.1109/ICRA.2011.5979805.

[5] Yang Cheng, M. Maimone, L. Matthies, Visual Odometry on the Mars Exploration Rovers, in: 2005 IEEE Int. Conf. Syst. Man Cybern., Waikoloa, HI, USA, 2006: pp. 903–910. doi:10.1109/ICSMC.2005.1571261.

[6] M. Maimone, Y. Cheng, L. Matthies, Two years of Visual Odometry on the Mars Exploration Rovers, J. F. Robot. 24 (2007) 169–186. doi:10.1002/rob.20184.

[7] O. Yılmaz, N. Aouf, L. Majewski, M. Sanchez-Gestido, G. Ortega, Using infrared based relative navigation for active debris removal, in: 10th Int. ESA Conf. Guid. Navig. Control Syst., Salzburg, Austria, 2017: pp. 1–16.

[8] B. Naasz, M. Moreau, Autonomous RPOD technology challenges for the coming decade, Adv. Astronaut. Sci. 144 (2012) 403–425.

[9] R. Opromolla, M.Z. Di Fraia, G. Fasano, G. Rufino, M. Grassi, Laboratory test of pose determination algorithms for uncooperative spacecraft, in: 4th IEEE Int. Work. Metrol. AeroSpace, Metroaerosp. 2017 - Proc., Padua, Italy, 2017: pp.

169–174. doi:10.1109/MetroAeroSpace.2017.7999558.

[10] J.O. Woods, J.A. Christian, Lidar-based relative navigation with respect to non-cooperative objects, Acta Astronaut. 126 (2016) 298–311. doi:10.1016/j.actaastro.2016.05.007.

[11] R. Volpe, G. Palmerini, M. Sabatini, Monocular and Lidar Based Determination of Shape , Relative Attitude and Position of a Non-Cooperative , Unknown Satellite, in: Int. Astronaut. Congr. (IAC 2017), Adelaide, Australia, 2017: pp. 25–29.

[12] H. Gómez Martínez, G. Giorgi, B. Eissfeller, Pose estimation and tracking of non-cooperative rocket bodies using Time-of-Flight cameras, Acta Astronaut. 139 (2017) 165–175. doi:10.1016/j.actaastro.2017.07.002.

[13] J. Galante, J. Van Eepoel, M. Strube, N. Gill, M. Gonzalez, A. Hyslop, B. Patrick, Pose Measurement Performance of the Argon Relative Navigation Sensor Suite in Simulated-Flight Conditions, in: AIAA Guid. Navig. Control Conf., American Institute of Aeronautics and Astronautics, Reston, Virigina, 2012: pp. 1–26. doi:10.2514/6.2012-4927.

[14] J.L. Sell, A. Rhodes, J.O. Woods, J.A. Christian, T. Evans, Pose Performance of LIDAR-Based Navigation for Satellite Servicing, AIAA/AAS Astrodyn. Spec. Conf. (2014) 1–14. doi:10.2514/6.2014-4360.

[15] J. Song, Sliding window filter based unknown object pose estimation, in: 2017 IEEE Int. Conf. Image Process., IEEE, 2017: pp. 2642–2646. doi:10.1109/ICIP.2017.8296761.

[16] R. Opromolla, G. Fasano, G. Rufino, M. Grassi, Spaceborne LIDAR-based system for pose determination of uncooperative targets, in: 2014 IEEE Int. Work. Metrol. Aerospace, Metroaerosp. 2014 - Proc., Benevento, Italy, 2014: pp. 265–270. doi:10.1109/MetroAeroSpace.2014.6865932.

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

[18] O. Kechagias-Stamatis, N. Aouf, H∞ LIDAR odometry for spacecraft relative navigation, IET Radar, Sonar Navig. 13 (2019) 771–775. doi:10.1049/iet-rsn.2018.5354.

[19] A.P. Rhodes, J.A. Christian, T. Evans, A Concise Guide to Feature Histograms with Applications to LIDAR-Based Spacecraft Relative Navigation, J. Astronaut. Sci. 64 (2017) 414–445. doi:10.1007/s40295-016-0108-y.

[20] A. Dietrich, J.W. McMahon, Orbit Determination Using Flash Lidar Around Small Bodies, J. Guid. Control. Dyn. 40 (2017) 650–665. doi:10.2514/1.G000615.

[21] A.B. Dietrich, J.W. McMahon, Robust Orbit Determination with Flash Lidar Around Small Bodies, J. Guid. Control. Dyn. 41 (2018) 2163–2184. doi:10.2514/1.G003023.

[22] S. Sharma, C. Beierle, S. D'Amico, Pose estimation for non-cooperative spacecraft rendezvous using convolutional neural networks, IEEE Aerosp. Conf. Proc. 2018-March (2018) 1–12. doi:10.1109/AERO.2018.8396425.

[23] M. Zarei-Jalalabadi, S.M.-B. Malaek, Motion estimation of uncooperative space objects: A case of multi-platform

fusion, Adv. Sp. Res. 62 (2018) 2665–2678. doi:10.1016/j.asr.2018.07.031.

[24] O. Kechagias-Stamatis, N. Aouf, M.A. Richardson, High-speed multi-dimensional relative navigation for uncooperative space objects, Acta Astronaut. 160 (2019) 388–400. doi:10.1016/j.actaastro.2019.04.050.

[25] R. Opromolla, G. Fasano, G. Rufino, M. Grassi, A review of cooperative and uncooperative spacecraft pose determination techniques for close-proximity operations, Prog. Aerosp. Sci. 93 (2017) 53–72. doi:10.1016/j.paerosci.2017.07.001.

[26] C. Bonnal, J.M. Ruault, M.C. Desjean, Active debris removal: Recent progress and current trends, Acta Astronaut. 85 (2013) 51–60. doi:10.1016/j.actaastro.2012.11.009.

[27] O. Kechagias-Stamatis, N. Aouf, D. Nam, 3D Automatic Target Recognition for UAV Platforms, in: 2017 Sens. Signal Process. Def. Conf., IEEE, London, UK, 2017: pp. 1–5. doi:10.1109/SSPD.2017.8233223.

[28] O. Kechagias-Stamatis, N. Aouf, M.A. Richardson, 3D automatic target recognition for future LIDAR missiles, IEEE Trans. Aerosp. Electron. Syst. 52 (2016) 2662–2675. doi:10.1109/TAES.2016.150300.

[29] P.J. Besl, N.D. McKay, A method for registration of 3-D shapes, IEEE Trans. Pattern Anal. Mach. Intell. 14 (1992) 239–256. doi:10.1109/34.121791.

[30] R. Opromolla, G. Fasano, G. Rufino, M. Grassi, Uncooperative pose estimation with a LIDAR-based system, Acta Astronaut. 110 (2015) 287–297. doi:10.1016/j.actaastro.2014.11.003.

[31] L. Liu, G. Zhao, Y. Bo, Point cloud based relative pose estimation of a satellite in close range, Sensors (Switzerland). 16 (2016). doi:10.3390/s16060824.

[32] A. Rhodes, E. Kim, J.A. Christian, T. Evans, LIDAR-based Relative Navigation of Non-Cooperative Objects Using Point Cloud Descriptors, in: AIAA/AAS Astrodyn. Spec. Conf., American Institute of Aeronautics and Astronautics, Reston, Virginia, 2016. doi:10.2514/6.2016-5517.

[33] A. Aldoma, F. Tombari, R.B. Rusu, M. Vincze, OUR-CVFH – Oriented, Unique and Repeatable Clustered Viewpoint Feature Histogram for Object Recognition and 6DOF Pose Estimation, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), 2012: pp. 113–122. doi:10.1007/978-3-642-32717-9_12.

[34] A.E. Johnson, M. Hebert, Using spin images for efficient object recognition in cluttered 3D scenes, IEEE Trans. Pattern Anal. Mach. Intell. 21 (1999) 433–449. doi:10.1109/34.765655.

[35] O. Kechagias-Stamatis, N. Aouf, V. Dubanchet, Evaluating 3D local descriptors and recursive filtering schemes for LIDAR-based uncooperative relative space navigation, J. F. Robot. (2019) rob.21904. doi:10.1002/rob.21904.

[36] A. Nemra, N. Aouf, Robust Airborne 3D Visual Simultaneous Localization and Mapping with Observability and Consistency Analysis, J. Intell. Robot. Syst. 55 (2009) 345–376. doi:10.1007/s10846-008-9306-6.

[37] I. Cvišić, J. Ćesić, I. Marković, I. Petrović, SOFT-SLAM: Computationally efficient stereo visual simultaneous

localization and mapping for autonomous unmanned aerial vehicles, J. F. Robot. 35 (2018) 578–595. doi:10.1002/rob.21762.

[38] T. Mouats, N. Aouf, L. Chermak, M.A. Richardson, Thermal Stereo Odometry for UAVs, IEEE Sens. J. 15 (2015) 6335–6347. doi:10.1109/JSEN.2015.2456337.

[39] T. Mouats, N. Aouf, A.D. Sappa, C. Aguilera, R. Toledo, Multispectral Stereo Odometry, IEEE Trans. Intell. Transp. Syst. 16 (2015) 1210–1224. doi:10.1109/TITS.2014.2354731.

[40] J. Zhang, S. Singh, Visual-lidar odometry and mapping: low-drift, robust, and fast, in: 2015 IEEE Int. Conf. Robot. Autom., IEEE, 2015: pp. 2174–2181. doi:10.1109/ICRA.2015.7139486.

[41] J. Zhang, S. Singh, LOAM: Lidar Odometry and Mapping in Real- time, IEEE Trans. Robot. 32 (2015) 141–148. doi:10.15607/RSS.2014.X.007.

[42] Q. Li, S. Chen, C. Wang, X. Li, C. Wen, M. Cheng, J. Li, LO-Net: Deep Real-Time Lidar Odometry, in: 2019 IEEE/CVF Conf. Comput. Vis. Pattern Recognit., IEEE, 2019: pp. 8465–8474. doi:10.1109/CVPR.2019.00867.

[43] M. Jaimez, J. Gonzalez-Jimenez, Fast Visual Odometry for 3-D Range Sensors, IEEE Trans. Robot. 31 (2015) 809–822. doi:10.1109/TRO.2015.2428512.

[44] D.-H. Kim, J.-H. Kim, Effective Background Model-Based RGB-D Dense Visual Odometry in a Dynamic Environment, IEEE Trans. Robot. 32 (2016) 1565–1573. doi:10.1109/TRO.2016.2609395.

[45] Y. Zhou, H. Li, L. Kneip, Canny-VO: Visual Odometry With RGB-D Cameras Based on Geometric 3-D–2-D Edge Alignment, IEEE Trans. Robot. 35 (2019) 184–199. doi:10.1109/TRO.2018.2875382.

[46] M.O.A. Aqel, M.H. Marhaban, M.I. Saripan, N.B. Ismail, Review of visual odometry: types, approaches, challenges, and applications, Springerplus. 5 (2016). doi:10.1186/s40064-016-3573-7.

[47] B. Chen, J. Cao, A. Parra, T.-J. Chin, Satellite Pose Estimation with Deep Landmark Regression and Nonlinear Pose Refinement, in: 2019 IEEE/CVF Int. Conf. Comput. Vis. Work., IEEE, 2019: pp. 2816–2824. doi:10.1109/ICCVW.2019.00343.

[48] M. Estébanez Camarena, L.M. Feetham, A. Scannapieco, N. Aouf, FPGA-based multi-sensor relative navigation in space: Preliminary analysis in the framework of the I3DS H2020 project, in: 69 Th Int. Astronaut. Congr. (IAC), Internation Astronautical Federation, Bremen, 2018: pp. 1–8.

[49] M. Boulekchour, N. Aouf, M. Richardson, Robust L∞ convex optimisation for monocular visual odometry trajectory estimation, Robotica. 34 (2016) 703–722. doi:10.1017/S0263574714001829.

[50] M. Boulekchour, N. Aouf, L∞ norm based solution for visual odometry, Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics). 8048 LNCS (2013) 185–192. doi:10.1007/978-3-642-40246-3_23.

[51] F. Gers, Long short-term memory in recurrent neural networks, EPFL, 2001.

[52] S. Wang, R. Clark, H. Wen, N. Trigoni, DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks, in: 2017 IEEE Int. Conf. Robot. Autom., IEEE, 2017: pp. 2043–2050. doi:10.1109/ICRA.2017.7989236.

[53] M. Valente, C. Joly, A. de La Fortelle, An LSTM Network for Real-Time Odometry Estimation, in: 2019 IEEE Intell. Veh. Symp., IEEE, 2019: pp. 1434–1440. doi:10.1109/IVS.2019.8814133.

[54] S. Katz, A. Tal, R. Basri, Direct visibility of point sets, ACM Trans. Graph. 26 (2007) 24. doi:10.1145/1276377.1276407.

[55] B. Alsadik, M. Gerke, G. Vosselman, Visibility analysis of point cloud in close range photogrammetry, ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. II–5 (2014) 9–16. doi:10.5194/isprsannals-II-5-9-2014.

[56] Optitrack, (2018). https://optitrack.com/ (accessed May 22, 2018).

[57] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, J. Wan, N.M. Kwok, A Comprehensive Performance Evaluation of 3D Local Feature Descriptors, Int. J. Comput. Vis. 116 (2016) 66–89. doi:10.1007/s11263-015-0824-y.

[58] O. Kechagias-Stamatis, N. Aouf, Histogram of distances for local surface description, in: 2016 IEEE Int. Conf. Robot. Autom., IEEE, Stockholm, Sweden, 2016: pp. 2487–2493. doi:10.1109/ICRA.2016.7487402.

[59] L.A. Alexandre, 3D Descriptors for Object and Category Recognition : a Comparative Evaluation, IEEE/RSJ Int. Conf. Intell. Robot. Syst. 34 (2012) 1–6. doi:10.1109/TPAMI.2011.263.

[60] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? The KITTI vision benchmark suite, in: 2012 IEEE Conf. Comput. Vis. Pattern Recognit., IEEE, 2012: pp. 3354–3361. doi:10.1109/CVPR.2012.6248074.

[61] O. Kechagias-Stamatis, N. Aouf, Fusing Deep Learning and Sparse Coding for SAR ATR, IEEE Trans. Aerosp. Electron. Syst. 55 (2019) 785–797. doi:10.1109/TAES.2018.2864809.

[62] O. Kechagias-Stamatis, Target recognition for synthetic aperture radar imagery based on convolutional neural network feature fusion, J. Appl. Remote Sens. 12 (2018) 1. doi:10.1117/1.JRS.12.046025.