FinderNet: A Data Augmentation Free Canonicalization aided Loop Detection and Closure technique for Point clouds in 6-DOF separation.

Sudarshan S Harithas*[1] Gurkirat Singh*[1] Aneesh Chavan[1] Sarthak Sharma[1] Suraj Patni[2]
Chetan Arora[2] Madhava Krishna[1]
[1]Robotics Research Center, IIIT Hyderabad [2]IIT Delhi

Abstract

We focus on the problem of LiDAR point cloud based loop detection (or Finding) and closure (LDC) for mobile robots. State-of-the-art (SOTA) methods directly generate learned embeddings from a given point cloud, require large data augmentation, and are not robust to wide viewpoint variations in 6 Degrees-of-Freedom (DOF). Moreover, the absence of strong priors in an unstructured point cloud leads to highly inaccurate LDC. In this original approach, we propose independent roll and pitch canonicalization of point clouds using a common dominant ground plane. We discretize the canonicalized point clouds along the axis perpendicular to the ground plane leads to images similar to digital elevation maps (DEMs), which expose strong spatial priors in the scene. Our experiments show that LDC based on learnt embeddings from such DEMs is not only data efficient but also significantly more robust, and generalizable than the current SOTA. We report an (average precision for loop detection, mean absolute translation/rotation error) improvement of (8.4, 16.7/5.43)% on the KITTI08 sequence, and (11.0, 34.0/25.4)% on GPR10 sequence, over the current SOTA. To further test the robustness of our technique on point clouds in 6-DOF motion we create and opensource a custom dataset called Lidar-UrbanFly Dataset (LUF) which consists of point clouds obtained from a LiDAR mounted on a quadrotor. More details on our website https://gsc2001.github.io/FinderNet/

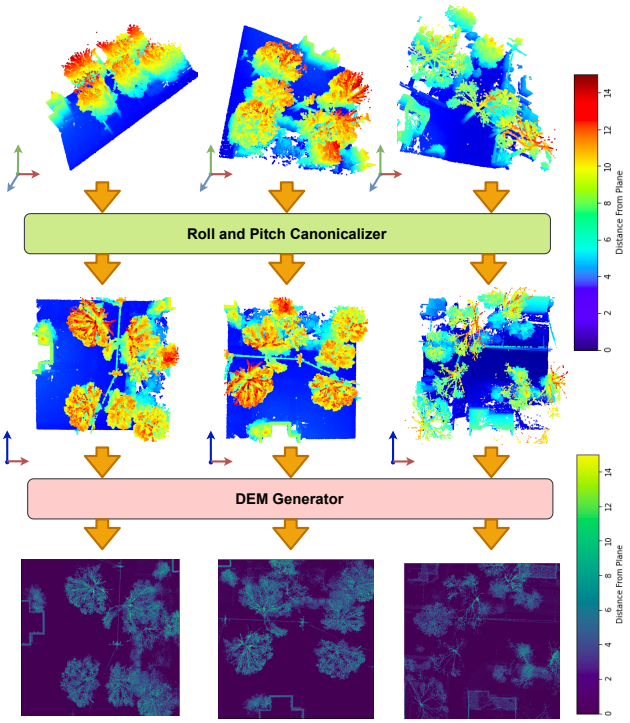

Figure 1: We observe that raw LiDAR point clouds (first row) lack spatial structure for robust loop detection and closure (LDC). We perform local roll and pitch canonicalization (second row), followed by discretization along the z-axis (third row), which leads to output similar to digital elevation maps (DEMs) and exposes rich scene structure in the input. Our model performs LDC on such DEMs, leading to high data efficiency, robustness, and generalizability to 6-DOF viewpoint variations.

1. INTRODUCTION

Loop detection and closure is a critical module in the SLAM (*Simultaneous Localization And Mapping*) pipeline to reduce accumulated drift in the estimation process. Recovering the closest possible match between a given query point cloud and a pre-built database is known as *loop detection* (or place recognition), whereas the process of estimating the relative transform between the query and the retrieved sample is known as *loop closure*. Although most components of our pipeline apply to generic point clouds, we assume LiDAR is the primary sensing modality.

The techniques are broadly split into two styles [31] : **(1)** *Loop Detection* These approaches typically use place recognition methods [23, 26, 28, 33] to detect loops and employ traditional point cloud registration algorithms such as [30, 34] to estimate the relative pose between the query and retrieved point clouds. **(2)** *Loop Detection and Closure*

*Equal Contribution

This work was funded in part by grants made available to The Robotics Research Center, IIIT Hyderabad from MathWorks India (Hyderabad).

Methods such as [3, 4, 20] perform place recognition and estimate the relative pose between the query and retrieved point cloud in an end-to-end pipeline without an external method to register point clouds . Our approach belongs to second category, where we detect and close loops as a part of a single pipeline without employing external point cloud registration.

Typically, these methods [3, 23, 26, 28, 33] either depend upon a combination of feature aggregation and data augmentation where they apply randomly sampled rigid transforms to the input point clouds to achieve viewpoint invariance or perform LDC through overlap estimation. Such a training procedure does not generalize to wide viewpoint variations. We take an original approach where a combination of canonical representations and differentiable latent space alignment is used to geometrically constrain view invariance into the system. Such a technique leads to SOTA performance on multiple datasets.

The cornerstone of our efforts is a *Roll and Pitch (RP) canonicalizer* and a *Differentiable Yaw Transformer* (DYT). The *RP Canonicalizer* makes use of the dominant ground plane hypothesis (commonly encountered in autonomous driving and drone applications) [4, 13, 15, 16, 20] to compensate for the roll and pitch between two point clouds. The roll and pitch canonicalized point clouds are converted into a *Digital Elevation Map* (DEM), a visual explanation of the process is given in Fig. 1. We further develop a *Differentiable Yaw Transformer* (DYT) that operates on the latent feature embeddings of the DEM to achieve yaw invariance, and provide viewpoint invariant loop detection with 6-DOF (SE(3)) relative motion. This is in contrast to existing techniques focusing only on yaw rotation [4, 12, 13, 29].

Contributions: **(1) Novel Pipeline:** Instead of directly operating on the raw point clouds that inherently lack structure, we convert the point clouds into a regularly spaced DEM via a roll and pitch canonicalizer (Section 3.1) with only the yaw to further deal with. The canonicalized DEM representation provides a structure that CNN backbones can readily process, bypassing equivariance issues that typically plague point cloud representations. The proposed framework goes beyond SOTA on a number of public datasets such as KITTI [9], and GPR [32] on established performance metrics for LDC. Specifically, the proposed framework is the best performing on 6-DOF pose recovery and it outperforms most prior art on the SE(2) LDC task [4, 13]. **(2) Achieving view-invariance through Canonicalization & Differentiable Alignment** Unlike previous methods that approach LDC through feature aggregation [3,23,28,33] or overlap estimation [4, 17], we approach LDC through a canonicalization and differentiable alignment procedure , that enables us to geometrically constrain view invariance into the network and enables training with no data augmentation, and achieves SOTA results on multiple datasets.

Method	Venue	VI	NDA	LD	LDC
OverlapNet [4]	RSS'20	✗	✗	✓	✓(Yaw)
ScanContext [13]	IROS'18	✗	NA	✓	✓(Yaw)
Dcpcr [25]	RAL'22	✓	NA	✗	✗
Retriever [26]	ICRA'22	✓	✗	✓	✗
Deep compression [27]	RAL'21	✓	✗	✗	✗
LCDNet [3]	TRO'22	✓	✗	✓	✓
Oreos [20]	IROS'19	✗	✗	✓	✓(Yaw)
PointnetVLAD [23]	CVPR'18	✓	✗	✓	✗
PCAN [33]	CVPR'19	✓	✗	✓	✗
SOE-Net [28]	CVPR'21	✓	✗	✓	✗
DH3D [6]	ECCV'20	✓	✗	✓	✓
Ours	****'23	✓	✓	✓	✓

Table 1: Acronyms: **VI:** 6-DOF View Invariance, **NDA:** No large Data Augmentation requirement, **LD:** Loop Detection, **LDC:** Joint Loop Detection and Closure, **NA:** Not Applicable.

Our latent space alignment is driven by the DYT which is a novel parameter estimation module which is used for differentiable grid sampling. In contrast to methods such as [4, 17, 20] the DYT allows us to estimates the relative yaw in a *self-supervised* manner, i.e. it does not require explicit supervision of the relative yaw between the two point clouds. The DYT demonstrates a decrease of 13.33% in yaw estimation error in the *Kitti00* sequence and against [4]. **(3) 6-DOF recovery:** Unlike previous approaches that show loop closure only as a SE(2) alignment, the proposed method recovers 6-DOF pose between the two candidate point clouds, even as it precludes the need for data augmentation, exploiting the inherent viewpoint invariance of the pipeline. On *LUF* dataset *FinderNet* shows a 10% improvement in *average-precision* in loop detection on , 2.91% and 9.54% decrease in the *Avergae translation error* and *Average rotation error* in the loop closure task against *LCDNet* [3] the SOTA LDC method. Table 1 gives a conceptual comparison of our method with contemporary techniques, where **LD** are methods that perform the task of loop detection and employ an external point cloud registration method [30, 34] to estimate the relative pose. **LDC** are methods that jointly estimates the loop and the relative pose through a single pipeline. **NDA** is set to true (or ✓) when a method can learn without data augmentation and **VI** are methods that can handle SE(3) motion.

2. Related works

Handcrafted Feature Descriptors: [13, 16, 24] rely on handcrafted feature descriptors to extract local geometric information and aggregate it to obtain a global descriptor suitable for loop detection and closure. [13, 16] assume presence of a dominant ground plane to detect and close loops. [13] follows a polar representation, where the ground is discretized into bins by splitting in both radial and azimuthal directions, and each bin storing the maximum height present in the vertical volume. [16] discretizes the ground plane into rectangular cells in a Cartesian form,

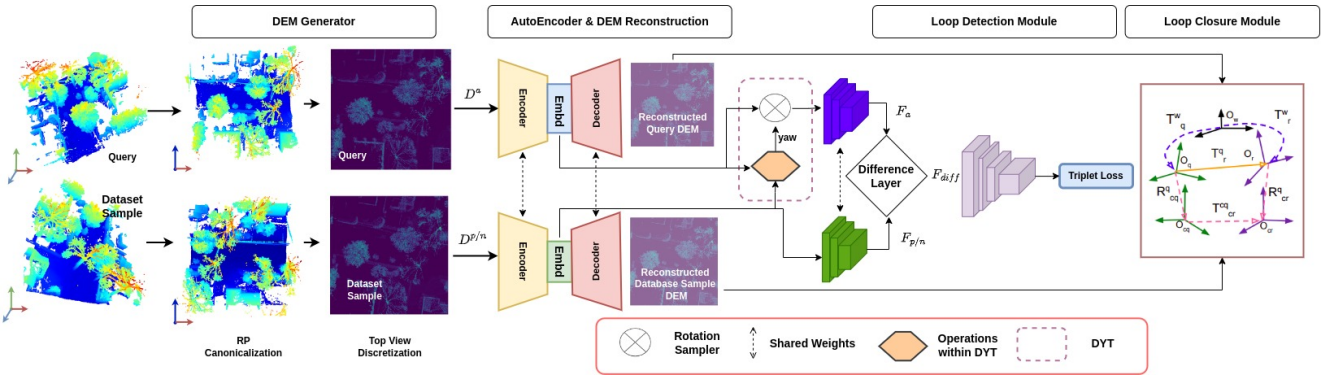

Figure 2: The figure demonstrates the overview of our pipeline; the two point clouds in the extreme left are the input query and database sample; the *DEM Generator* (section 3.1) generates a discredited top view of the point cloud; and the *DEM* are further passed through the autoencoder structure (section 3.2). The *Differentiable Yaw Transformer* (DYT) (section 3.3) is used for the yaw alignment, the operations within the DYT include *CPC*, *Horizontal padding of polar embedding*, and *Correlation*; each of these are explained in section 3.3. The complete set of operations is shown as a single orange hexagon; the result of these operations is a scalar yaw value, which is fed into the rotation sampler. The designed network performs loop detection (section 3.4) and closure (section 3.5).

and each cell storing it's point cloud density. Our representation of DEM is a discrete Cartesian representation of the ground plane, where each grid cell stores the maximum height of the points present in it. However, unlike [13, 16], we perform a complete 6-DOF estimation and loop closure.

Learning Based Approaches for Loop Detection: PointNet [18] proposes a neural network model that directly consumes point clouds while maintaining permutation invariance. *PointNetVLAD* [23] uses [18] and NetVLAD [1] to generate global descriptors for place recognition. *PCAN* [33] uses [18] as the backbone architecture to extract local features and the corresponding attention maps along with [1] for feature aggregation. However, both [23, 33] uses PointNet as a backbone architecture, which processes each point separately via a MLP, not capturing local neighbourhood information. Recently, *Retriever* [26] detects loops directly in the compressed feature representation using *Perceiver* [11] based mechanism to aggregate the local features. All the above methods use aggregated local features, to compute a global descriptor that is viewpoint invariant, such methods require expensive data augmentation. We propose a canonicalization procedure in order to explicitly enforce viewpoint invariance and in contrast to [23, 26, 33] which only perform loop detection, our method performs LDC.

Learning Based Approaches for LDC: LCDNet [3] proposes an end-to-end trainable system, with a *Unbalanced Optimal Transport* algorithm to estimate 6-DOF relative transform between two point clouds. DH3D [6] aggregates local features using hierarchical network to obtain global features for loop detection. Both [3, 6] rely on an expensive 6-DOF data augmentation of the input point cloud in order to achieve orientation invariance. The proposed framework bypasses data augmentation through explicit roll-pich canonicalization followed by yaw alignment. Overlap-based approaches such as *OverlapNet* [4] and *OverlapTransformer* [17] are trained using explicit overlap information on range images [2]. OREOS [20] proposes two separate branches: one for loop closure and other for loop detection. Unlike [4, 17, 20] that only estimate the relative yaw between the input point clouds, we estimate the full 6-DOF relative pose.

3. Methodology

Our goal is to develop a 6-DOF viewpoint invariant place recognition framework for 3D point clouds for LDC. The overview of our method is shown in Fig. 2. We first canonicalize the point cloud, and then discretize it to get a DEM representation (Section 3.1). We use an *autoencoder* style encoder-decoder network to learn the latent representation for the DEM (Section 3.2).

To achieve yaw invariance for loop detection in the latent space, we have designed a *Differentiable Yaw Transformer* (DYT), it transforms the latent query embedding to rotationally align with the latent embedding of the database sample (Section 3.3).The output of the DYT is used for loop detection (Section 3.4). Once a loop is detected, the DEM from the query and retrieved point clouds can be used to estimate the relative 6-DOF relative pose for the loop closure (Section 3.5).

3.1. DEM Generation

DEMs are digital representations of an input point cloud, capturing the elevation of the terrain or overlaying objects. DEMs have rich representation power, preserving the feature rich regions like edges and corners, and at the same time can be assimilated by *CNN* architectures. Moreover, unlike range images that preserve only yaw [4], DEMs preserve both yaw and planar translation, making them a useful representation for 6-DOF point cloud registration.

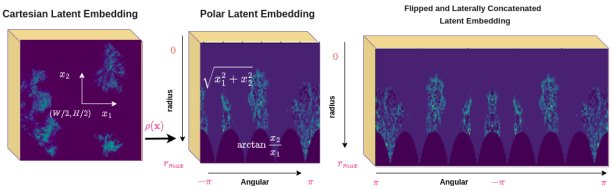

Figure 3: The image to the extreme left shows a sample *DEM* latent space in Cartesian form. The image in the center depicts the same embedding in a polar form; the image to the right is the result of flipping and concatenation operation.

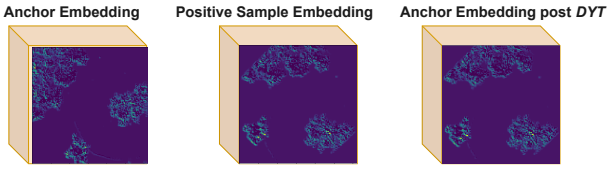

Figure 4: Visualization of the yaw alignment using DYT. Note that the anchor and the positive sample are not yaw aligned initially, however post the *DYT* operation the two embeddings are aligned. We show the first channel of the feature volume as binary image for ease of visualization.

Plane Parameterization: Consider an input point cloud $\mathbf{P}_c$ with its corresponding ground plane $\mathbf{r}_c$, and the world ground-plane $\mathbf{r_w}$. We aim to align the planes $\mathbf{r}_c$ and $\mathbf{r}_w$ by estimating the relative roll and pitch (RP) between them. We center the input point cloud ($\mathbf{P}_c$) and extract the ground plane $\mathbf{r}_c$ using RANSAC. The ground plane is parameterized by ($\mathbf{n}_c$, $\mathbf{C}_c$), where $\mathbf{n}_c \in \mathbf{R}^3$ is a unit vector perpendicular to the plane and $\mathbf{C}_c = \{c_i^c \mid i = \{1...n\}\} \in \mathbf{R}^{n\times 3}$ is the set of points $c_i^c \in \mathbf{R}^3$, s.t. c_i^c lies on $\mathbf{r_c}$ and $\|c_i^c\| = 1$. The world ground plane $\mathbf{r}_w$ is parameterized similarly as ($\mathbf{n}_w$, $\mathbf{C}_w$), where $n_w = [0, 0, 1]$ and $\mathbf{C}_w = \{c_i^w \mid i = \{1...n\}\} \in \mathbf{R}^{n\times 3}$ is the set of points $c_i^w \in \mathbf{R}^3$, s.t. c_i^w lies on $\mathbf{r_w}$ and $\|c_i^w\| = 1$. Note that the world ground plane is not estimated through data, instead is a *constructed canonical plane of reference*. The canonicalization for roll (α), and pitch (β) involves two steps. First we obtain a coarse estimate of (α) and (β) by aligning the normals $\mathbf{n}_c$ and $\mathbf{n}_w$. Post that, we do a finer estimate through Iterative Closest Point (ICP).

Coarse RP Canonicalization: Given the normals $\mathbf{n}_c = [n_{c,x}, n_{c,y}, n_{c,z}]$ from RANSAC, and $\mathbf{n}_w = [0, 0, 1]$, we estimate the relative roll α and pitch β by solving:

$$\begin{bmatrix}0\\0\\1\end{bmatrix} = \begin{bmatrix}\cos(\alpha) & 0 & \sin(\alpha)\\ \sin(\beta)\sin(\alpha) & \cos(\beta) & -\cos(\alpha)\sin(\beta)\\ -\cos(\beta)\sin(\alpha) & \sin(\beta) & \cos(\alpha)\cos(\beta)\end{bmatrix}\begin{bmatrix}n_{c,x}\\n_{c,y}\\n_{c,z}\end{bmatrix}.$$

The obtained closed-form solution is given as:

$$\alpha = \arctan\left(\frac{-n_{c,x}}{n_{c,z}}\right), \beta = \arctan\left(\frac{n_{c,y}}{n_{c,z}\cos(\alpha) - n_{c,x}\sin(\alpha)}\right)$$

Fine Grained Canonicalization with ICP: We use the coarse estimates of α and β as described above, and refine them using ICP (initialized with coarse estimates) as:

$$R(\alpha, \beta) = \arg\min_{(\alpha,\beta)} ||\mathbf{C_w} - R(\alpha, \beta)\mathbf{C_c}||^2$$

Top-view Discretization: After performing the roll and pitch canonicalization, the top-view of the point cloud is discretized into uniform $2D$ grid cells to obtain the DEM $\mathbf{D}_c$ of point cloud $\mathbf{P}_c$. We define a grid G of dimension $G_w \times G_h$, and resolution d_g. Each grid cell $g_i \in G$ is assigned a set of points P_i^g based on the resolution d_g, and given a height value $h_i^g = \max(h(p) \mid p \in P_i^g)$, where $h(p)$ is the height of the point p, thus converting a point cloud to a DEM. Such a grid representation can be readily assimilated by Deep Networks ideally suited to exploit such structural information.

3.2. Learning The Latent Representation

We intend to use the DEMs to perform the LDC task, and an autoencoder-decoder with bottleneck architecture is used, where the encoder part reduces the dimensionality of the input DEM to a lower-dimensional latent space, and the decoder part reconstructs the original input from this latent representation. By constraining the dimensionality of the latent space, the model forces the encoded representation to capture the most salient features of the input data. The autoencoder-decoder structure further enforces the structural consistency within the latent space. We use ϕ to denote the embedding, and $\phi \in R^{4\times 125\times 125}$ and the detailed description of our encoder-decoder architecture is given in the supplementary.

3.3. Differentiable Yaw Transformer *(DYT)* for Parameterized Grid Sampling

Viewpoint invariance is an important property for robust place recognition/loop detection. Previous methods [3, 20] try to achieve this through data augmentation, where they rotate an input point cloud through randomly selected rotation angles. However, such methods do not generalize to complex sequences or large changes in viewpoint. [3] acknowledges that data augmentation by itself need not be sufficient for viewpoint invariance. Therefore, we propose a *Differentiable Yaw Transformer* (DYT) module that achieves viewpoint invariance without the need for explicit data augmentation. It achieves this by receiving the latent embedding of anchor and positive (or negative) DEM (denoted as ϕ_A, ϕ_p, and ϕ_n respectively) as its input and returning the relative yaw denoted by $\psi \in R$ at the output. Then, it rotates the anchor DEM so that the relative yaw between the anchor and positive is zero. The operations within the DYT are detailed below.

Cartesian to Polar Conversion (CPC): Let $\mathbf{G}$ be a group of rotation transformations (in SO(2)) parameterized by ψ s.t. $T_\psi : R^d \rightarrow R^d, \forall T_\psi \in SO(2)$. The canonical coordinate for $\mathbf{G}$ are defined such that a rotation by T_ψ in the

Cartesian coordinates appears as a translation by ψ in the canonical coordinates. The polar coordinate system forms such canonical coordinates for the group of rotation transformations [8, 22], and can be obtained from Cartesian coordinates $\mathbf{x} : (x_1, x_2)$ as:

$$\rho(\mathbf{x}) = \left(\arctan \frac{x_2}{x_1}, \sqrt{x_1^2 + x_2^2} \right). \quad (1)$$

CPC is performed for each channel of the embedding tensor ϕ, that results in an output tensor of the same size. The *CPC* process is shown in the first two columns of Fig. 3

Horizontally Padding Polar Embedding: As described above two embeddings related by a yaw rotation in the Cartesian coordinates are related by a translation after conversion to polar coordinates. However, if we try to estimate translation directly, the estimation process can only correlate between the overlapping regions. We observe that the horizontal axis of the polar latent embedding lies within the range $[-\pi, \pi]$ and is cyclic. The cyclic property allows us to pad the embedding by copying the embedding, flipping it (the flipped embedding will be within the range $[\pi, -\pi]$), and then use the flipped version to horizontally pad the embedding. The resulting embedding is shown in Fig. 3 (extreme right). The operation doubles the size of the latent embedding to $4 \times 125 \times 250$, and allows us to use full embedding for translation estimation.

Correlation Layer: After padding the polar embedding from the positive (negative) embedding, we try to locate anchor embedding in it using correlation. We implement the layer as a convolutional layer with polar latent embedding of the anchor as a kernel, and perform cross-correlation over the horizontally padded polar feature volume of the positive/negative sample. This results in a 1D output of size $1 \times 1 \times 126$. The output of the correlation layer divides the 360 degrees of rotation into 126 bins, each of resolution 2.85 degrees (approximately). We apply softmax over the correlation score output to convert the score vector to the probability vector for various candidate translations. The predicted translation is multiplied by 2.85 to convert to predicted rotation angle. Modules similar to correlation layer have been previously used in [4], however, their setting required explicit yaw supervision, one of our contribution is to relax this requirement by formulating a *soft* yaw estimation as a part of the self-supervised DYT. This results in improved performance for large view-point changes as demonstrated by the *Kitti-08* and *LUF* sequences in Section 5.1.

Rotation Sampler: We construct a rotation matrix $R_\psi \in$ SO(2) from the predicted yaw angle (ψ) as determined from the previous step. Similar to [10], we use R_ψ to differentiably sample from the input feature volume and produce a warped output feature map, denoted as $\widehat{\phi}$. The operation is denoted as $\otimes$ in Fig. 2. Note that the operation is performed on $4 \times 125 \times 125$ dimensional embedding tensor in Cartesian coordinates. Fig. 4 depicts the result of the DYT module, it may be seen that the anchor and positive sample do not share the same orientation at the input of DYT. However, post-DYT, they have same orientation. For simplicity of illustration, we only show the first channel of the $4 \times 125 \times 125$ tensor. The warped anchor tensor is sent to the next module for loop detection.

3.4. Loop Detection

Our pipeline achieves rotation invariance using the RP canonicalizer and DYT. Additionally, the difference layer within the loop detection module provides translation invariance and measures the similarity between the two yaw-aligned DEMs. A fully convolutional network (CNN) is translation-equivariant. Our loop detection module consists of shared CNN layers to extract features $F_a \in R^{H \times W \times C}$ from the anchor and $F_{p/n} \in R^{H \times W \times C}$ features from the positive or negative sample DEM. The difference layer takes the two feature volumes as input and computes all pairs absolute differences between the pixels. To implement all pairs absolute difference we first construct a tile tensor $T_a \in R^{HW \times HW \times C}$ by first reshaping F_a to a $HW \times 1 \times C$ tensor, and then repeating first column in each channel by HW times. Mathematically: $\forall i \in \{0, 1, 2, .., H-1\}$ and $j \in \{0, 1, 2, .., W-1\}$.

$$T_a(iW + j, k, c) = F_a(i, j, c), \quad \forall k \in [0, HW - 1].$$

We compute T_p and T_n similarly, but additionally transpose each channel of the tensor at the end. This is equivalent to:

$$T_{p/n}(k, iW + j, c) = F_{p/n}(i, j, c), \quad \forall k \in [0, HW - 1].$$

T_a and $T_{p/n}$ allow us to compute all pair difference as: $F_{\text{diff}} = |T_a - T_{p/n}|$.

The difference layer results in a feature volume F_{diff} that quantifies the shared information between the two DEMs. F_{diff} is passed through proposed CNN architecture (details in the supplementary material), resulting in a single scalar value indicating the distance between the two DEMs. A low value indicates loop detection. We train the proposed loop detection module using triplet based contrastive loss:

$$\mathcal{L}_{\text{triplet}} = \max \left(0, d\left(\widehat{\phi}_a, \phi_p\right) - d\left(\widehat{\phi}_a, \phi_n\right) + \xi \right), \quad (2)$$

where ϕ is the DEM encoding, $\widehat{\phi}$ is the yaw aligned DEM encoding, d is the distance between the two encoding computed by the loop detection module, and ξ is the margin for the triplet loss. Note that during back-propagation, the DEM encoder receives gradient both from the MSE loss of the autoencoder, as well as from the above triplet loss. Whereas the decoder is trained only using the MSE loss.

3.5. Loop Closure

After performing the loop detection process, we estimate the SE(3) rigid body transform to align the query and retrieved point cloud. This process is known as loop closure (or point cloud registration). Let $\mathbf{P}_q$ be the query point cloud, with pose $\mathbf{T}_q^w$, centered at $\mathbf{O}_q$. Let $\mathbf{P}_r$ be the retrieved point cloud, centered at origin $\mathbf{O}_r$ with a pose $\mathbf{T}_r^w$. Refer to the loop closure block in the extreme right of Fig. 2. Both $\mathbf{T}_q^w$ and $\mathbf{T}_r^w$(denoted in dotted blue) are in the world frame of reference, and are unknown. We aim to find the relative transformation $\mathbf{T}_r^q$ (in solid yellow) that aligns $\mathbf{P}_q$ and $\mathbf{P}_r$. To estimate the relative SE(3) pose, we first estimate the relative SE(2) transform between $\mathbf{P}_q$ and $\mathbf{P}_r$. Post that, its combined with the initially estimated roll and pitch canonicalization to obtain the SE(3) pose estimation.

To estimate the relative SE(2) transform we decode the query and retrieved DEMs from their respective encoding. By performing loop closure on the decoded DEM it becomes possible to utilize *FinderNet* in bandwidth-constrained scenarios, such as collaborative SLAM, in this context, only the compressed latent embeddings would need to be transmitted and decoded at the receiver to estimate the relative pose. Then, key-points and correspondences between the query and retrieved DEMs is obtained using [5, 19]. The SE(2) pose is obtained from the following optimization problem

$$\underset{\psi, \mathbf{t}_{cr}^{cq}}{\arg\min} \| \left(\mathbf{R}(\psi)_{cr}^{cq} a_i + \mathbf{t}_{cr}^{cq}\right) - b_i \|^2. \tag{3}$$

Here, a_i, and b_i are the corresponding points on the query and target DEM respectively. The rotation matrix $\mathbf{R}(\psi)_{cr}^{cq} \in$ SO(2) is parameterized by the yaw angle ψ and the translation vector is denoted by $\mathbf{t}_{cr}^{cq} \in R^2$. The translation vector is scaled to the metric scale using the grid resolution d_g (c.f. Top-view Discretization within Section 3.1). The yaw angle ψ for the optimization is initialized using the yaw estimates from the DYT module. Let $\mathbf{R}(\alpha_q, \beta_q)_{cq}^{q}$ and $\mathbf{R}(\alpha_r, \beta_r)_{cr}^{r}$ (shown in pink in Loop Closure module of Fig. 2) be the rotation matrices that align the query ($\mathbf{P}_q$) and retrieved point cloud ($\mathbf{P}_r$) to their respective roll-pitch compensated frames $\mathbf{O}_q$ and $\mathbf{O}_r$. To estimate the SO(3) rotation matrix, we combine $\mathbf{R}_{cq}^{q}, \mathbf{R}_{cr}^{cq}$ and $\mathbf{R}_{cr}^{q}$: $\mathbf{R}_r^q = \mathbf{R}_{cq}^{q} \mathbf{R}_{cr}^{cq} (\mathbf{R}_{cr}^{r})^{-1}$.

We obtain the translation $\mathbf{t}_r^q$ by combining $\mathbf{t}_{cr}^{cq}$ and $d_r - d_q$: where d_r and d_q are the distance of the LiDAR from the estimated ground plane obtained (it is esimated along with the ground plane parameters through RANSAC). $\mathbf{t}_r^q = [\mathbf{t}_{cr}^{cq}(0), \mathbf{t}_{cr}^{cq}(1), d_r - d_q$.

4. Datasets and Implementation Details

We use PyTorch, and train on a single NVIDIA GeForce GTX 1080 GPU, using a batch size of 12 and ADAM [14] as optimizer for 200 epochs for 8 hours. The learning rate is initialized to 4×10^{-4} and halved every 50 epochs. A $50m \times 50m$ point cloud is converted to a linearly scaled DEM representation of size 500×500 pixels. The triplet margin, k, in Eq. (2) is set to 0.75. Unlike [3, 4, 23, 28, 33], we do not perform any augmentation on the input point clouds.

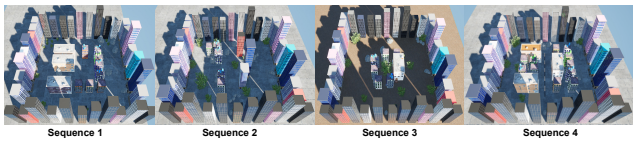

Figure 5: A glimpse of the Lidar-UrbanFly Dataset (LUF) Environment that we created. Train Data: Sequence (1,2, 3) and Test Data Sequence 4

To demonstrate the ability of the DEM to expose the underlying spatial structure and show the generalization of our method across point clouds with varying densities we choose three publicly available LiDAR datasets [**?**, 9, 32]. To further test the robustness of the method to 6-DOF motions we generate a synthetic dataset from a quadrotor.

(1) KITTI [9]: It consists of 11 sequences, similar to [3] we train on $05, 06, 07, 09$ and test on 00 and 08. **(2) Lidar UrbanFly Dataset *(LUF)*** : Using the *Unreal* Editor [7] we create a custom environment consisting of a buildings, trees and uneven roads to evaluate LDC methods. The environment is scanned by a 64 channel LiDAR mounted on a quadrotor in 6-DOF motion. We create four such environments as shown in Fig. 5, Sequence $(1, 2, 3)$ are used for training and 4 for testing. **(3) GPR** [32]: This dataset consists a total of 15, we use sequence $1, 2, 3, 4, 5, 6, 8, 9, 11, 12$ for training and report evaluation results on sequence 10 and 15. Similar to [3] we consider two point clouds to form a loop when the ground truth distance between them is less than $4m$. This rule allows us to sample triplets for training, an *anchor* and *positive* pair is formed when the distance between their poses is less than $4m$, *anchor* and *negative* pair is formed when distance is between $4m$ to $10m$.

5. Experiments and Results

We quantitatively and qualitatively demonstrate the ability of our method in performing 6-DOF LDC in this section. We tested our algorithm's robustness through challenging scenarios such as Loop Detection with a 6-DOF change in viewpoint and Loop Closure without any initial guess. Additionally, we integrate our pipeline with *LIO-SAM* [21] to measure its real-time efficacy. Finally, we conducted a detailed ablation study to test the efficacy of individual components. For more information and demonstration please check the supplementary.

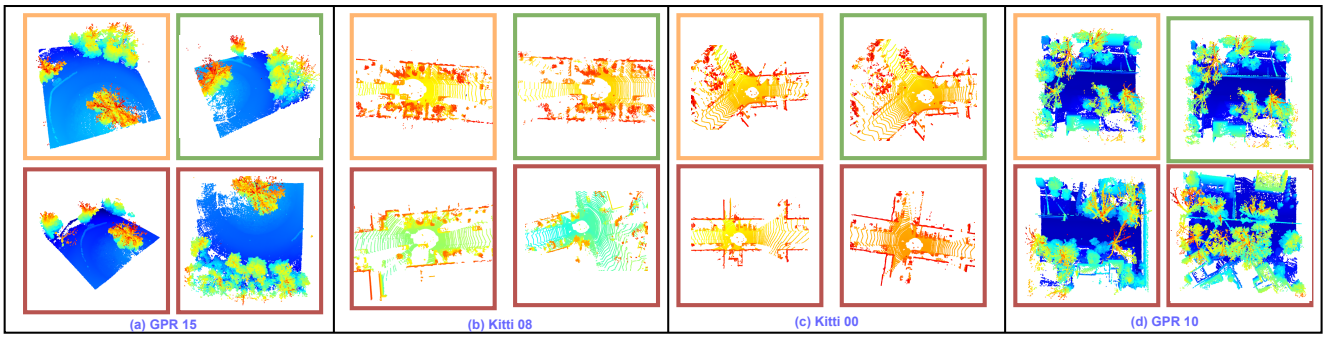

Figure 6: The figure depicts the recall of our method on various sequences. For each of the four sequences, the point cloud in the orange box (top left) is the query point cloud, and the one within the green box (top right) is the top retrieved one. The point clouds in the red box (second row) are the second and third retrieved point clouds (left to right). This results demonstrates the ability of *FinderNet* to learn spatial priors. Note that the top-retrieved results are correct in all cases.

5.1. Loop Detection Results

We benchmark against *LCDNet* [3], *PointNetVLAD* [23], *PCAN* [33] and *OverlapNet* [4] which are the SOTA deep learning based methods for loop detection, however they lack robustness to large view-point changes. Moreover, they are trained with significant augmentation which limits its generalization. We use the official code and pre-trained models released by the respective authors. For fairness in comparison we retrain the model on datasets for which the pretrained model was not available i.e the retrained models are $[3, 4, 17, 23, 28, 33]$ for *GPR*, $[23, 28, 33]$ for *Kitti*, and we retrain all models for the *LUF* environment. Additionally, we benchmark against ScanContext [13], a handcrafted feature based LDC method.

We employ *Average Precision (AP)* which is an effective metric for evaluating loop detection [3, 4]. To measure the *Average Precision (AP)*, we follow **Protocol 2** suggested by [3], which is proven to be an effective benchmarking metric. Here is a brief overview of the procedure: given a query point cloud A, we compare it to all the point clouds in the database B. For each pair of scans (A, B_i), we calculate their distance (lower distance implies greater similarity) using the method described in Section 3.4. If the distance is less than a fixed threshold, it is considered a loop. We then check the Euclidean distance between the ground truth poses of the LiDAR scans; if the poses are less than $4m$ apart, they are considered a *true positive*, while if the distance is greater than $4m$, they are considered as *false positive*.By varying this fixed threshold, we obtain multiple values of *precision-recall*, and its correponding results are presented in Table 2. *LCDNet* [3] is the SOTA LDC method on the *KITTI* dataset. We observe that on *KITTI08*, a challenging sequence which involves opposite views, our performance is better than *LCDNet* by approximately 8.4%. On the *KITTI00* sequence we are second best to SOTA (lower by approximately 1%). Similarly on both the *GPR* sequences $(10, 15)$ our method has the highest *AP* beating the closest *LCDNet* by approximately 11% and 4.5% respectively. On the *LUF* dataset, which consists of 6-DOF viewpoint changes our method outperforms *LCDNet* by 10%.

Fig. 6 presents the top 3 point clouds recalled by the DEM for a specific query to demonstrate its capability to learn the underlying spatial structure of point clouds.This demonstrates the ability of network to identify similar spatial structures. For example, In the *GPR15* sequence, the presence of common structural elements like trees in both the input and recalled point clouds suggests that the underlying geometry of point clouds has been exposed by the DEM.

5.2. Loop Closure Evaluation

In this section we compare the proposed point cloud registration method against three categories of algorithms: **(1) Loop Detection and only yaw estmation** approaches such as ScanContext [13], OverlapNet [4] only estimate the yaw and not the complete 6-DOF pose. **(2) Loop Detection and 6-DOF Pose Estimation:** LCDNet [3] is a SOTA method in the 6-DOF LDC task and we choose to compare against it. **(3) Only 6-DOF relative pose estimation**: We compare with the SOTA point cloud registration technique (they do not perform loop detection), TEASER++ [30]. We also benchmark against classical method, ICP [34].

The official code open-sourced by the authors is used to benchmark [3, 4, 13, 30] and we implement ICP using *Open3d* [35]. Our experimental results are shown in Table 3, we evaluate our method based on *Average translation error (ATE)* in meters and *Average Rotation Error (ARE)* in degrees. Our method has the lowest *ATE* on the *KITTI* dataset (both the sequences), *GPR10* sequence and the *LUF* dataset. It also has the lowest rotation error on *KITTI08, GPR10, LUF* and *GPR15*. On *KITTI00* sequence, the error of *FinderNet* is higher than *LCDNet* by 0.91^0. To improve results, we may use outlier-resilient robust ICP formulation

Integration with LIO-SAM: We tested our pipeline's ef-

Method	KITTI		GPR		*LUF*
	KITTI-00	KITTI-08	GPR-10	GPR-15	Seq-4
LCDNet [3]	**0.89**	0.76	0.82	0.88	0.69
OverlapNet [4]	0.61	0.22	0.75	0.57	NA
PointNetVLAD [23]	0.40	0.39	0.50	0.54	0.67
PCAN [33]	0.46	0.20	0.39	0.20	0.58
ScanContext [13]	0.49	0.20	0.66	0.62	NA
SOE-Net [28]	0.52	0.47	0.74	0.73	0.60
OverlapTransformer [17]	0.68	0.27	0.76	0.69	NA
Ours	0.88	**0.84**	**0.91**	**0.92**	**0.76**

Table 2: AP Comparison for loop detection. NA: Not applicable as [4] and [13] are only for SE(2) motions.

Method	KITTI		GPR		LUF
	KITTI-00	KITTI-08	GPR-10	GPR-15	Seq-4
LCDNet [3]	0.77/**1.07**	1.62/3.13	1.44/1.14	**0.50**/4.81	1.82/38.86
OverlapNet* [4]	−/3.6	−/65.29	−/7.85	−/6.25	-
Teaser++ [30]	2.93/16.13	3.24/28.98	2.68/16.87	2.47/20.34	2.05/44.12
ScanContext* [13]	−/1.89	−/3.20	−/4.37	−/4.26	-
ICP [34]	2.23/9.12	2.31/161.16	2.32/7.81	2.87/8.36	2.15/85.24
Ours	**0.72**/1.98	**1.35**/**2.96**	**0.95**/**0.85**	0.82/**1.14**	**1.78**/**35.15**

Table 3: Point Cloud Registration Comparison with SOTA. Result format: *TE(meters)/RE(degrees)*. "*" are algorithms that only estimate yaw, and not directly comparable with our 6-DOF method. "-" indicates not applicable as [4] and [13] are only for SE(2) motions.

ficacy by integrating it with LIO-SAM, a state-of-the-art LiDAR Inertial SLAM method. LIO-SAM is a method that utilizes a factor graph based backend optimization, and has been proven to provide reliable state estimates. The integration of *Lio-SAM* and *FinderNet* has been tested on the challenging *Kitti08* sequence. This sequence presents significant view-point changes, including opposite side (180^0) and 90^0 shifts. The results, depicted in Figure 7, demonstrate that employing FinderNet results in a 16% reduction in RMSE compared to the LIO-SAM's LDC (L2 distance based) [21]. Refer to the supplementary mate- rial for the implementation details and additional statistics.

Similar to *OverlapNet* [4] *ScanContext* [13] we utilize the geometry of the factor graph for the LDC task, for every new state x_{i+1} added to the factor graph a local area of $15m$ is searched and the closest subset of possible matches is recovered. Our pipeline robustly estimates a viewpoint invariant loop from these initial set of matches. We measure the distance (as explained in Section 3.5) between the query all the point clouds in the subset, if the sample with the closest distance is lesser than a fixed threshold it is considered as a loop and a new link would be added into the graph for optimization. The results are depicted in Figure 7, it demonstrates that employing *FinderNet* results in a 16% reduction in *RMSE* compared to the *LIO-SAM*'s LDC (L2 distance-based) [21]. Refer to the supplementary material for the details of experiments.

Ablation Study: To evaluate the performance of *RP Canonicalizer* and DYT, we conducted ablation studies. The presence of the *RP-Canonicalizer* provided two critical advantages. Firstly, it allowed our method to operate on point clouds in 6-DOF motion, as demonstrated in the LDC results on *LUF* dataset (Tab. 2 and Tab. 3). Secondly, in the absence of the canonicalizer, we had to rely on methods such as [30, 34] to estimate the 6-DOF relative pose. However, we observed from Tab. 3 our loop closure pipeline provided improved accuracy over [30, 34]. Furthermore, we independently study the performance of the *RP-Canonicalizer*, we record that the Coarse RP Canonicalizer had an error of 3.249/4.1582 (R/P) degrees, while the fine alignment had an error of 1.2598/1.352 (R/P) degrees.

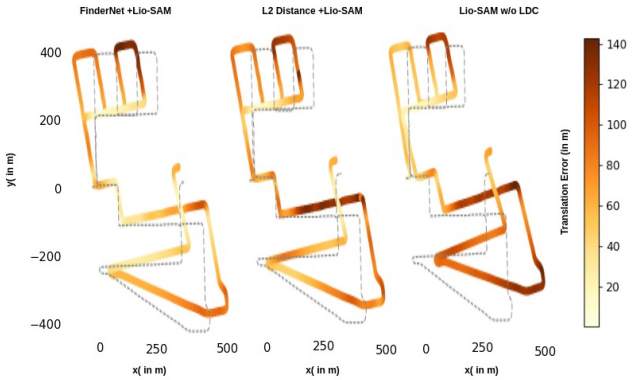

Figure 7: The figure depicts the translation error obtained by integrating multiple LDC methods with *LIO-SAM* [21] on kitti-08 sequences. The *RMSE* of translation error for the total trajectory without LDC (right) is $\mathbf{48.79m}$, if Euclidean Distance based LDC is used (full *LIO-SAM* center image) an *RMSE* of $\mathbf{35.69m}$ is observed. However, integrating *FinderNet* (left) the *RMSE* reduces to $\mathbf{29.96m}$. The dashed line represents the ground truth trajectory. Note: This plot depcits the top view of the trajectory where the drift in the *z-axis* (perpendicular to the ground plane) is not visible. However, the translation error is a function of all the three axis (x, y and z) and is shown as an intensity plot.

To analyze the importance of the DYT, we replaced it with the *Spatial Transformer* [10], which predicts the parameters for an affine transform. However, we observed that such a network led to poor performance and an *AP* of 0.0551, 0.0136, and 0.0818 for the *Kitti-00* [9], *LUF*, and *GPR* [32] datasets, respectively (the *AP* values for DYT based pipeline is shown in Table. 2). The DYT was able to estimates yaw with an error of 3.12 degrees.

A detailed explanation of the experimental procedure, and additional results related to the entire ablation study is available in the supplementary material.

6. Conclusion and Future Work

We develop a novel method for 6-DOF LDC for point clouds . Our approach utilizes canonicalization and DYT to achieve viewpoint invariance for large rotation angles. Furthermore, unlike the previous works our method does not require data-augmentation for training. *FinderNet* demonstrates significant improvement over SOTA on both real-world and simulated datasets. In future, we would like to perform resilient LDC on dynamic scenes.

References

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5297–5307, 2016.

[2] Igor Bogoslavskyi and Cyrill Stachniss. Fast range image-based segmentation of sparse 3d laser scans for online operation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 163–169. IEEE, 2016.

[3] Daniele Cattaneo, Matteo Vaghi, and Abhinav Valada. Lcdnet: Deep loop closure detection and point cloud registration for lidar slam. *IEEE Transactions on Robotics*, 2022.

[4] X. Chen, T. Läbe, A. Milioto, T. Röhling, O. Vysotska, A. Haag, J. Behley, and C. Stachniss. OverlapNet: Loop Closing for LiDAR-based SLAM. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[5] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 224–236, 2018.

[6] Juan Du, Rui Wang, and Daniel Cremers. Dh3d: Deep hierarchical 3d descriptors for robust large-scale 6dof relocalization. In *European Conference on Computer Vision*, pages 744–762. Springer, 2020.

[7] Epic Games. Unreal engine.

[8] Carlos Esteves, Christine Allen-Blanchette, Xiaowei Zhou, and Kostas Daniilidis. Polar transformer networks. *arXiv preprint arXiv:1709.01889*, 2017.

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[10] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

[11] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021.

[12] Giseop Kim, Sunwook Choi, and Ayoung Kim. Scan context++: Structural place recognition robust to rotation and lateral variations in urban environments. *IEEE Transactions on Robotics*, 2021.

[13] Giseop Kim and Ayoung Kim. Scan context: Egocentric spatial descriptor for place recognition within 3d point cloud map. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4802–4809. IEEE, 2018.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Kok-Lim Low. Linear least-squares optimization for point-to-plane icp surface registration. *Chapel Hill, University of North Carolina*, 4(10):1–3, 2004.

[16] Lun Luo, Si-Yuan Cao, Bin Han, Hui-Liang Shen, and Junwei Li. Bvmatch: Lidar-based place recognition using bird's-eye view images. *IEEE Robotics and Automation Letters*, 6(3):6076–6083, 2021.

[17] Junyi Ma, Jun Zhang, Jintao Xu, Rui Ai, Weihao Gu, and Xieyuanli Chen. Overlaptransformer: An efficient and yaw-angle-invariant transformer network for lidar-based place recognition. *IEEE Robotics and Automation Letters*, 2022.

[18] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[19] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[20] Lukas Schaupp, Mathias Bürki, Renaud Dubé, Roland Siegwart, and Cesar Cadena. Oreos: Oriented recognition of 3d point clouds in outdoor scenarios. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3255–3261. IEEE, 2019.

[21] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 5135–5142. IEEE, 2020.

[22] Kai Sheng Tai, Peter Bailis, and Gregory Valiant. Equivariant transformer networks. In *International Conference on Machine Learning*, pages 6086–6095. PMLR, 2019.

[23] Mikaela Angelina Uy and Gim Hee Lee. Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4470–4479, 2018.

[24] Ying Wang, Zezhou Sun, Cheng-Zhong Xu, Sanjay E Sarma, Jian Yang, and Hui Kong. Lidar iris for loop-closure detection. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5769–5775. IEEE, 2020.

[25] Louis Wiesmann, Tiziano Guadagnino, Ignacio Vizzo, Giorgio Grisetti, Jens Behley, and Cyrill Stachniss. Dcpcr: Deep compressed point cloud registration in large-scale outdoor environments. *IEEE Robotics and Automation Letters*, 7(3):6327–6334, 2022.

[26] Louis Wiesmann, Rodrigo Marcuzzi, Cyrill Stachniss, and Jens Behley. Retriever: Point cloud retrieval in compressed 3d maps. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2022.

[27] Louis Wiesmann, Andres Milioto, Xieyuanli Chen, Cyrill Stachniss, and Jens Behley. Deep compression for dense point cloud maps. *IEEE Robotics and Automation Letters*, 6(2):2060–2067, 2021.

[28] Yan Xia, Yusheng Xu, Shuang Li, Rui Wang, Juan Du, Daniel Cremers, and Uwe Stilla. Soe-net: A self-attention and orientation encoding network for point cloud based place recognition. In *Proceedings of the IEEE/CVF Conference*

on computer vision and pattern recognition, pages 11348–11357, 2021.

[29] Xuecheng Xu, Huan Yin, Zexi Chen, Yuehua Li, Yue Wang, and Rong Xiong. Disco: Differentiable scan context with orientation. *IEEE Robotics and Automation Letters*, 6(2):2791–2798, 2021.

[30] H. Yang, J. Shi, and L. Carlone. TEASER: Fast and Certifiable Point Cloud Registration. *IEEE Trans. Robotics*, 2020.

[31] Huan Yin, Xuecheng Xu, Sha Lu, Xieyuanli Chen, Rong Xiong, Shaojie Shen, Cyrill Stachniss, and Yue Wang. A survey on global lidar localization. *arXiv preprint arXiv:2302.07433*, 2023.

[32] Peng Yin, Shiqi Zhao, Ruohai Ge, Ivan Cisneros, Ruijie Fu, Ji Zhang, Howie Choset, and Sebastian Scherer. Alita: A large-scale incremental dataset for long-term autonomy. *arXiv preprint arXiv:2205.10737*, 2022.

[33] Wenxiao Zhang and Chunxia Xiao. Pcan: 3d attention map learning using contextual information for point cloud based retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12436–12445, 2019.

[34] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *International journal of computer vision*, 13(2):119–152, 1994.

[35] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing. *arXiv preprint arXiv:1801.09847*, 2018.