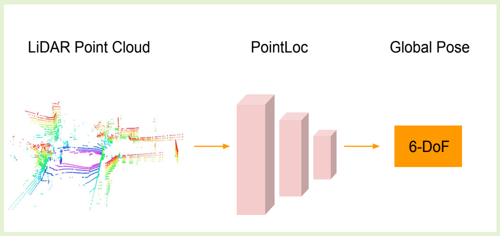
PointLoc: Deep Pose Regressor for LiDAR Point Cloud Localization

Wei Wang, Bing Wang, Peijun Zhao, Changhao Chen, Ronald Clark, Bo Yang, Andrew Markham, and Niki Trigoni

***Abstract*—In this paper, we present a novel end-to-end learning-based LiDAR sensor relocalization framework, termed PointLoc, which infers 6-DoF poses directly using only a single point cloud as input. Compared to visual sensor-based relocalization, LiDAR sensors can provide rich and robust geometric information about a scene. However, point clouds of LiDAR sensors are unordered and unstructured making it difficult to apply traditional deep learning regression models for this task. We address this issue by proposing a novel PointNet-style architecture with self-attention to efficiently estimate 6-DoF poses from 360° LiDAR sensor frames. Extensive experiments on recently released challenging Oxford Radar RobotCar dataset and real-world robot experiments demonstrate that the proposed method can achieve accurate relocalization performance.**



***Index Terms*—LiDAR Sensor Relocalization, LiDAR Point Cloud, Sensor Applications**

I. Introduction

LIDAR sensor-based localization methods have achieved impressive accuracy [1], [2]. For our motivating scenarios like indoor service robots, emergency rescue robots, and autonomous delivery service robots, several solutions have already mounted LiDARs on these robots for localization and obstacle avoidance. A typical LiDAR sensor localization system usually includes a feature extraction module, a feature matching algorithm, an outlier rejection step, a matching cost function, a spatial searching or optimization method and a temporal optimization or filtering mechanism [3]. Although these geometric localization methods achieve high accuracy in some scenarios, they require significant hand-engineering efforts to tune the huge amount of hyper-parameters, and depend heavily on the running environments.

A number of novel map-based approaches have been proposed to estimate global poses [3]–[5]. For example, Barsan and Wang *et al.* [4] proposed to learn descriptors from LiDAR intensity, and relocalization was performed by matching descriptors against pre-built intensity maps. However, although these methods achieve considerable accuracy, it is hard to build the map and the computing complexity increases greatly when the map gets larger. Additionally, they require other systems to provide an accurate initial pose as coarse localization first.

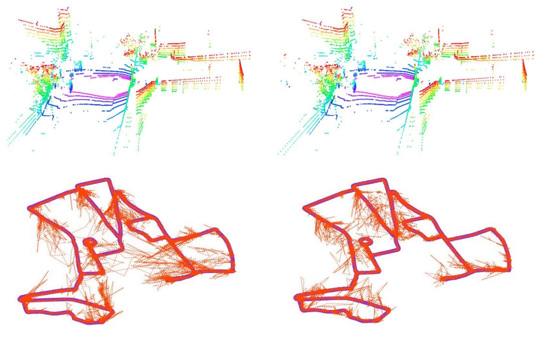


Fig. 1: PointLoc results for the challenging Oxford Radar RobotCar dataset [6], [7]. We directly feed a point cloud of the LiDAR sensor from a single timestamp to the neural network for predicting the 6-DoF pose without the requirement of pre-built maps. The estimations of PointLoc are robust regardless of weather, and outperform the state-of-the-art DNN-based LiDAR and visual sensor relocalization methods significantly.

Thus, most localization solutions employ Global Navigation Satellite System (GNSS) to provide pose estimations. Unfortunately, GNSS is not always available such as in indoor environments and the accuracy of GNSS cannot be guaranteed in areas like large cities where high-rising buildings can block the GNSS signals. To this end, Uy *et al.* [8] proposed a point cloud retrieval-based localization method to deal with the situation when the GNSS is absent. It obtains a 6-DoF pose with respect to the pre-built map in the form of

This work is funded by the NIST under Grant 70NANB17H185.

Wei Wang, Bing Wang, Peijun Zhao, Changhao Chen, Bo Yang, Andrew Markham, and Niki Trigoni are with the Department of Computer Science, University of Oxford, UK (e-mail: {firstname.lastname}@cs.ox.ac.uk).

Ronald Clark is with the Department of Computing, Imperial College London, UK (e-mail: ronald.clark@imperial.ac.uk).

reference database. Dube *et al.* [9] further proposed SegMap to improve the storage efficiency of the reference database by storing data-driven descriptors of individual objects in point clouds of LiDAR sensors. However, retrieval-based approaches inherently suffer from several issues. First, the time complexity of finding the closest match between the query point cloud and the reference point cloud is $O(\mathrm{n})$ where n is the number of point clouds, which is not suitable for common real-time application scenarios. Second, the point cloud-based method requires a reference database, which occupies $O(\mathrm{n})$ storage space and cannot be deployed on many mobile robots. Third, the recall rate of it is often not good enough [8].

Recently, learning-based approaches have emerged as a promising tool to build up a completely end-to-end localization system. These methods do not require any reference databases during runtime, and the learned features tend to be general and robust. These kind of localization approaches train a neural network to directly predict the pose. Their time complexity during inference time is $O(1)$ and the space it occupies is only the model size, which addresses the drawbacks of point cloud retrieval-based methods. Early attempts in this direction include PoseNet and its variations [10]–[12]. However, all the current pose regression approaches utilize RGB images of visual sensors as inputs, which have several problems. Visual sensors are sensitive to the change of environments, resulting in suboptimal localization performance. In addition, the input images are restricted to a narrow Field-of-View (FoV). These aspects restrict the application of these approaches to the real world. Compared with RGB images of visual sensors, point clouds, acquired by LiDAR sensors, capture 360° 3-D space, and provide much richer geometric information of a specific location. In addition, the features extracted from point clouds tend to be more robust compared to those extracted from images. However, point clouds of LiDAR sensors are unordered and unstructured, making it difficult to learn features for localization. Motivated by this, we design a neural network to use LiDAR point clouds as input for robust and accurate localization.

In this paper, we propose a novel neural network-based 3-D pose regressor, named PointLoc, to accurately estimate the 6-DoF pose using point clouds of LiDAR sensors. The neural network directly takes a primitive point cloud as input and estimates the 6-DoF pose in an end-to-end fashion. The performance shows significant improvement over the learning-based LiDAR and visual sensor relocalization methods. Fig.1 illustrates the superior performance of our PointLoc approach in different environments found in the Oxford Radar RobotCar dataset.

In summary, our contributions are as follows:

- To the best of our knowledge, this is the first LiDAR sensor-based approach for deep global pose regression in an end-to-end fashion. Our proposed architecture with a self-attention module can further improve the accuracy of the predicted 6-DoF absolute poses.
- We conduct real-world robot experiments in an indoor environment. We collect and create a new indoor LiDAR-visual sensors dataset, dubbed vReLoc, and release it for studying the indoor relocalization task.
- Comprehensive experiments and an ablation study on these two new datasets have been done to evaluate our proposed method. Results demonstrate that the PointLoc model outperforms the state-of-the-art DNN-based LiDAR and visual sensor relocalization methods by a large margin.

The rest of the paper is organized as follows. Section II introduces visual sensor relocalization, learning-based localization systems, DNN-based LiDAR odometry and deep learning on LiDAR point clouds. Problem formulation is given in Section III. The detailed LiDAR sensor relocalization method is illustrated in Section IV. Section V explains the collected indoor dataset vReLoc. Experiments and ablation studies are presented in Section VI. Section VII concludes the work and give the future research directions.

II. RELATED WORK

In this section, we review different learning-based approaches for localization, LiDAR odometry which estimates ego-motions between consecutive point clouds, and DNN architectures on point clouds.

A. Visual Sensor Relocalization

For dealing with the drawbacks of map registration methods, recent works propose learning-based approaches to estimate the global pose directly [10]–[16]. They take images, either single or sequential, as inputs to train a neural network model for predicting absolute poses. The key to these methods is to learn a deep pose regressor, which usually comprises a feature extractor and a regressor [10], [16], [17]. For example, PoseNet related works [10], [11], [18] proved the feasibility of predicting the global pose using a single RGB image by regressing the pose directly. Brahmbhatt *et al.* [13] utilized the relative pose between two images as a geometric constraint to estimate the pose. Although DNN-based relocalization methods can solve the downsides of retrieval-based approaches, the performance of translation and rotation estimation is still not satisfactory enough to be applied to real-world scenarios [19], which calls for further work on learning algorithms. Our work follows this line of study, aiming to improve the accuracy of deep global pose regression with LiDAR sensors.

B. Learning-based Localization Systems

Learning-based localization systems have gained significant interests recently. Almalioglu *et al.* [20] employed recurrent neural network for robust MMWave radar-based ego-motion estimation. CellinDeep [21] adopted DNN to capture the non-linear relationship between the cellular signal and its location for robust and accurate indoor localization. Alshamaa *et al.* [22] proposed a decentralized kernel algorithm for sensor localization in indoor wireless environments. Silva *et al.* [23] applied transfer learning and machine learning to images of a Kinect sensor for the localization of mobile robots. Hoang *et al.* [24] proposed a semi-sequential probabilistic method to improve the performance of the indoor localization with extensive on-site experiments. Li *et al.* [25] developed a centralized indoor localization method using pseudo-label along with federated learning for the improved indoor localization. AdapLoc [26] utilized the CNN and domain adaptation for the device-free WiFi localization in dynamic environments. In contrast, our work proposes to apply deep learning to LiDAR sensors for global localization.

C. DNN-based LiDAR Odometry

Recent works propose learning-based methods to estimate LiDAR Odometry, which calculates relative poses between consecutive LiDAR scans. Wang *et al.* [27] proposed a deep parallel neural network to directly predict relative poses. Li *et al.* [28] developed a learning-based fusion framework with 2-D LiDAR and IMU sensors for odometry estimation. Horn *et al.* [29] developed a flow embedding approach to solve the fusion problem of point clouds for LiDAR Odometry. 3DFeat-Net [30] was developed to learn both 3-D feature detector and descriptor for point cloud matching using week supervision. Lu *et al.* [31] proposed a Virtual Corresponding Points method to align two point clouds accurately. Wang *et al.* [32] designed novel sub-network architectures to address difficulties in the ICP method. These point cloud registration approaches can be leveraged to predict LiDAR Odometry. However, different from these methods, our work focuses on LiDAR relocalization, which estimates global poses rather than relative poses. Fig. 2 illustrates the difference between these two localization tasks. LiDAR odometry estimates relative poses between consecutive point clouds, while LiDAR relocalization predicts absolute poses w.r.t the world coordinate.

D. Deep Learning on Point Clouds

DNN-based feature extraction methods for point clouds [33]–[36] have gained significant success in recent years. VoxelNet [37] was developed to learn feature embeddings in voxels for object detection. PointNet++ related works [33], [34], [36] have been proposed to directly process unordered point sets and learn features from points, which showed impressive performance on tasks of 3-D object detection, part segmentation, and semantic segmentation. Detailed introductions and applications of deep learning for point clouds can be found in the recent survey paper [38].

III. Problem Statement

We design a DNN-based framework for performing deep global pose regression using point cloud data from a LiDAR sensor, which is LiDAR relocalization. We predict the absolute 6-DoF poses of the mobile agent within previously visited areas. A typical use case for our method would be when a mobile agent has already visited the query places before, and then has to localize itself again when it moves across the previously-visited environment. To enable a more generic and reliable relocalization system, we only consider one LiDAR sensor input at a single timestamp rather than sequential inputs.

For each timestamp t, the agent receives one point cloud frame $\mathbf{P}_t = \{\mathbf{x}_i \mid i = 1, ..., N\}$ from LiDAR, where each point

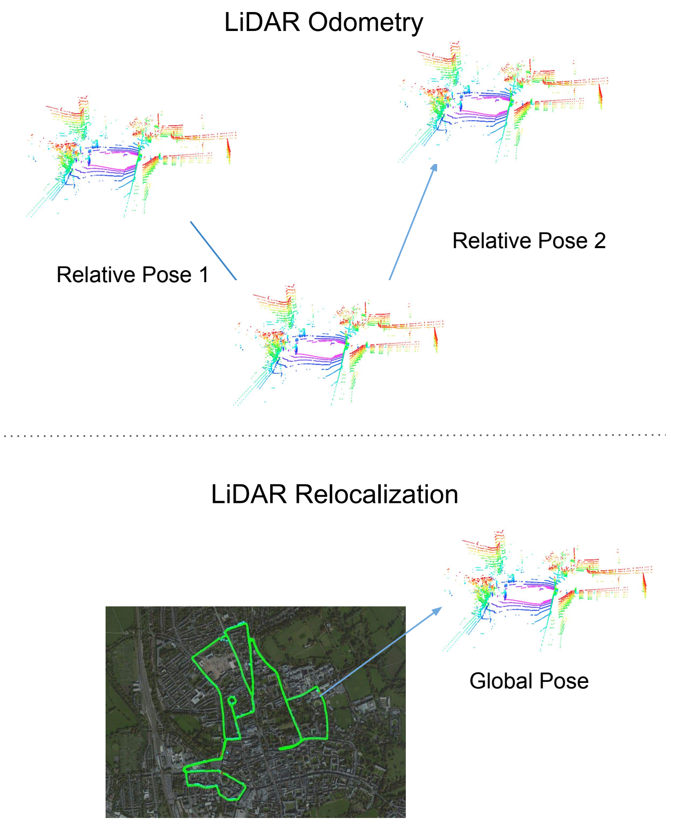


Fig. 2: The difference between LiDAR odometry and LiDAR relocalization [6], [7]. LiDAR odometry estimates relative poses between consecutive point clouds, which produces accumulative drifts over time, while LiDAR relocalization predicts absolute poses w.r.t the world coordinate, which requires agents previously traverse scenes. These are two different tasks in localization, and this work focuses on the LiDAR relocalization.

$\mathbf{x}_i$ is a vector of describing its coordinate (x, y, z). Therefore, the shape for each $\mathbf{P}_t$ is $(N, 3)$. The relocalization of the agent is parameterized by a 6-DoF pose $[\mathbf{t}, \mathbf{r}]^T$ with respect to the world coordinate, where $\mathbf{t} \in R^3$ is a 3-D translation vector and $\mathbf{r} \in R^4$ is a 4-D rotation vector (quaternion). To this end, deep 3-D pose regressors learn a function $\mathcal{F}$ such that $\mathcal{F}(\mathbf{P}_t) = (\mathbf{t}, \mathbf{r})^T$, where the function $\mathcal{F}$ is usually a neural network for DNN-based methods.

IV. Deep Point Cloud Relocalization

This section introduces our proposed PointLoc, a deep 3-D pose regressor for predicting the global pose from LiDAR sensors. The overall architecture is illustrated in Fig. 3. Our system consists of point cloud pre-processing, a point cloud encoder, a self-attention module, a group all layers module, and a pose regressor. The point cloud data are down-sampled to a fixed shape $(N, 3)$ as an input. The whole design is based on the PointNet-style structure, which can theoretically learn a critical subset of points for relocalization. We introduce each module individually.

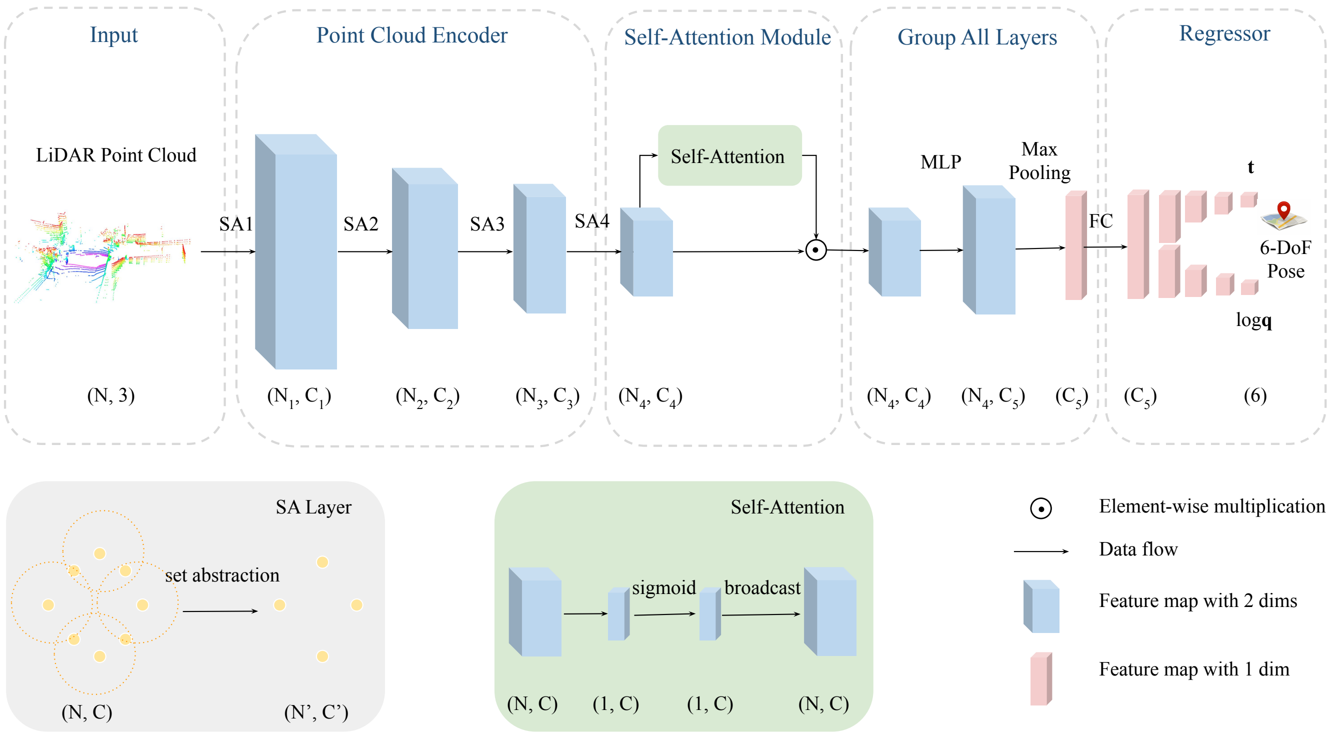


Fig. 3: The architecture of PointLoc. It consists of a point cloud encoder, a self-attention module, a grouping all (GA) layers module and a pose regressor. Numbers in the parentheses represent the dimensions of feature tensors. (N, 3) represents the 3-D coordinates x, y and z of a point cloud. The encoder is composed of 4 consecutive set abstraction (SA) layers. Each SA layer shown in the diagram [39] consists of a sampling layer, a grouping layer and a PointNet layer [34]. For more details about the SA layer, please refer to PointNet++ [34] and FlowNet3D [39]. The learnt point features are sent to self-attention module for eliminating the noisy features. Afterwards, these features are fed into group all (GA) layers for down-sampling to a feature vector. Finally, the pose regressor predicts the 6-DoF pose.

A. Point Cloud Pre-Processing

The purpose of this module is to pre-process raw point clouds to fit into the neural network. Each point cloud frame of a LiDAR sensor scan contains a different number of points. However, our neural network requires the same point cloud dimensions $(N, 3)$ for its inputs. To tackle this problem, we adopt the random point cloud sampling strategy. We ensure that all the point cloud inputs have the same shape $(N, 3)$. N is set to 20,480 in this work since the average number of points in a point cloud of the Radar Robotcar Dataset [6], [7]in our experiments is around 21,000 and we want to keep the information as much as possible.

B. Point Cloud Encoder

The goal of this module is to extract features from the point cloud. The feature representation extracted by the point cloud encoder plays a critical role in achieving accurate and reliable relocalization. Intuitively, human beings can utilize key points and features in a scene to identify where they are and conventional geometric methods are capable of performing precise localization by exploiting key points of the point cloud data. Inspired by this, if a neural network learns a subset of key points from the original point cloud data relevant to the localization task, we can take better advantage of these key features to identify a location. Existing literature [34], [40] has proved the critical-subset theory, i.e. for any point cloud $\mathbf{P}$, a PointNet-like structure can identify a salient point subset $\mathbf{C} \subseteq \mathbf{P}$, making it a desirable choice for our relocalization task.

Specifically, PointNet exploits the multi-layer perceptron (MLP), feature transformation module, and max pooling layer to approximate a permutation invariant function for point cloud classification and segmentation. In fact, it is a universal continuous set function approximator, described as:

$$f(x_1, ..., x_N) = \phi(\mathbf{MAX}\{h(x_i) \mid x_i \in \mathbf{P}\}) \qquad (1)$$

where ϕ and h are two continuous functions (they are usually instantiated to be an MLP), and **MAX** denotes the max pooling layer [33]. PointNet++ extends PointNet by recursively capturing the hierarchical features on point sets in a metric space [33]. From the aforementioned Eq. 1, the result of the PointNet structure is determined by $u = \mathbf{MAX}\{h(x_i) \mid x_i \in \mathbf{P}\}$, and the **MAX** operation takes N vectors as input and outputs one vector of element-wise maximums. Thus, there exists one $x_i \in \mathbf{P}$ such that $u_j = h_j(x_i)$, where u_j is the j^{th}dimension of u, and μ_j is the j^{th} dimension of $h(x_i)$. These points can be aggregated into a critical subset $\mathbf{C} \subseteq \mathbf{P}$, where $\mathbf{C}$ determines u and then $\phi(u)$ (more details can be found in [34], [40]).

Consequently, the critical-subset theory is applicable to neural networks of the structure of $\phi(\mathbf{MAX}\{h(x_i) \mid x_i \in \mathbf{P}\}$. The proposed PointLoc is built upon PointNet++, consisting of such a structure and thus can learn the critical subset from point clouds of LiDAR sensors in theory.

We design our point cloud encoder based on the set abstraction (SA) layer of PointNet++ [33], [36]. The point cloud encoder is composed of 4 consecutive SA layers. Each SA layer is composed of a sampling layer, a grouping layer and a PointNet layer [34]. The SA layer takes a feature matrix $\mathbf{F} \in R^{N \times C}$ as input where N is the point number and C is the feature dimension of each point, and outputs a feature matrix $\mathbf{F}^{'} \in R^{N^{'} \times C^{'}}$ where $N^{'}$ is the sub-sampled point number and $C^{'}$ is the new feature dimension of each point (from the size (N_1, C_1) to (N_4, C_4) in Fig. 3). We also a leverage multi-scale grouping strategy [33] inside the SA layer for robust feature learning. Specifically, the layer adopts farthest point sampling to sample $N^{'}$ regions with x_j being the region centers, and for each region with radius r, it extracts local features with a symmetric function as [39]:

$$\mathbf{F}_{\mathbf{j}}^{'} = \mathbf{MAX}_{\{i \mid ||x_i - x_j|| \leq r\}}\{h(\mathbf{F}_i, x_i - x_j)\} \tag{2}$$

where $\mathbf{F}_i$ is the i^{th} row of $\mathbf{F}$, $\mathbf{F}_j^{'}$ is the j^{th} row of $\mathbf{F}^{'}$, $h : R^C \rightarrow R^{C^{'}}$ is the MLP, and $\mathbf{MAX}$ is the max pooling layer.

C. Self-Attention Module

The aim of this module is to remove outliers like moving objects from the previous extracted features for better relocalization performance. Prior works [16], [17] have proved that the self-attention mechanism can improve visual sensor relocalization by removing noisy features. Therefore, we also design a neural module to automatically remove the dynamic features before regressing the final poses. Inspired by the recent works [16], [17], [41], we introduce a self-attention module to learn a mask, which attempts to remove outlier features of moving objects from the original point features by conducting the element-wise dot product between the point features and the mask.

Given a set of point features $\mathbf{F} \in R^{N \times C}$ which are learned from the point cloud encoder, our attention module aims to learn a mask $\boldsymbol{M} \in R^{1 \times C}$ for the features $\mathbf{F}$. To achieve this, we use a shared MLP followed by a $sigmoid$ function to take the features $\mathbf{F}$ as input and then directly generate the mask $\boldsymbol{M}$. After that, we broadcast and mask the features $\mathbf{F}$ by $\boldsymbol{M}$, obtaining weighted features $\hat{\mathbf{F}}$ for subsequent pose regression. Specifically, since the dimensions of the point features $\mathbf{F}$ is $N \times C$ and the dimension of the learned mask $\mathbf{M}$ is $1 \times C$, we broadcast the dimensions of the mask from $1 \times C$ to $N \times C$. Afterwards, we mask the features $\mathbf{F}$ by the broadcasted mask $\mathbf{M}$ via conducting the element-wise dot product in order to remove noisy features of the point features $\mathbf{F}$. Formally, this self-attention module is defined as follows:

$$\hat{\mathbf{F}} = \mathbf{F} \cdot \boldsymbol{M} \tag{3}$$

where dot means element-wise dot product between $\mathbf{F}$ and $\mathbf{M}$.

D. Group All Layers Module

The target of this module is to aggregate features from all previous layers to generate an embedded feature vector. Specifically, shown in Fig. 3, the input of the group all layers (GA) module is a point feature set of size $N_4 \times C_4$, and then the point features are propagated to an updated point feature set of size $N_4 \times C_5$ via MLP, where C_5 is larger than C_4. Next, it is down-sampled to the C_5 dimension feature vector through the max pooling layer. The embedded feature vector is then forwarded to an FC layer. After the FC layer, the C_5 dimensional feature vector is finally sent to the pose regressor for predicting the translation $\mathbf{t}$ and rotation $\mathbf{r}$ respectively.

E. Pose Regressor

The purpose of this module is to predict the ultimate pose. After the FC layer, the C_5 dimensional feature vector from the previous module is finally sent to the pose regressor for predicting the translation $\mathbf{t}$ and rotation $\mathbf{r}$ respectively. The pose regressor is composed of two branches of consecutive fully-connected (FC) layers. Each branch consists of 4 fully connected (FC) layers. The sizes of FC layers decrease gradually to learn features. We choose Leaky Relu as the activation function after each FC layer except for the last FC layer. The last FC layers of these two branches regress the translation and rotation separately.

F. Loss Function

Our goal is to estimate the 6-DoF pose $[\mathbf{t}, \mathbf{r}]^T$. Prior works [10]–[12], [42] directly predict quaternions and use an l_1 or l_2 loss, but such a representation is over-parameterized and normalization of the output quaternion is required at the cost of worse accuracy [13]. Odometry tasks with DNNs [27], [43] usually regress Euler angles, which are also not suitable here since they wrap around 2π. Consequently, we employ the definition of the loss function in [13] for training our neural network, which is adapted from [11]. Given K training samples $\mathcal{G} = \{\mathbf{P}_t \mid t = 1, ..., K\}$ and their corresponding ground-truth poses $\{[\hat{\mathbf{t}}, \hat{\mathbf{r}}]_t^T \mid t = 1, ..., K\}$, the parameters of the PointLoc are learned via the following loss function:

$$\mathcal{L}(\mathcal{G}) = \|\mathbf{t} - \hat{\mathbf{t}}\|_1 e^{-\beta} + \beta + \|\log \mathbf{q} - \log \hat{\mathbf{q}}\|_1 e^{-\gamma} + \gamma \tag{4}$$

where β and γ are balanced factors to jointly learn translation and rotation. It is worth noting that the β and γ are learnable factors during training, which are initialized by β^0 and γ^0 respectively. $\log \mathbf{q}$ is the logarithmic form of a unit quaternion $\mathbf{q} = (u, \mathbf{v})$, where u is a scalar and $\mathbf{v}$ is a 3-D vector. It is defined as:

$$\log \mathbf{q} = \begin{cases} \frac{\mathbf{v}}{\|\mathbf{v}\|} \cos^{-1} u, & \text{if } \|\mathbf{v}\| \neq 0 \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{5}$$

V. Indoor LiDAR Sensor Dataset for Relocalization

There is a lack of public datasets in the indoor environment with LiDAR sensors. In order to boost the research in this area, we collected a new dataset dubbed vReLoc with rich

TABLE I: Dataset Descriptions on the vReLoc.

Sequence	Scenario	Training	Test
Seq-03	static	✓	
Seq-12, Seq-15	one-person walking	✓	
Seq-16	two-persons walking	✓	
Seq-05, Seq-06, Seq-07	static		✓
Seq-14	one-person walking		✓

sensor modalities, e.g. vision and LiDAR sensors on a mobile robot platform. Our dataset has been released online to benefit future researchers[1].

The experimental robot is Turtlebot 2, mounted with a Velodyne HDL-32E LiDAR sensor and an Intel RealSense Depth Camera D435. The sensors have been carefully calibrated. The Velodyne is a lightweight pulsed laser for Detection and Ranging, which features 32 lasers across over a $40°$ vertical field-of-view and a $360°$ horizontal field-of-view. It runs at a frequency of 10Hz. Each point cloud in the dataset contains $\sim$60,000 points. The camera was employed to capture RGB images, and the size of each image is $640 \times 480 \times 3$. A Vicon Motion Tracker system is leveraged for acquiring accurate ground truth 6-DoF poses. 10 Bonita B10 cameras are used in the system, installed around the area where the dataset is collected. Each Bonita B10 has the resolution of 1 megapixel with 250 fps frame rate, and an operating range of up to 13 m. The system can track the pose of the robot at a precision of $\sim$1cm.

The size of the Vicon room is about $4m \times 5m$. We lay out several obstacles in the scene. For the relocalization task, the scene is fixed through the whole data collection process. We utilized the Robot Operating System (ROS) for robot control and data collection. Timestamps were recorded on every frame of each sensor by the ROS, and we synchronized world timestamp across different systems from the same Network Time Protocol (NTP) server.

A total of 18 sequences were collected of various lengths. Since the Velodyne LiDAR, RealSense camera and Vicon motion tracker system run in different frequencies, we synchronized these systems so that the image of the visual sensor and the point cloud of the LiDAR sensor in each timestamp has the same 6-DoF pose. For the static scenario, there are no moving objects in the scene. For other scenarios, there are people randomly walking in the scene. Sequences 01-10 come from the static environment, sequences 11-15 are the one-person moving scenario, and sequences 16-18 are two-persons moving scenario. In order to better represent real-world situations, in our experiments, we specially chose challenging sequences as the training dataset. We report our training and test sequences from the vReLoc dataset in Table I.

VI. Experiments

In this section, we evaluate our proposed approach on the recently released outdoor Oxford Radar RobotCar [6], [7] dataset and our proposed indoor vReLoc dataset and compare to state-of-the-art methods.

[1]https://github.com/loveoxford/vReLoc

A. Implementation Details

Adam [44] is applied to train our network with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We set the initial values $\beta_0 = 0.0$ and $\gamma_0 = -3.0$ of the loss function following MapNet [13] and AtLoc [17]. From our experiment, if we change these values within a short range, the results remain almost the same, which is reasonable since these two parameters are learnable and they will adjust themselves to different values during training phase. The learning rate is set to 0.001, and we train 100 epochs on both datasets. For baseline image approaches, we also used data augmentation to improve the accuracy of predictions. Following the convention of existing works [10], [11], [13], [17], we calculate the mean error for outdoor datasets and the median error for indoor datasets.

The parameter settings of the point cloud encoder is shown in Table II. For the parameters of point cloud encoder, especially parameters of the set abstraction layers, we employ the parameter settings from PointLoc++ and VoteNet since these settings have been demonstrated effective in point cloud-related tasks for feature extraction. In Table III, we present the parameter settings of the self-attention module. The parameter setting of the group all layers is shown in Table IV. We show the parameter setting of the pose regressor in Table V. The neural network was trained on 2 NVIDIA TITAN V GPUs on a GPU server with 100 epochs, and it takes around 33 minutes to train the neural network for 1 epoch. The batch size is 32, and it consumes around 5.5 G of memory during training.

TABLE II: Parameter Setting of the Point Cloud Encoder.

Layer Name	Point Num	Radius	Sample Num	MLP
SA1	2048	0.2	64	[0, 64, 64, 128]
SA2	1024	0.4	32	[128, 128, 128, 256]
SA3	512	0.8	16	[256, 128, 128, 256]
SA4	256	1.2	16	[256, 128, 128, 256]

TABLE III: Parameter Setting of the Self-Attention Module.

Layer Name	Point Dimension	Feature Dimension
Self-Attention Module	256	256

TABLE IV: Parameter Setting of the Group All Layers.

Layer Name (Ignore Max Pooling Layer)	Feature Dimension
Multi-Layer Perceptron (MLP)	[256, 256, 512, 1024]
Fully-Connected Layer (FC)	1024

TABLE V: Parameter Setting of the Pose Regressor. $\mathbf{t}$ branch and $log\mathbf{q}$ branch share the same parameter setting.

Layer Name	Feature Dimension	LeakyReLu
FC1 + LeakyReLu	[1024, 512]	0.2
FC2 + LeakyReLu	[512, 128]	0.2
FC3 + LeakyReLu	[128, 64]	0.2
FC4	[64, 3]	NA

TABLE VI: Dataset Descriptions on the Oxford Radar RobotCar.

Scene	Time	Tag	Training	Validation	Test
FULL1	2019-01-11-14-02-26	sun	✓		
FULL2	2019-01-14-12-05-52	overcast	✓		
FULL3	2019-01-14-14-48-55	overcast	✓		
FULL4	2019-01-18-15-20-12	overcast	✓		
FULL5	2019-01-15-14-24-38	overcast		✓	
FULL6	2019-01-10-11-46-21	rain			✓
FULL7	2019-01-15-13-06-37	overcast			✓
FULL8	2019-01-17-14-03-00	sun			✓
FULL9	2019-01-18-14-14-42	overcast			✓

B. Baselines

To validate the performance of the proposed PointLoc, we compare it with several state-of-the-art learning-based open-source LiDAR sensor localization and visual sensor relocalization approaches. For LiDAR sensor localization approaches, we choose PointNetVLAD [8] and Deep Closest Point (DCP) [32]. PointNetVLAD is a large-scale point cloud retrieval-based approach, which can be utilized for LiDAR sensor relocalization. We create the triplet training dataset, increase the point number from 4,096 to 8,192, and set the loss margin from 0.5 to 1.0 to improve the performance, while other hyper-parameters are kept the same as the vanilla PointNetVLAD. The validation sequence FULL 5 is chosen for building up the reference database as the localization map. DCP is a DNN-based point cloud registration approach, which employs the PointNet [34] and DGCNN [45] as the embedding network. Although we deal with different tasks, we can adapt it for relocalization. Specifically, the DCP aims at the task of point cloud odometry, which demonstrates that the feature extraction design of this neural network is effective. Our task is to estimate global 6-DoF poses (relocalization) from point clouds. Therefore, we utilized the feature extraction module of DCP in our design to compare the performance. For the visual sensor relocalization baselines, we choose PoseNet17 [11] since it outperforms PoseNet and Bayesian PoseNet in previous works. AtLoc [17] is selected for comparison since it is the state-of-the-art single image-based learning approach. We also choose LSTM-Pose [15] as the sequential baseline. Moreover, we also compare with MapNet [13] because it is the state-of-the-art sequential visual sensor approach. We note that sequential methods generally perform better than single image ones by utilizing time constraints. However, for the relocalization task, this past information is not always available as discussed before. We still compare with them to examine how competitive our method is. We note that we implement baseline methods and tune them for the best performance.

C. Results on the Oxford Radar RobotCar

The Oxford Radar RobotCar dataset [6] is a radar extension to the Oxford RobotCar dataset [7], providing data from dual Velodyne HDL-32E LiDARs and Grasshopper2 monocular cameras. The ground truth poses are obtained by a NovAtel SPAN-CPT ALIGN inertial and GPS navigation system (GPS/INS).

Dataset Description The data were gathered in January 2019 over thirty-two traversals of a central Oxford route, and the duration and distance of each traversal are $\sim$32mins and $\sim$9.05km respectively. The resolution of a captured RGB image is 1280×960, and each point cloud has $\sim$21,000 points. We observe that the dataset is large-scale, covers various weather conditions and has moving objects like people and cars in the scenes, all of which have significant influence on the accuracy of relocalization task, and therefore it is quite challenging. Since there is a timestamp misalignment between camera and LiDAR sensors, we synchronize timestamps with scripts and interpolate (GPS/INS) measurements to coincide with the ground truth poses. For time synchronization, the FPS (Frames per Second) of camera is 16HZ and the FPS of LiDAR is 20.02 HZ. Therefore, for every timestamp of the camera images, we collected the corresponded point cloud by searching the closest timestamp from the LiDAR point clouds. Like MapNet and AtLoc, the missing GPS/INS data is handled by interpolating values from visual odometry data which is provided by the Radar RobotCar. We report the training and test sequences we used from the Oxford Radar RobotCar in Table VI.

Results The test results of the Radar RobotCar are presented in Table VII. Following the plotting style of relocalization work [16], the trajectories of FULL7 and FULL8 of DCP, MapNet, AtLoc, and PointLoc are shown in Fig. 4. The PointLoc improves the LiDAR point cloud retrieval-based approach PointNetVLAD by 46.47% in translation and 70.45% in rotation, which proves the effectiveness of our proposed method. As seen from Table VIII, PointLoc can satisfy real-time operation and the storage space is small. The inference time is around 0.1sec, which means that by using this system, the real-time localization is achievable at 10 Hz. This indicates that PointLoc is better than existing point cloud retrieval-based approaches for relocalization. Moreover, The PointLoc improves the DCP with PointNet by 29.46% in translation and 37.80% in rotation, which reveals that the proposed embedding neural network can effectively learn meaningful features for relocalization. For DCP with DGCNN, the whole neural network is difficult to train and requires large computational resources due to the large number of points and the essence of the graph architecture. Furthermore, the proposed PointLoc consistently outperforms the camera relocalization baselines by a large margin. For the best performance of deep camera relocalization, the PointLoc improves the AtLoc by 66.86% in translation and 78.83% in rotation. These results demonstrate that instead of utilizing RGB images as the sensory input, LiDAR point cloud can significantly improve the relocalization accuracy. Our pose predictions are even better than the sequential approaches like MapNet. In addition, the variance of learning-based camera relocalization is much larger than our approach. Therefore, the PointLoc can have stable estimation across the test dataset, which indicates that the point cloud relocalization method is more robust than the visual relocalization.

D. Results on a Real-World Indoor Robot

We also validate our proposed PointLoc on the real-world indoor LiDAR-visual sensors dataset. Our experimental design

TABLE VII: Mean translation error (m) and rotation error (°) for the state-of-the-art methods on the Oxford Radar RobotCar. For PointNetVLAD and DCP, the type of input is LiDAR point clouds. The sensory data type of PoseNet17, LSTM-Pose, MapNet, and AtLoc is camera RGB image.

Scene	PointNetVLAD [8]	DCP [32]	PoseNet17 [11]	LSTM-Pose [15]	MapNet [13]	AtLoc [17]	PointLoc (Ours)
FULL6	28.48m, 5.19°	18.45m, 2.08°	51.05m, 6.41°	38.47m, 5.36°	32.16m, 5.40°	28.57m, 7.99°	**13.81m, 1.53°**
FULL7	17.62m, 3.95°	14.84m, 2.17°	80.29m, 6.51°	54.59m, 4.56°	47.79m, 5.44°	36.86m, 5.17°	**9.81m, 1.27°**
FULL8	23.59m, 5.87°	16.39m, 2.26°	111.24m, 12.78°	77.57m,9.74°	51.93m, 7.73°	58.63m, 8.53°	**11.51m, 1.34°**
FULL9	13.71m, 2.57°	13.60m, 1.86°	45.50m, 3.96°	26.16m, 2.56°	14.93m, 2.84°	10.67m, 2.88°	**9.51m, 1.07°**
Average	20.85m, 4.40°	15.82m, 2.09°	72.02m, 7.42°	49.20m, 5.49°	36.70m, 5.35°	33.68m, 6.14°	**11.16m, 1.30°**

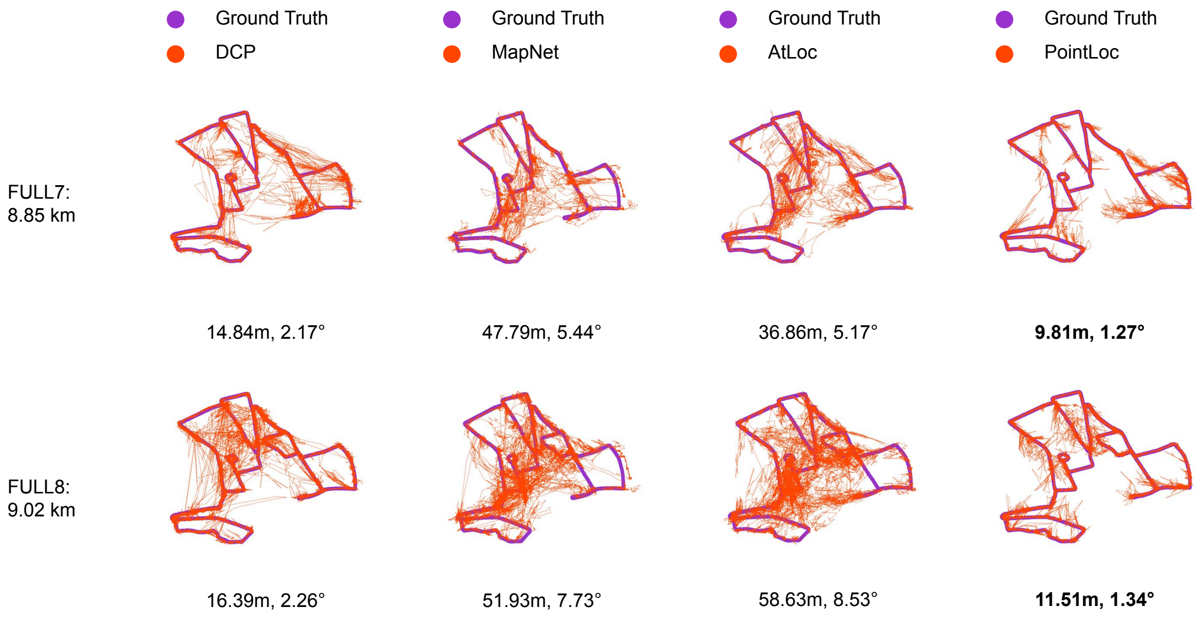


Fig. 4: Trajectories of DCP, MapNet, AtLoc and the proposed PointLoc on FULL7 and FULL8 with mean translation error (m) and rotation error (°). The darkorchid line is the ground truth poses, and the orange-red dot line shows the estimated poses. Our PointLoc outperforms the existing LiDAR and camera relocalization approaches by a significant margin.

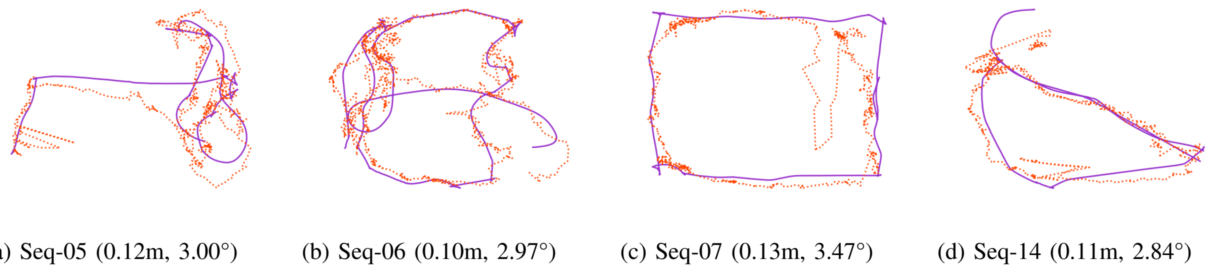


Fig. 5: Trajectories of the proposed PointLoc on the vReLoc test dataset with median translation error (m) and rotation error (°). The darkorchid line is the ground truth poses, and the orangered dot line is the estimated poses.

TABLE VIII: Comparisons of the computational time and storage space of PointLoc and PointNetVLAD.

	Computational Time	Storage Space of the System
PointLoc	0.0985 sec	13 MB
PointNetVLAD	38.0242 sec	283 MB

TABLE IX: Results showing the mean translation error (m) and rotation error (°) of ablation studies on the Oxford Radar RobotCar.

Scene	w/o SA	4096 Points	Pose Regressor	PointLoc
FULL6	15.24m, 1.95°	**13.69m**, 1.95°	49.51m, 9.79°	**13.81m, 1.53** °
FULL7	11.32m, 1.72°	10.33m, 1.40°	41.53m, 8.33°	**9.81m, 1.27°**
FULL8	13.63m, 1.80°	12.35m, 1.34°	42.76m, 9.26°	**11.51m, 1.34°**
FULL9	11.03m, 1.41°	**8.91m, 1.01°**	36.80m, 7.01°	9.51m, 1.07°
Average	12.10m, 1.62°	11.32m, 1.43°	42.65m, 8.60°	**11.16m, 1.30°**

simulates the real-world scenarios of robot movements like service robots inside a large shopping mall. The robot moved forward and backward, halting when it faces obstacles. Data was collected under three conditions: static environment, one-person walking, and two-persons walking. We named the collected dataset vReLoc since it was acquired in a Vicon room for indoor relocalization task. It includes in total 18 robot movement sequences in an indoor Vicon environment.

The median errors and trajectories of test results using our PointLoc are plotted in Fig. 5. The results demonstrated that PointLoc can be successfully applied to the real-world indoor scenarios for LiDAR sensor localization.

E. Ablation Study

To explore the impact of different components of PointLoc, we conduct the ablation studies in Table IX. For ablation experiments, we keep all the architecture designs the same as PointLoc except that we do not contain self-attention module (w/o SA), sample 4096 points from raw point clouds (4096 Points), and utilize two fully-connected layers to predict 6-DoF poses directly (Pose Regressor). We report results on the Oxford Radar RobotCar dataset. Without self-attention module (w/o SA), the performance decreases by 7.77% in translation and 19.75% in rotation. This indicates that the self-attention module indeed enhances the accuracy of relocalization. Meanwhile, the PointLoc improves the same architecture with 4096 points (4096 Points) by 1.4% in translation and 9.09% in rotation, which demonstrates that the more sampled points can improve the performance of localization. Furthermore, the PointLoc improves the architecture with two fully-connected layers of pose regressor (Pose Regressor) by 73.83% in translation and 84.88% in rotation, which reveals the effectiveness of our design of Multi-Layer Perceptrons (MLPs) of two branches.

VII. Conclusion

This paper presents a novel LiDAR sensor relocalization approach, PointLoc, based on deep learning. Leveraging a point-based neural network, it achieves better relocalization accuracy than previous LiDAR and visual sensor-based relocalization approaches. The approach can be applied to large-scale relocalization and robot navigation scenarios for meter-level localization requirements. It can also be leveraged in indoor environments or urban areas full of high-rise buildings as a complement when the GNSS is absent. In the future, more explorations can be done for further improving the relocalization accuracy such as eliminating the noisy point features from the point cloud or exploring intensity information for better relocalization performance.

References

[1] H. Kim, B. Liu, C. Y. Goh, S. Lee, and H. Myung, "Robust vehicle localization using entropy-weighted particle filter-based data fusion of vertical and road intensity information for a large scale urban area," *RA-L*, vol. 2, no. 3, pp. 1518–1524, 2017.

[2] J. Levinson and S. Thrun, "Robust Vehicle Localization in Urban Environments Using Probabilistic Maps," *ICRA*, pp. 4372–4378, 2010.

[3] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song, "L3-Net : Towards Learning based LiDAR Localization for Autonomous Driving," *CVPR*, pp. 6382–6391, 2019.

[4] I. A. Barsan, S. Wang, A. Pokrovsky, and R. Urtasun, "Learning to Localize Using a LiDAR Intensity Map," *CoRL*, pp. 605–616, 2018.

[5] X. Wei, I. A. Barsan, S. Wang, J. Martinez, and R. Urtasun, "Learning to Localize Through Compressed Binary Maps," *CVPR*, pp. 10 308–10 316, 2019.

[6] D. Barnes, M. Gadd, P. Murcutt, P. Newman, and I. Posner, "The Oxford Radar RobotCar Dataset: A Radar Extension to the Oxford RobotCar Dataset," *ICRA*, pp. 6433–6438, 2020.

[7] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *IJRR*, vol. 36, no. 1, pp. 3–15, 2016.

[8] M. A. Uy and G. H. Lee, "PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition," *CVPR*, pp. 4470–4479, 2018.

[9] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, "SegMap: 3D Segment Mapping using Data-Driven Descriptors," *RSS*, 2018.

[10] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization," *ICCV*, pp. 2938–2946, 2015.

[11] A. Kendall and R. Cipolla, "Geometric Loss Functions for Camera Pose Regression with Deep Learning," *CVPR*, pp. 6555–6564, 2017.

[12] R. Clark, S. Wang, A. Markham, N. Trigoni, and H. Wen, "VidLoc: A deep spatio-temporal model for 6-DoF video-clip relocalization," *CVPR*, pp. 2652–2660, 2017.

[13] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz, "Geometry-Aware Learning of Maps for Camera Localization," *CVPR*, pp. 2616–2625, 2018.

[14] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu, "Image-Based Localization Using Hourglass Networks," *ICCV-W*, pp. 870–877, 2017.

[15] F. Walch, C. H. L. Leal-taix, T. Sattler, and S. H. D. Cremers, "Image-based localization using LSTMs for structured feature correlation," *ICCV*, 2017.

[16] Z. Huang, Y. Xu, J. Shi, and X. Zhou, "Prior Guided Dropout for Robust Visual Localization in Dynamic Environments," *ICCV*, pp. 2791–2800, 2019.

[17] B. Wang, C. Chen, C. X. Lu, P. Zhao, N. Trigoni, and A. Markham, "AtLoc: Attention Guided Camera Localization," *AAAI*, vol. 34, no. 06, pp. 10 393–10 401, 2020.

[18] A. Kendall and R. Cipolla, "Modelling uncertainty in deep learning for camera relocalization," *ICRA*, pp. 4762–4769, 2016.

[19] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-taix, "Understanding the Limitations of CNN-based Absolute Camera Pose Regression," *CVPR*, pp. 3297–3307, 2019.

[20] Y. Almalioglu, M. Turan, C. X. Lu, N. Trigoni, and A. Markham, "Milli-RIO: Ego-Motion Estimation with Low-Cost Millimetre-Wave Radar," *IEEE Sensors Journal*, vol. 21, no. 3, pp. 3314–3323, 2020.

[21] H. Rizk, M. Torki, and M. Youssef, "CellinDeep: Robust and Accurate Cellular-Based Indoor Localization via Deep Learning," *IEEE Sensors Journal*, vol. 19, no. 6, pp. 2305–2312, 2019.

[22] D. Alshamaa, F. Mourad-Chehade, and P. Honeine, "Decentralized Kernel-Based Localization in Wireless Sensor Networks Using Belief Functions," *IEEE Sensors Journal*, vol. 19, no. 11, pp. 4149–4159, 2019.

[23] S. P. P. Da Silva, J. S. Almeida, E. F. Ohata, J. J. Rodrigues, V. H. C. De Albuquerque, and P. P. Reboucas Filho, "Monocular Vision Aided Depth Map from RGB Images to Estimate of Localization and Support to Navigation of Mobile Robots," *IEEE Sensors Journal*, vol. 20, no. 20, pp. 12 040–12 048, 2020.

[24] M. T. Hoang, B. Yuen, X. Dong, T. Lu, R. Westendorp, and K. Reddy, "Semi-sequential probabilistic model for indoor localization enhancement," *IEEE Sensors Journal*, vol. 20, no. 11, pp. 6160–6169, 2020.

[25] W. Li, C. Zhang, and Y. Tanaka, "Pseudo Label-Driven Federated Learning-Based Decentralized Indoor Localization via Mobile Crowdsourcing," *IEEE Sensors Journal*, vol. 20, no. 19, pp. 11 556–11 565, 2020.

[26] R. Zhou, H. Hou, Z. Gong, Z. Chen, K. Tang, and B. Zhou, "Adaptive Device-Free Localization in Dynamic Neural Networks," *IEEE Sensors Journal*, vol. 21, no. 1, pp. 548–559, 2021.

[27] W. Wang, M. R. U. Saputra, P. Zhao, P. Gusmao, B. Yang, C. Chen, A. Markham, and N. Trigoni, "DeepPCO : End-to-End Point Cloud Odometry through Deep Parallel Neural Network," *IROS*, pp. 3248–3254, 2019.

[28] C. Li, S. Wang, Y. Zhuang, and F. Yan, "Deep Sensor Fusion between 2D Laser Scanner and IMU for Mobile Robot Localization," *IEEE Sensors Journal*, vol. 21, no. 6, pp. 8501–8509, 2019.

[29] M. Horn, N. Engel, V. Belagiannis, M. Buchholz, and K. Dietmayer, "DeepCLR: Correspondence-less architecture for deep end-to-end point cloud registration," *arXiv*, 2020.

[30] Z. J. Yew and G. H. Lee, "3DFeat-net: Weakly supervised local 3D features for point cloud registration," *ECCV*, pp. 607–623, 2018.

[31] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan, and S. Song, "DeepVCP: An End-to-End Deep Neural Network for Point Cloud Registration," *ICCV*, pp. 12–21, 2019.

[32] Y. Wang and J. Solomon, "Deep Closest Point: Learning Representations for Point Cloud Registration," *ICCV*, pp. 3522–3531, 2019.

[33] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," *NeurIPS*, pp. 5099–5108, 2017.

[34] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," *CVPR*, pp. 77–85, 2017.

[35] T. Le and Y. Duan, "PointGrid: A Deep Network for 3D Shape Understanding," *CVPR*, pp. 9204–9214, 2018.

[36] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep Hough Voting for 3D Object Detection in Point Clouds," *ICCV*, pp. 9276–9285, 2019.

[37] Y. Zhou and O. Tuzel, "VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection," *CVPR*, pp. 4490–4499, 2018.

[38] W. Liu, J. Sun, W. Li, T. Hu, and P. Wang, "Deep learning on point clouds and its application: A survey," *Sensors*, vol. 19, no. 19, 2019.

[39] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D : Learning Scene Flow in 3D Point Clouds," *CVPR*, pp. 529–537, 2019.

[40] T. Zheng, C. Chen, J. Yuan, B. Li, and K. Ren, "PointCloud Saliency Maps," *ICCV*, pp. 1598–1606, 2019.

[41] B. Yang, S. Wang, A. Markham, and N. Trigoni, "Robust Attentional Aggregation of Deep Feature Sets for Multi-view 3D Reconstruction," *IJCV*, pp. 53–73, 2019.

[42] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni, "VINet: Visual-Inertial Odometry as a Sequence-to-Sequence Learning Problem," *AAAI*, vol. 31, no. 1, 2017.

[43] S. Wang, R. Clark, H. Wen, and N. Trigoni, "DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks," *ICRA*, pp. 2043–2050, 2017.

[44] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *ICLR*, 2015.

[45] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic Graph CNN for Learning on Point Clouds," *ACM TOG*, vol. 38, no. 5, pp. 1–12, 2019.

Wei Wang is currently a PhD candidate at Department of Computer Science, University of Oxford. Before that, he obtained his Master of Science Degree from the Carnegie Mellon University and BEng Degree from the North China Electric Power University. His research interests include robotics, machine learning, and deep learning for sensory data.

Bing Wang is currently a PhD student at Department of Computer Science, University of Oxford. Before that, he obtained his BEng Degree at Shenzhen University, China. His research interest lies in camera localization, feature detection, description & matching, and cross-domain representation learning.

Peijun Zhao is currently a PhD candidate at Department of Computer Science, University of Oxford. He has a Bachelor's degree in Computer Science and Technology from Tsinghua University. His research interests include Cyber-Physical Systems and Human-Computer Interactions.

Changhao Chen received his PhD degree in Department of Computer Science, University of Oxford. Before that, he obtained his MEng degree at National University of Defense Technology, China, and BEng Degree at Tongji University, China. His research interest lies in machine learning for signal processing, and intelligent sensor systems, with applications on ubiquitous localization and pedestrian navigation using mobile devices.

Ronald Clark is a research fellow at Imperial College London. He obtained his PhD from the University of Oxford Department of Computer Science. His work lies at the intersection of computer vision and machine learning. His research mainly focuses on allowing machines to interpret and understand the 3D world around them.

Bo Yang is an Assistant Professor in the Department of Computing at The Hong Kong Polytechnic University where he leads the Visual Learning and Reasoning (vLAR) Group, focusing on the fundamental research problems in machine learning, computer vision, and robotics. He completed his D.Phil/Ph.D degree (2016.10-2020.09) in the Department of Computer Science at the University of Oxford. He obtained an M.Phil degree from the University of Hong Kong and a B.Eng degree from Beijing University of Posts and Telecommunications.

Andrew Markham is an Associate Professor and he works on sensing systems, with applications from wildlife tracking to indoor robotics to checking that bridges are safe. He works in the cyber-physical systems group. He designs novel sensors, investigates new algorithms (increasingly deep and reinforcement learning-based), and applies these innovations to solving new problems. Previously he was an EPSRC Postdoctoral Research Fellow, working on the UnderTracker project. He obtained his Ph.D. from the University of Cape Town, South Africa, in 2008, researching the design and implementation of a wildlife tracking system using heterogeneous wireless sensor networks.

Niki Trigoni is a Professor at the Oxford University Department of Computer Science and a fellow of Kellogg College. She obtained her DPhil at the University of Cambridge (2001), became a postdoctoral researcher at Cornell University (2002-2004), and a Lecturer at Birkbeck College (2004-2007). At Oxford, she is currently Director of the EPSRC Centre for Doctoral Training on Autonomous Intelligent Machines and Systems, a program that combines machine learning, robotics, sensor systems and verification/control. She also leads the Cyber-Physical Systems Group, which is focusing on intelligent and autonomous sensor systems with applications in positioning, healthcare, environmental monitoring and smart cities.