LiDAR and Radar Based Localization with Deep Neural Networks



Wei Wang

Christ Church

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Hilary 2022

This thesis is dedicated to my parents, family members and friends
for their love and support.

Acknowledgements

With unforgettable four years at Oxford, I feel so grateful to many people during my DPhil journey.

First and foremost, I would like to greatly thank my supervisors Prof. Niki Trigoni and Prof. Andrew Markham. From their comprehensive guidance and help, I understand how to consider a problem like a researcher, how to formulate a research question from the real world, and how to solve a research problem from different aspects. Their attitudes towards research encourage me to solve fundamental problems, which I believe is beneficial for my whole life.

Second, I would like to greatly thank all my research collaborators. We work together to solve many challenging problems these days, and I learn lots of research experience and skills from them. Their fruitful discussions always inspire me and shape my researches. Furthermore, I also thank my lab members from Cyber Physical System Group.

Third, I would like to greatly thank my college - Christ Church. I really enjoy the delicious food and everyday formal dinner in the hall. I am so proud of my college. Meanwhile, I greatly thank my friends at Oxford. We have so many happy hours together.

Fourth, I would like to greatly thank the NIST for funding my DPhil study. I am very honored to work on the NIST grant “Pervasive, Accurate, and Reliable Location Based Services for Emergency Responders”.

Finally, I would like to greatly thank my parents and family members for their endless love and support.

Abstract

Localization is a fundamental problem in many application scenarios, including robotics, computer vision and autonomous driving. In this thesis, localization is defined as two sub-tasks: odometry and relocalization. Odometry estimates the relative poses of a sensor in real-time when provided with a sequential series of sensory input data, and relocalization predicts the global poses of a sensor from previously traversed scenes. Recently, deep neural networks have been applied to visual localization and shown promising results. However, vision sensors such as cameras are greatly impacted by environmental conditions which limits the application of purely visual methods in the real world. Active sensors, such as LiDAR and radar, can capture rich geometric and semantic information in the forms of point clouds and polar scans and they are significantly more robust to environmental variations. Inspired by the recent deep visual localization methods, this thesis studies how to apply deep learning for the localization task using LiDAR and radar sensors. Specifically, this thesis studies three aspects of LiDAR and radar based localization with deep neural networks. First, an end-to-end parallel neural network termed DeepPCO is proposed for the LiDAR odometry task. It consists of two sub-networks to predict translation and orientation respectively. Second, a novel LiDAR point cloud-based method for deep LiDAR relocalization is introduced, termed PointLoc. It learns features from point clouds directly for estimating global poses in an end-to-end manner. In addition, a new indoor LiDAR-visual dataset termed vReLoc is released for studying the LiDAR relocalization problem. Third, a radar-based relocalization system, RadarLoc, is proposed by utilizing deep learning and FMCW radar scans. It designs a self-attention module with a nested encoder-decoder architecture to mask out outliers. Geometric constraints are leveraged in the loss function to improve the overall performance. Extensive experiments on large-scale datasets have been conducted to demonstrate the effectiveness of the above three approaches. This thesis also discusses implications and future research directions of this field.

Contents

1 Introduction **1**

1.1 Motivation . . . 1
1.2 Problem Formulation . . . 2
1.3 Research Challenges and Objectives . . . 3
1.4 Contributions . . . 5
1.5 Thesis Structure . . . 7

2 Literature Review **8**

2.1 Introduction to Localization System . . . 8
2.2 Odometry Estimation with Deep Neural Networks . . . 9
2.2.1 Visual Odometry . . . 9
2.2.2 LiDAR Odometry . . . 10
2.2.3 Radar Odometry . . . 11
2.3 Relocalization with Deep Neural Networks . . . 14
2.3.1 Visual Relocalization . . . 14
2.3.2 LiDAR Relocalization . . . 15
2.3.3 Radar Relocalization . . . 16
2.4 Summary . . . 17

3 LiDAR Odometry with Deep Neural Networks **19**

3.1 Introduction . . . 19
3.2 Problem Statement . . . 21
3.3 Method Details . . . 21
3.3.1 Approaches of Point Cloud Encoding . . . 21
3.3.1.1 2-D Encoding of a Point Cloud . . . 22
3.3.1.2 3-D Encoding of a Point Cloud . . . 23
3.3.2 DeepPCO Architecture . . . 24
3.3.2.1 End-to-End Network Architecture . . . 24

3.3.2.2 Output and Cost Function . . . 26
3.4 Experiments . . . 26
3.4.1 Dataset . . . 27
3.4.2 Implementation and Settings . . . 27
3.4.3 Testing Results . . . 27
3.4.4 Trajectory Evaluation . . . 27
3.4.5 Comparison to Conventional Approach . . . 28
3.4.6 Ablation Study . . . 29
3.5 Summary . . . 31

4 LiDAR Relocalization with Deep Neural Networks 33
4.1 Introduction . . . 33
4.2 Problem Statement . . . 36
4.3 Method Details . . . 36
4.3.1 Deep Point Cloud Relocalization . . . 36
4.3.1.1 Point Cloud Pre-Processing . . . 36
4.3.1.2 Point Cloud Encoder . . . 37
4.3.2 Self-Attention Module . . . 38
4.3.3 Group All Layers Module . . . 39
4.3.4 Pose Regressor . . . 39
4.3.5 Loss Function . . . 40
4.4 New Dataset . . . 40
4.5 Experiments . . . 41
4.5.1 Implementation Details . . . 41
4.5.2 Baselines . . . 42
4.5.3 Results on the Oxford Radar RobotCar . . . 43
4.5.4 Results on a Real-World Indoor Robot . . . 45
4.5.5 Ablation Study . . . 46
4.6 Summary . . . 47

5 Radar Relocalization with Deep Neural Networks 49
5.1 Introduction . . . 49
5.2 Problem Statement . . . 51
5.3 Method Details . . . 51
5.3.1 Self-Attention for Robust Relocalization . . . 52
5.3.2 Radar Encoder . . . 53
5.3.3 Deep Pose Regressor . . . 54

5.3.4 Loss Function with Geometric Constraints 54
5.4 Experiments 55
5.4.1 Dataset 55
5.4.2 Implementation 56
5.4.3 Baselines 56
5.4.4 Results 56
5.4.5 Ablation Study 58
5.5 Summary 59

6 Conclusions **60**
6.1 Key Contributions 60
6.2 Discussion 62
6.3 Future Work 64
6.4 Conclusions 65

Bibliography **66**

List of Figures

2.1 High-level introduction of a spatial intelligence system. 8
2.2 The typical architecture of deep visual odometry, *e.g.* DeepVO [51]. 9
2.3 The typical architecture of deep LiDAR odometry with projected depth images from LiDAR point clouds [39]. 11
2.4 The difference between LiDAR odometry and LiDAR relocalization [1, 35]. 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. 12
2.5 The typical architecture of deep radar odometry, *e.g.* Masking by Moving [3]. 13
2.6 The difference between radar odometry and radar relocalization [1, 35]. Radar odometry predicts relative poses between consecutive radar scans and thus has accumulative drifts over time, while radar relocalization estimates global poses w.r.t the environment's coordinate system, and assumes that the environment has been traversed before. These are two different tasks in localization, and this work focuses on the radar relocalization. . . . 13
2.7 The typical architecture of deep camera relocalization, *e.g.* Prior Guided Visual Localization [21]. 14
2.8 Hierarchical feature learning architecture of PointNet++ [42]. 15
2.9 The typical architecture of point cloud retrieval-based neural network, *e.g.* PointNetVLAD [48]. 16
2.10 The typical procedure of correspondence matching LiDAR relocalization, *e.g.* L3-Net [34]. 16
2.11 Architecture of Contrastive Radar Relocalization [15]. 17

3.1 System overview of the proposed DeepPCO framework. A sequential point cloud data stream is sent to the system. Two consecutive point cloud data are sent in parallel to data encoding module, which implements the 2-D panoramic projection method. Projected depth images are then stacked to feed to DeepPCO neural network for predicting 6-DOF pose. These relative transformations are further combined to form a moving trajectory. . 20
3.2 A sample point cloud from KITTI dataset. 22
3.3 Visualization of various point cloud encoding approaches for deep learning. 23
3.4 The architecture of DeepPCO. Two consecutive point clouds are encoded into panoramic depth images. The two images are sent to two Sub-Networks simultaneously, both of which are trained using 6-DOF pose ground truths. The Translation Sub-Network is used to predict translation vector (x, y, z) while the FlowNet Orientation Sub-Network infers orientation vector (i, j, k). Translation and orientation vectors at the end of the architecture are concatenated to output the desired 3D transformation. All the initial configurations are listed, and we employ Dropout to prevent overfitting during training. Values shown in convolutional layers above stand for the number of channels or depths. 25
3.5 Trajectories of testing results on Sequence 04 and 10. 28
3.6 Sample projected depth images from sequence 04 in the latter time line. Images are plotted using depth values from range [0, 255]. The surroundings are vacant and lack features. 29
3.7 Ablation experiment of single branch fully connected layers. All the parameter configurations of convolutional layers and fully connected layers are the same as DeepPCO. Different from DeepPCO in which transformation vector is trained using two branches, 3-D translation (x, y, z) and orientation (i, j, k) are jointly trained and inferred by just one branch here. . 31

4.1 PointLoc results for the challenging Oxford Radar RobotCar dataset [1, 35]. We directly feed the LiDAR point cloud 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 relocalization methods significantly. 34

4.2 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 [32] consists of a sampling layer, a grouping layer and a PointNet layer [41]. For more details about the SA layer, please refer to PointNet++ [41] and FlowNet3D [32]. 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. 37
4.3 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. 46
4.4 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. 47

5.1 System overview of the proposed RadarLoc relocalization framework. A raw FMCW radar scan is first transformed into a Cartesian Radar Image. The radar image is then fed to RadarLoc, which directly estimates the 6-DoF pose in an end-to-end manner. 50
5.2 The architecture of RadarLoc. RadarLoc consists of a self-attention module, a radar encoder and a deep pose regressor. A raw FMCW radar scan is transformed into a Cartesian radar image, and then it is fed into a self-attention module to learn a soft attention map. DenseNet [20] is employed as the radar encoder to extract useful features for relocalization. The deep pose regressor predicts the parameterized translation $\mathbf{p} \in R^3$ and rotation $log\mathbf{q} \in R^3$ [6]. 51
5.3 The architecture of self-attention module. The self-attention module is a nested encoder-decoder structure [67]. For better visualization, we only depict 3 levels in the figure, and in our implementation, the nested structure has 6 levels. We adopt a soft attention mechanism, and fuse output features at the upper level to have a fine-grained attention map. 52

5.4 Visual comparisons of all localization approaches for Sequence 10. Poses were projected from 6-DoF to 3-DoF with exception to RadarSLAM, which outputs 3-DoF poses originally. 57
5.5 Cumulative distributions of translation and rotation errors. 57

Chapter 1

Introduction

1.1 Motivation

Localization is one of the most significant problems in many application scenarios, ranging from robotics to Virtual Reality (VR). The localization problem has two key sub-tasks: odometry and relocalization. Existing methods employ various sensor modalities and algorithmic approaches to solve these two tasks. Conventional approaches to localization usually utilize geometric information. Although this is a natural formulation and can solve the problem in some circumstances, they can suffer from lack of robustness as traditional feature extraction methods are not always effective.

In recent years, we have witnessed the rapidly growing trend of data-driven approaches of applying deep learning to challenging tasks, achieving excellent performance in numerous areas, such as classification, semantic segmentation, and object detection. Deep Neural Networks (DNNs) (*e.g.* Convolutional Neural Networks) can effectively and accurately extract features from sensory data such as RGB images and point cloud data for different tasks and applications. Furthermore, the development of big data infrastructures has contributed to the widespread success of deep learning methods for these problems. There are several works that take advantage of deep learning to solve localization tasks recently, showing promising results [6, 11, 23, 24, 51]. Different types of neural network architectures, which were trained on large datasets of RGB images, have been proposed for visual localization. However, RGB images are not always available and easily influenced by outliers like brightness, reflection, and darkness in the environment. In these cases, the camera cannot perceive sufficient information from the scene, which limits the application of purely visual methods in real world scenarios. Active sensors, such as LiDAR and radar, can be excellent alternatives for the above situations. They can capture rich geometric and semantic information in the forms of point clouds and polar scans. They are also robust

to varying environmental conditions, which are encountered widely in many scenarios like autonomous driving, indoor rescue, and robotics tasks.

LiDAR sensors are superior in many ways to cameras, making them suitable for both indoor and outdoor localization. For the emerging requirements of agents to operate under more complex, unstructured environments and in conditions unfavourable for LiDAR sensors *e.g.* indoor firefighters with smoke, LiDAR localization systems may fail [3]. In contrast, radar sensors can penetrate further, through fog, rain, snow, smoke and sunshine [3]. Therefore, radar localization systems can be a supplement or replacement for LiDAR localization systems. Nevertheless, learning-based methods have not yet been well applied to either LiDAR or radar sensory data yet. Therefore, this thesis explores how to employ deep neural networks for LiDAR and radar based localization.

To conclude, inspired by the recent impressive performance of deep learning approaches for visual localization tasks, we study how to apply deep learning approaches for the prediction of 6-DoF poses using LiDAR and radar sensors.

1.2 Problem Formulation

The localization problem can be considered to have two different parts. One is Odometry, which is a key component for Simultaneous Localization and Mapping (SLAM) systems. It refers to the process of incrementally estimating the relative poses of a sensor in real-time when provided with a sequential series of sensory input data. The other is relocalization, which estimates the 6-Degree-of-Freedom (6-DoF) poses of sensors from previously visited scenes. The visited information and sensor data are stored and encoded inside databases, file systems or neural networks. Some of the steps involved in solving the localization problem include feature detection, feature matching and motion estimation. Although conventional approaches can solve the problem to some degree, there is still space for improvement in terms of robustness, efficiency and accuracy. With the advancement of deep learning, neural networks can effectively extract features from images and point clouds, which may lead to more robust and accurate localization performance. Importantly, the ultimate results of both threads in this thesis is to estimate 6-DoF poses. We assume that the installation, deployment, and attachment of the LiDAR and radar sensors are fixed and the type of agents of collecting data is the same for the training, validation and test datasets in the thesis. For odometry, scenes for the training dataset and test dataset can be different. For relocalization, since it requires agents to have traversed the same environments before, the scenes of training dataset and test dataset must be the same, albeit taken at different times.

Formally, the 6-DoF pose estimation can be described by the rigid body transformation matrix, which can be denoted by

$$T_k = \begin{bmatrix} R_k & t_k \\ \mathbf{0} & \mathbf{1} \end{bmatrix}$$

where k denotes the k-th sensor input: either a LiDAR point cloud or a radar scan, R is SO(3), and t is the 3-dimension translation vector. The rotation matrix can be transformed from either Euler angles or quaternion. Therefore, the goal of the localization is to find SE(3).

1.3 Research Challenges and Objectives

Geometric localization methods require significant hand-engineering efforts to tune the huge amount of hyper-parameters, and are sensitive to outliers. Meanwhile, a well known issue of deep learning approaches is that they require large, high-quality, datasets to train neural networks. Acquiring such datasets is challenging due to the significant effort required to obtain calibrated and labelled data. Even with high quality datasets for training, the task of extracting robust features from LiDAR point clouds and radar scans for localization remains challenging. Moreover, when applying deep neural networks to the localization problem, how to develop effective and efficient neural network architectures for pose estimation remains an open problem. We summarize research challenges as follows:

- **Geometric Localization.** On one hand, conventional LiDAR localization systems usually include multiple modules, which need to be carefully designed. Thus, they often fail to generalise successfully to different environments. Furthermore, they require significant human effort to tune hyper-parameters for good localization performance. On the other hand, radar localization is an emerging and promising research area that warrants further research. Recent pioneering work [19] has proposed RadarSLAM, which is based on geometric radar odometry and composed of four subsystems. Therefore, the geometric radar odometry still requires the manual design of modules and careful tuning of hyperparameters.

- **Data Acquisition.** In order to predict 6-DoF poses with deep neural networks, we need to collect enough data. Ideally, the training datasets should cover as many scenarios as possible. For example, if the datasets are collected by an agent such as car or mobile robot, the agent should go straight and turn as much as possible so that collected data are diverse. Specially, we need high-quality point clouds, radar scans and corresponding ground truths for training, validation and testing. Furthermore,

we still require enough data to train neural networks. Public datasets like the Oxford Radar Robotcar [1, 35] are valuable data sources, but they are limited to outdoor settings.

- **Point Cloud Feature Extraction.** Neural networks require input formats to be highly regular. Network architectures for RGB images have been well studied, however, how to effectively extract features from point clouds remains challenging since point clouds are unordered and unstructured. There have been some recent methods for extracting features from point clouds [27, 41, 56], but there remain three open problems. First, these networks were proposed to solve tasks other than localization, *e.g.* point cloud segmentation, object detection and classification. Whether these neural networks can be directly applied to our problem remains a question. Second, among these methods, which architecture is most suitable for our tasks also need to be carefully evaluated. Finally, designing novel neural network architectures that are particularly suited for localization warrants further research.

- **Feature Format and Storage.** For the problem of point cloud odometry, we can directly utilize features during the inference phase. Nevertheless, when we consider the relocalization problem, the previously processed or calculated features need to be effectively stored. There are multiple choices for feature storage. First, we can store them in an in-memory format such as Hash Table or KD-Tree for real-time application. Second, we can save them into a relational database such as MySQL. Third, we can store them into a big data framework such as HDFS for large-scale real world applications. However, the requirement of large storage space prevents the deployment of these methods to the real world. Other issues that arise are how to design the data structure to store the extracted features and original sensory data, and what kind of information we should store. These are some of the key challenges that motivate our research.

- **Outliers.** The indoor and outdoor scenes usually contain pedestrians, cars, buses *etc.*. These moving objects, which are often considered as outliers, significantly affect the performance of localization. However, the data collected from the real world inevitably encompass outliers, especially due to the wide scan range of radar and LiDAR sensor data. Hence, in order to propose accurate and robust solutions, we need to consider the research question of how to reduce the effect of outliers on localization.

- **Real-world Experiments.** Conducting real world experiments is a crucial part for solving localization tasks. Since in our research we leverage a variety of sensors, the first thing that we need to consider is how to capture LiDAR and radar data simultaneously. For comparison with visual localization methods, we also need to collect corresponding RGB images. These sensors need to be carefully calibrated and synchronised.

The overarching challenges described above lead to the fundamental question of this thesis: How can we achieve accurate and robust pose estimation based on LiDAR and radar sensory data with deep neural networks? More specifically, this core question can be divided into three research objectives.

- The first objective is: **How can we best use LiDAR point cloud data to perform accurate odometry estimation?** Particularly, there are several point cloud encoding approaches for neural networks. We need to study which one is best for LiDAR odometry prediction. Meanwhile, existing works on visual odometry employ single neural networks for translation and rotation estimations, which may downgrade the performance of both estimations. Therefore, we also need to study how to predict the best results for each of these targets. This objective is studied in Chapter 3.

- The second objective is: **Without explicitly building a geometric point-cloud map, can we use neural networks to act as a map implicitly for relocalization?** Traditional LiDAR relocalization requires the creation of maps of scenes, which requires considerable human effort and large storage space. Meanwhile, the process is divided into multiple procedures. We aim to find whether we can design an end-to-end relocalization system with high accuracy and low storage requirements. This objective is studied in Chapter 4.

- The third objective is: **Given FMCW radar scans, how can we design a robust and accurate relocalization system?** To answer this question, we need to handle the kinds of outliers typically encountered in FMCW radar scans. We also need to design which neural network architecture is most suitable for radar images. Thus, we attempt to study how to design a robust and accurate radar relocalization system. This objective is studied in Chapter 5.

1.4 Contributions

In this section, the main contributions of each chapter in this thesis are summarized as follows.

- In Chapter 3, we investigate how to exploit deep learning to estimate point cloud odometry (PCO), which may serve as a critical component in point cloud-based downstream tasks or learning-based systems. Odometry is of key importance for localization in the absence of a map. There is considerable work in the area of visual odometry (VO), and recent advances in deep learning have brought novel approaches to VO, which directly learn salient features from raw images. These learning-based approaches have led to more accurate and robust VO systems. However, they have not been applied to point cloud data yet. Specifically, we propose a novel end-to-end deep parallel neural network called DeepPCO, which is able to estimate the 6-DOF poses using consecutive point clouds. It consists of two parallel sub-networks to estimate 3-D position and orientation respectively rather than a single neural network. We validate our approach on KITTI Visual Odometry/SLAM benchmark dataset with different baselines. Experiments demonstrate that the proposed approach achieves good performance in terms of pose accuracy. The proposed DeepPCO significantly outperforms start-of-the-art learning-based LiDAR odometry approaches Two stream [39] and PointNet [41]. The work is published in:

  Wei Wang, Muhamad Risqi U. Saputra, Peijun Zhao, Pedro Gusmão, Bo Yang, Changhao Chen, Andrew Markham, and Niki Trigoni. DeepPCO: End-to-End Point Cloud Odometry through Deep Parallel Neural Network. International Conference on Intelligent Robots and Systems, 2019 [53].

- In Chapter 4, we present a novel end-to-end learning-based LiDAR relocalization framework, termed PointLoc, which infers 6-DoF poses directly using only a single point cloud as input, without requiring a pre-built map. Compared to RGB image-based relocalization, LiDAR frames can provide rich and robust geometric information about a scene. However, LiDAR point clouds 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 input 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. The proposed PointLoc significantly outperforms state-of-the-art point cloud retrieval-based approach PointNetVLAD [48] and deep camera relocalization approach AtLoc [50]. The work is published in:

  Wei Wang, Bing Wang, Peijun Zhao, Changhao Chen, Ronald Clark, Bo Yang, Andrew Markham, and Niki Trigoni. PointLoc: Deep Pose Regressor for LiDAR Point Cloud Localization. IEEE Sensors Journal, 2022 [54].

- In Chapter 5, we investigate how to exploit deep learning to predict global poses from FMCW radar scans. Relocalization is a fundamental task in the field of robotics and computer vision. There is considerable work in the field of deep camera relocalization, which directly estimates poses from raw images. However, learning-based methods have not yet been applied to the radar sensory data. Specifically, we propose a novel end-to-end neural network with self-attention, termed RadarLoc, which is able to estimate 6-DoF global poses directly. We also propose to improve the localization performance by utilizing geometric constraints between radar scans. We validate our approach on the recently released challenging outdoor dataset Oxford Radar RobotCar. The proposed RadarLoc significantly outperforms geometric radar-based localization approach RadarSLAM [19] and deep camera relocalization method AtLoc [50]. The work is published in:

  Wei Wang, Pedro P. B. de Gusmão, Bo Yang, Andrew Markham, and Niki Trigoni. RadarLoc: Learning to Relocalize in FMCW Radar. International Conference on Robotics and Automation, 2021 [52].

1.5 Thesis Structure

The remainder of this thesis is organised as follows:

- Chapter 2 presents a comprehensive review of localization, LiDAR localization, Radar localization, and deep learning for LiDAR and Radar based localization.
- Chapter 3 presents our DeepPCO, an end-to-end point cloud odometry system through deep parallel neural network.
- Chapter 4 presents our PointLoc, a LiDAR relocalization system with deep pose regressor and attention.
- Chapter 5 presents our RadarLoc, a radar relocalization system with the geometric loss function and a nested encoder-decoder style architecture.
- Chapter 6 concludes the entire thesis and identifies directions for future work.

Chapter 2

Literature Review

In this chapter, we provide an overview of prior work on localization techniques with particular emphasis on LiDAR and radar-based systems.

We first illustrate the typical architecture of a localization system in Section 2.1. We then review previous work on two main components: i) odometry estimation, which predicts relative poses in Section 2.2, and ii) global localization which estimate absolute poses in Section 2.3.

2.1 Introduction to Localization System

A localization system is a critical component for mobile agents, such as autonomous vehicles and other robots. A spatial intelligence system is depicted in Fig. 2.1 [9]. The system consists of several components including odometry estimation, mapping and global localization. Odometry estimates the relative pose transformation between two frames of sensor data. Relocalization calculates the global pose of agents with prior scene or environmental knowledge. In this thesis, we focus on the problems of LiDAR odome-

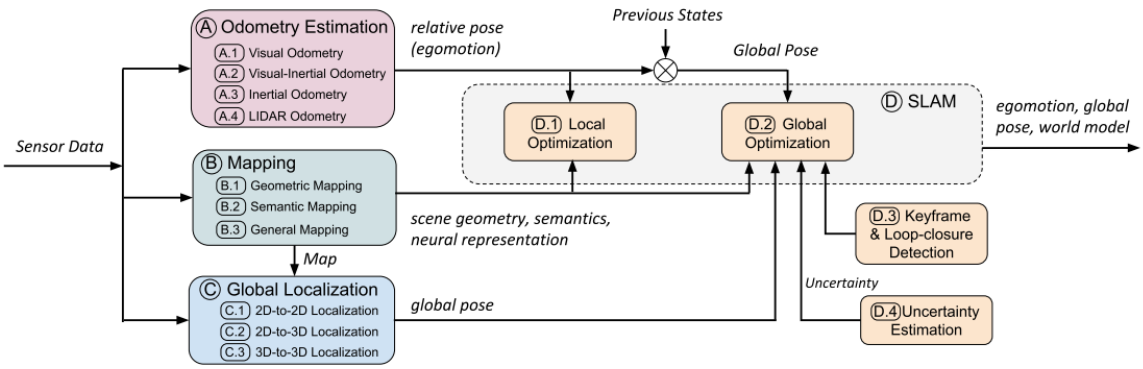


Figure 2.1: High-level introduction of a spatial intelligence system.

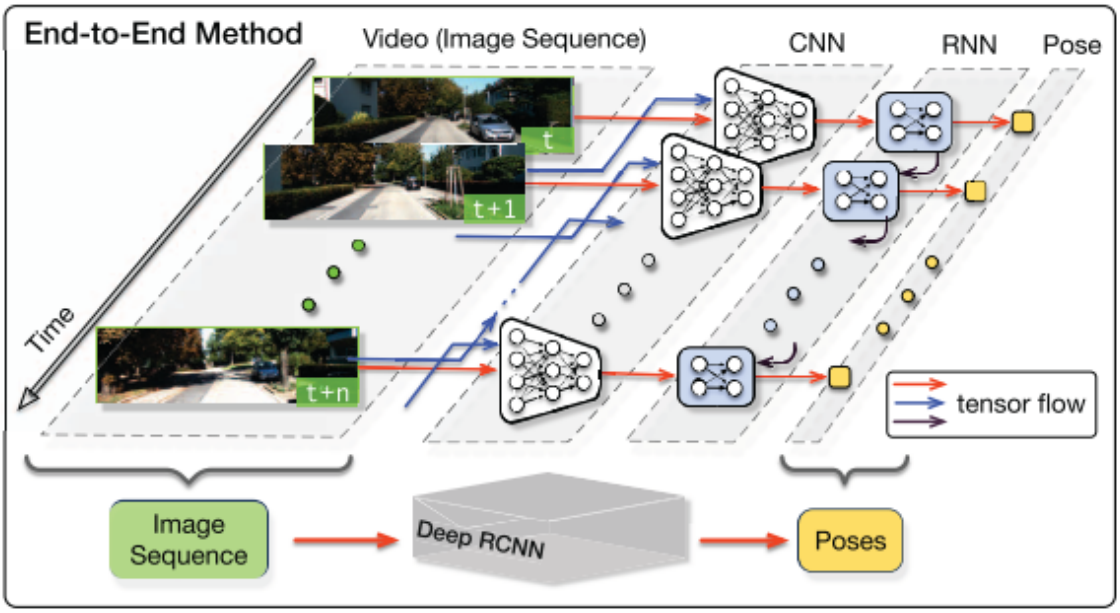


Figure 2.2: The typical architecture of deep visual odometry, *e.g.* DeepVO [51].

try, LiDAR relocalization and Radar Relocalization. The proposed solutions in this thesis are tested with Velodyne LiDAR sensors, and Navtech CTS350-X Frequency-Modulated Continuous-Wave (FMCW) scanning radar.

2.2 Odometry Estimation with Deep Neural Networks

In this section, we review learning-based approaches to odometry estimation. In most situations, relative motion estimations are composed to form a global trajectory. Therefore, it is significant to have accurate odometry estimations to ensure highly accurate localization performance. We discuss odometry estimation with three different types of sensor, namely visual, LiDAR and radar.

2.2.1 Visual Odometry

Instead of hand-engineered features and geometric algorithms to track them through consecutive images, learning-based systems automatically learn the correspondences and relationships between images. These type of systems are usually trained in an end-to-end manner require minimal tuning effort by expert engineers. A typical neural network architecture for deep visual odometry is shown in Fig. 2.2 [51]. Specifically, Wang et al. [51] proposed an end-to-end framework for monocular VO consisting of a CNN branch built upon FlowNet [14], followed by a Recurrent Convolutional Neural Network structure (RCNN).

Features are first extracted for two consecutive images through CNN and then forwarded to an LSTM. Ronald et al. [11] treated visual-inertial odometry as a sequence-to-sequence problem, and made use of an LSTM to embed IMU information to predict pose. Constante et al. [12] presented a novel motion estimation network called LS-VO, which is based on an Auto-Encoder (AE) scheme to find a non-linear representation of the Optical Flow (OF) manifold. [38] used raw optical flow images as inputs and proposed a VO system called Flowdometry, whose architecture consists of CNN. The key advantage of this method is that thanks to careful engineering design, it achieved a 23.796x speedup over existing learning-based VO. Gomez-Ojeda et al. [17] developed a learning-based image enhancement approach to solve the robustness problem in VO. The network architecture consists of a CNN and an LSTM, and the LSTM was proved to help reduce noise by incorporating the temporal information from past sequences. Li et al. [30] presented UnDeepVO to estimate the 6-DOF poses of a monocular camera and the depth of its view. Stereo image pairs were harnessed for recovering absolute scale and the loss function was defined on spatial and temporal dense information. Yang et al. [59] discussed a novel approach called Deep Virtual Stereo Odometry (DVSO), which first predicts depth using monocular images, and then incorporates them into Direct Sparse Odometry (DSO) as direct virtual stereo measurements. All the aforementioned work take 2-D images as inputs of the neural network.

2.2.2 LiDAR Odometry

Classical geometry-based lidar odometry (LO) usually employed variations of Iterative Closest Point (ICP) [5] for scan matching. [46] introduced a probabilistic framework combining ICP and 'point-to-plane' algorithms to model locally planar surface structure from both scans, which can be applied to registration of frame-to-frame scans. The state-of-the-art LO system named LOAM was proposed by [62], which consists of feature point extraction, finding feature point correspondences, motion estimation and then send to mapping components. The key idea is to optimize a large number of variables via two algorithms. One algorithm employs low fidelity to estimate velocity and the other one runs at very low frequency for fine matching and point cloud registration. These two stages can be concluded as fast scan-to-scan and precise scan-to-map. Learning-based LiDAR odometry approaches take either raw point clouds or projected depth images as inputs and estimate relative poses directly. Fig. 2.2 [39] depicts the typical architecture of a deep LiDAR odometry system with projected depth images from LiDAR point clouds. Nicolai et al. [39] proposed a two-stream CNN architecture for frame-to-frame point cloud odometry estimation. They pre-processed point cloud data into 2-D images as inputs for their neural

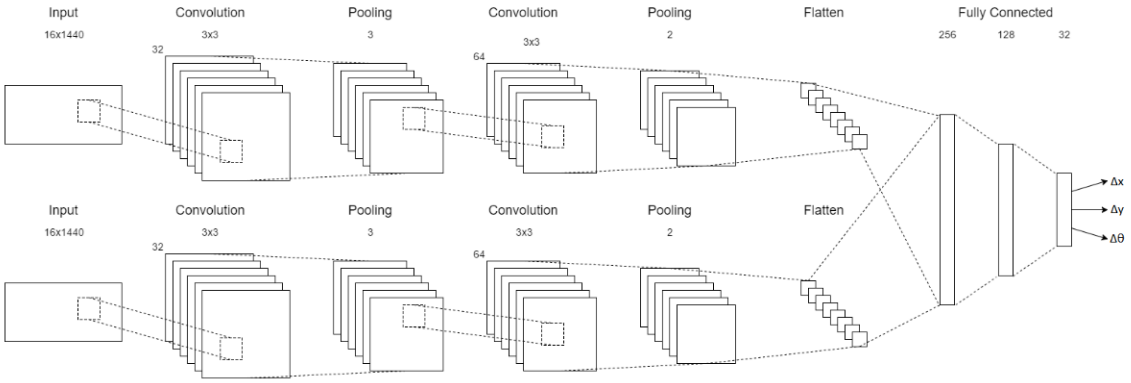


Figure 2.3: The typical architecture of deep LiDAR odometry with projected depth images from LiDAR point clouds [39].

network. They collect point cloud data using VLP-16 LiDAR sensor mounted on Turtlebot. Their experimental results demonstrate that it is possible to apply deep learning to the LO task. Wang *et al.* [53] proposed a deep parallel neural network to directly predict relative poses. 3DFeat-Net [60] was developed to learn both 3-D feature detectors and descriptors for point cloud matching using weak supervision. Lu *et al.* [33] proposed a Virtual Corresponding Points method to align two point clouds accurately. Wang *et al.* [55] 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. 2.4 illustrates the difference between these two localization tasks.

2.2.3 Radar Odometry

We first discuss the radar geometry, and then recent radar odometry works. A 360° FMCW radar continuously scans the surrounding environment with a total of M azimuth angles. The radar emits a beam and collapses the return signal for each azimuth angle [19]. The raw scan of the FMCW radar is a polar image, which can be transformed into a Cartesian image. Formally, given a point (a,b) where a is the azimuth and b is range on a raw polar image, the range angle θ in the corresponding Cartesian coordinate is:

$$\theta = 2\pi \cdot a \,/\, M \tag{2.1}$$

Thus, the corresponding coordinate $\mathbf{Z}$ in the Cartesian image can be calculated as:

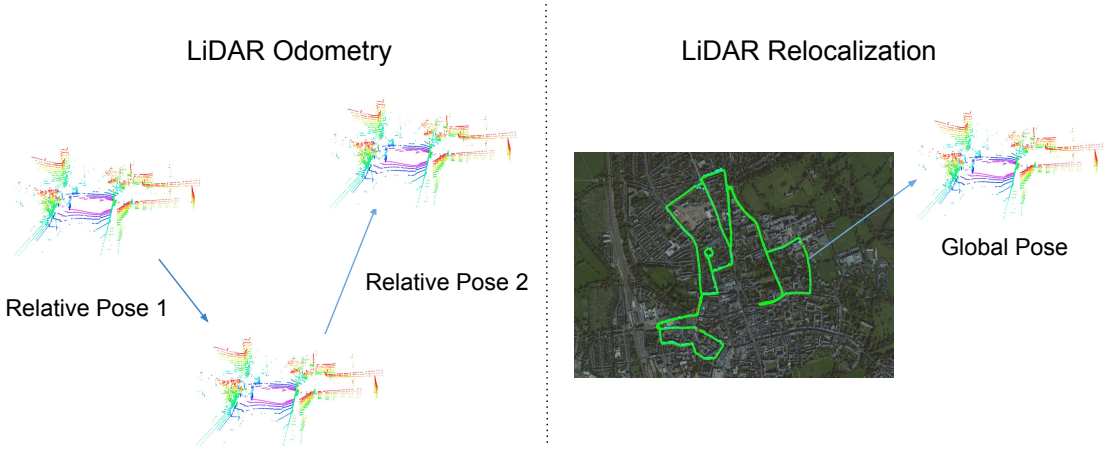


Figure 2.4: The difference between LiDAR odometry and LiDAR relocalization [1, 35]. LiDAR odometry estimates relative poses between consecutive point clouds, which produces accumulative drifts over time, while LiDAR relocalization predicts absolute poses w.r.t the world coordinate, which requires agents previously traverse scenes. These are two different tasks in localization, and this work focuses on the LiDAR relocalization.

$$\mathbf{Z} = \begin{bmatrix} \alpha \cdot cos\theta \cdot b \\ \alpha \cdot sin\theta \cdot b \end{bmatrix} \tag{2.2}$$

where α is a scaling factor between the image pixel space and the world metric space. Cartesian representation of the radar scan is visually comprehensible, and is better for neural networks to learn and optimize than the raw polar representation.

Recent works have been proposed to utilize radar scans for ego-motion estimation, which is known as radar odometry. Fig. 2.5 [3] shows the typical architecture of a deep radar odometry system. Specifically, Cen *et al.* [8] extracted landmarks from radar scans and then conducted scan matching to predict ego-motion based on unary descriptors and pairwise compatibility scores. Barnes *et al.* [3] developed a robust and real-time radar odometry system based on deep correlative scan matching with learnt feature embedding and self-supervised distraction-free module. Afterwards, they proposed a deep key point detection approach for radar odometry estimation and metric localization by embedding a differentiable point-based motion estimator [2]. Note that different from these methods, our work focuses on radar-based absolute localization, which predicts global poses w.r.t. the world coordinate rather than relative poses. Fig. 2.6 illustrates the differences between these two different localization tasks.

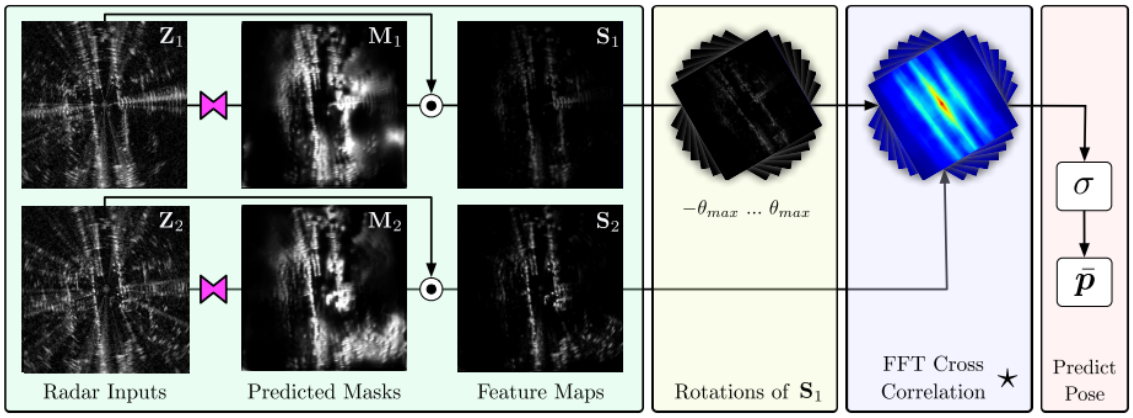


Figure 2.5: The typical architecture of deep radar odometry, *e.g.* Masking by Moving [3].

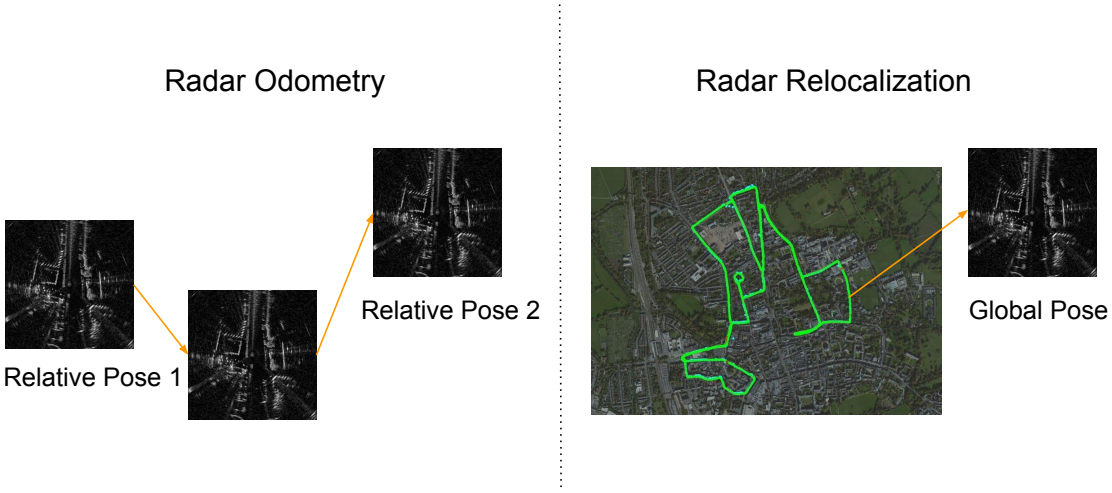


Figure 2.6: The difference between radar odometry and radar relocalization [1, 35]. Radar odometry predicts relative poses between consecutive radar scans and thus has accumulative drifts over time, while radar relocalization estimates global poses w.r.t the environment's coordinate system, and assumes that the environment has been traversed before. These are two different tasks in localization, and this work focuses on the radar relocalization.

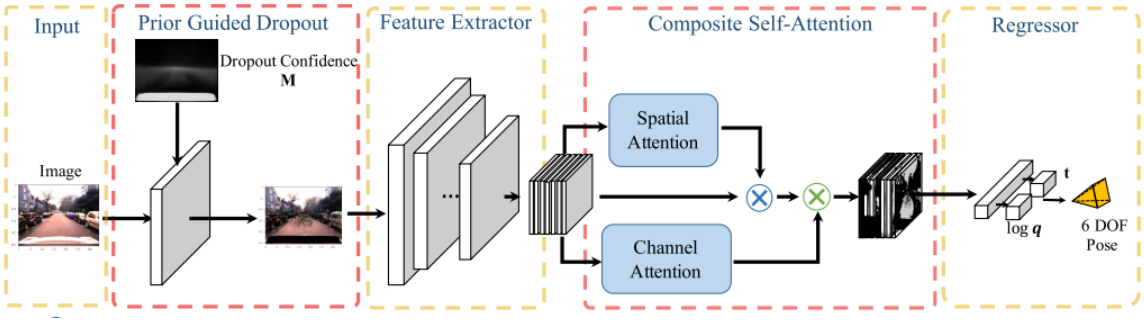


Figure 2.7: The typical architecture of deep camera relocalization, *e.g.* Prior Guided Visual Localization [21].

2.3 Relocalization with Deep Neural Networks

In this section, we review learning-based approaches to relocalization, which predicts the global poses with respect to the environment's coordinate system. Different from odometry estimation, relocalization requires that the same environment has been traversed previously. In other words, environmental priors should be provided and exploited to perform global pose estimations.

2.3.1 Visual Relocalization

For dealing with the drawbacks of map registration methods, recent works propose learning-based approaches to estimate the global pose directly with camera images [6, 10, 21, 23, 24, 37, 49]. They take images, either single or sequential, as inputs to train a neural network model for predicting absolute poses. Fig. 2.7 [21] illustrates the typical architecture of deep camera relocalization, which predicts global poses in an end-to-end manner. The key to these methods is to learn a deep pose regressor, which usually comprises a feature extractor and a regressor [21, 24, 50]. For example, PoseNet related works [22–24] proved the feasibility of predicting the global pose using a single RGB image by regressing the pose directly. Brahmbhatt *et al.* [6] utilized the relative pose between two images as a geometric constraint to estimate the pose. Huang *et al.* [21] and Wang *et al.* [50] have demonstrated that self-attention modules can significantly improve the accuracy of relocalization and guide the network to ignore distracting information from foreground objects in a dynamic environment. 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 [45], 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 and Radar sensors.

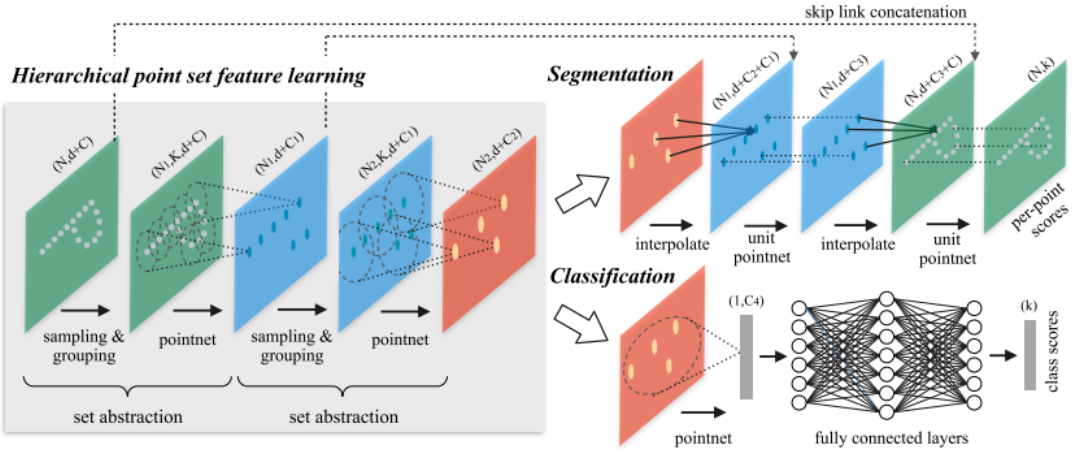


Figure 2.8: Hierarchical feature learning architecture of PointNet++ [42].

2.3.2 LiDAR Relocalization

LiDAR relocalization employs deep learning to extract features from 3-D point clouds for localization, which requires to design effective deep neural networks. We first introduce neural networks for point clouds, and then introduce learning-based LiDAR relocalization.

DNN-based feature extraction methods for point clouds [27, 40–42] have gained significant attention in recent years. VoxelNet [66] was developed to learn feature embeddings in voxels for object detection. PointNet++ related works [40–42] 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. The illustration of PointNet++ is shown in Fig. 2.8 [42]. In this thesis, we follow the design ideas to develop a point-based neural network to solve the LiDAR relocalization problem. Detailed introductions and applications of deep learning for point clouds can be found in the recent survey paper [31].

LiDAR relocalization concerns the prediction of global poses within a given previously visited scene with LiDAR sensors. There are two existing learning-based approaches for LiDAR relocalization. If the scene is processed and represented as a reference database, this can be concluded as point cloud retrieval approach [48, 64]. Global descriptors are extracted from given point clouds and stored into a point cloud database together with ground truth poses or place information during training phase. Fig. 2.9 [48] reveals the typical feature extraction neural network for point cloud retrieval approaches. During inference, the query point cloud is fed into a neural network that generates the feature vector, which is then compared with feature vectors from the reference database. If the scene is represented as a pre-built map, this can be named as correspondence matching [4, 34, 57, 61]. The global

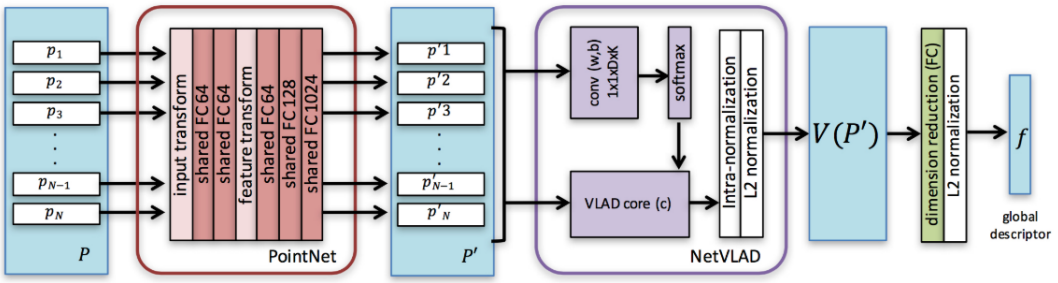


Figure 2.9: The typical architecture of point cloud retrieval-based neural network, *e.g.* PointNetVLAD [48].

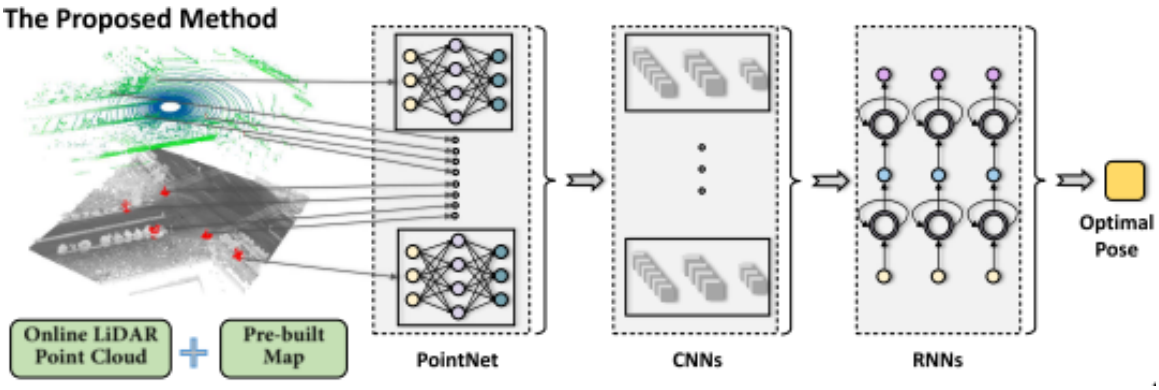


Figure 2.10: The typical procedure of correspondence matching LiDAR relocalization, *e.g.* L3-Net [34].

pose of a point cloud is achieved by establishing the correspondence between the current point cloud and the pre-built map. Fig. 2.10 [34] describes the typical procedure of correspondence matching approaches, wherein the neural network takes online LiDAR scans, a pre-built map and a coarse pose as inputs and estimates the optimal pose. Wei *et al.* [57] proposed to reduce storage requirements by learning to compress the map representation. LocNet [61] developed a semi-handcrafted neural network to learn the representation of LiDAR scans and a Monte-Carlo localization method for global metric localization. Different from these methods, this thesis proposes an end-to-end LiDAR relocalization method to regress global poses directly.

2.3.3 Radar Relocalization

Compared with visual relocalization and LiDAR relocalization, radar relocalization is relatively underexplored. Saftescu *et al.* [43] introduced a topological radar localizaiton system using rotationally-invariant metric learning for place recognition with the FMCW radar. Gadd *et al.* [15] proposed an unsupervised constrastive learning approach for radar relocal-

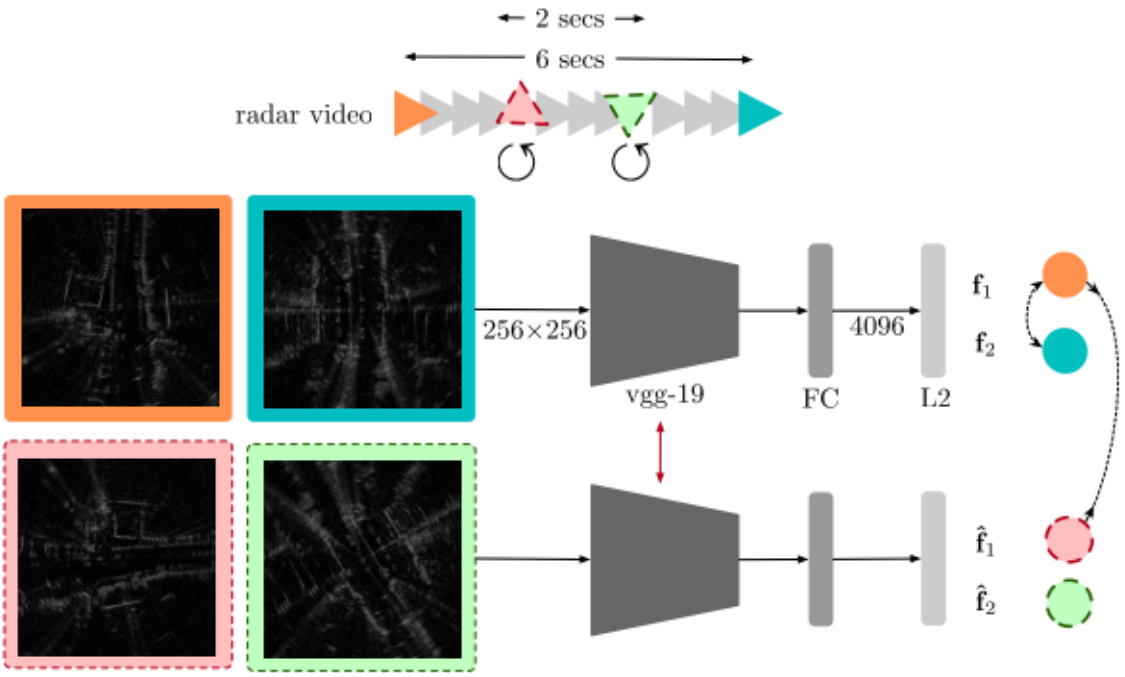


Figure 2.11: Architecture of Contrastive Radar Relocalization [15].

ization by utilizing the temporal successitivity of data collected by a mobile platform. The key is to learn a robust radar descriptor to differentiate from other descriptors so that the feature matching can be accurate enough to retrieve the desired global pose. Fig. 2.11 [15] illustrates the neural network of constrastive radar relocalization. Unfortunately, such feature correspondence approach inherently suffers from the retrieval-based approaches considering the complexity of time and storage, which calls for more effective relocalization methods. RSL-Net [47] was proposed to localize a vehicle in an overhead image using FMCW radar. However, it requires the overhead imagery to perform the localization task.

2.4 Summary

From the literature reviews above, existing LiDAR and radar based localization systems mainly rely on geometric, map-based and retrieval-based methods. Although they can achieve excellent performance in some circumstances, they suffer from several drawbacks. For geometry-based approaches, they require the design of multiple sub-systems and tune numerous hyper-parameters, and can be impacted by running environments. For map-based methods, it is time-consuming to build high-fidelity maps, and the costs of developing such maps are high, especially for large-scale cities. Meanwhile, these systems usually require

accurate initial poses as coarse localization first. For retrieval-based approaches, they need to create reference databases before performing pose estimations. The storage space and computing time will increase linearly when the scale of environments enlarge, which cannot satisfy the real-time localization requirements and be deployed on the edge systems such as service robots. End-to-end deep pose estimation methods inspire a novel direction to develop LiDAR and radar based localization systems, which can achieve impressive localization performance with relatively low time and space complexity. These learning-based localization methods motivate works presented in the following Chapter 3, 4 and 5, leading to LiDAR and radar based localization systems with deep neural networks.

Chapter 3

LiDAR Odometry with Deep Neural Networks

3.1 Introduction

Visual odometry (VO) is a fundamental research topic in the field of computer vision and robotics. The problem it addresses is how to estimate an agent's relative pose by examining changes in projected geometrical features captured from a monocular camera. Conventional visual odometry techniques include feature extraction and matching steps, which have been extensively studied for a number of decades. However, recent advances in deep learning brings another paradigm to VO by directly inferring the 6 Degree-of-Freedom (6-DoF) camera poses through end-to-end learning. This leads to a number of advantages in terms of not requiring hand-crafted features and being able to learn directly from large datasets. Deep learning-based VO approaches achieve very impressive results compared to conventional geometry-based approaches [44, 51].

Point cloud data, such as that generated by a LiDAR or Depth Camera, can provide richer information about the 3-D structure of the environment. This intuitively provides a better perspective on how the sensor moves over time. Conventional point-cloud based odometry relies on geometric approaches and has shown excellent performance [62]. However, these techniques suffer from issues of robustness such as being sensitive to outliers and having a low adaptability to different environments [7]. Scan matching, which is a fundamental task in point-cloud based odometry, is prone to outliers introduced by hardware failure, agent vibration when collecting data, or unexpected sensor movements. Moreover, when the environment is featureless or objects within environment are deformable, scan matching may also fail. These systems usually need other sensors such as an Inertial Measurement Unit (IMU) to compensate for these failures [63]. Another complexity is that operators need to manually fine-tune a large number of model parameters.

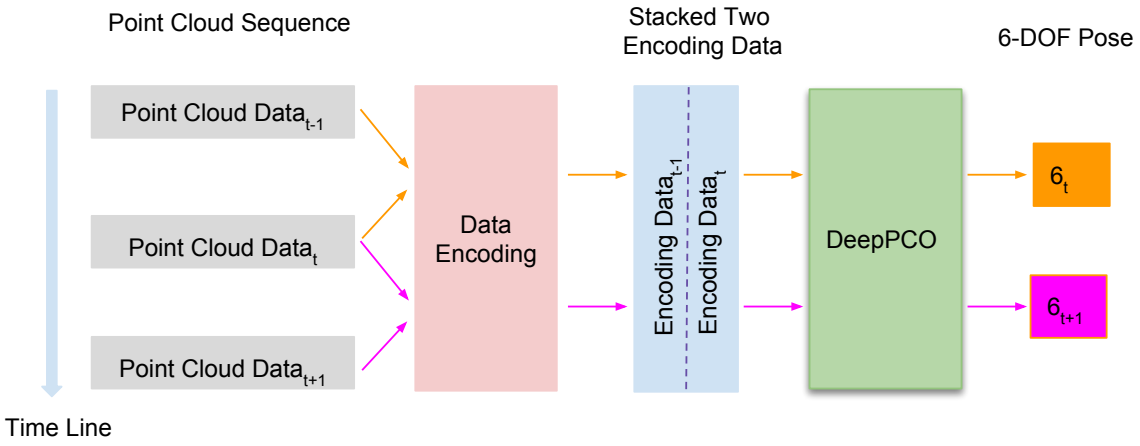


Figure 3.1: System overview of the proposed DeepPCO framework. A sequential point cloud data stream is sent to the system. Two consecutive point cloud data are sent in parallel to data encoding module, which implements the 2-D panoramic projection method. Projected depth images are then stacked to feed to DeepPCO neural network for predicting 6-DOF pose. These relative transformations are further combined to form a moving trajectory.

Motivated by the success of deep learning as applied to VO, we consider whether a similar approach is applicable for estimating odometry from point cloud data, obviating the need for manually engineering features and model parameters, which we define as Point Cloud Odometry (PCO) task. In particular, deep learning may eliminate the problem of noisy scan matching, potentially improving robustness. Nevertheless, a challenging problem in the case of point clouds is that the points are unordered, which raises issues for convolutional kernels which inherently assume a structured input.

In this thesis, we investigate in depth two key issues which impact the performance of PCO using end-to-end deep learning. The first is the representation (encoding) of the point cloud itself. This ranges from the raw point cloud to various approaches to projecting it to a 2-D scene. The second issue comes from the choice of the model architecture itself. In particular, we consider estimating the 6-DOF pose (translation and rotation) in a single regression network, or splitting the translation and rotation estimation tasks and regressing them in two sub-networks.

Based on these investigations, we propose a novel approach, termed DeepPCO, which combines the use of 2-D panoramic depth projections with two sub-networks to achieve accurate odometry, as shown in Fig.3.1. DeepPCO eliminates the intermediate modules (*e.g.* scan matching, geometric estimation) of a classical pipeline. We compare against various baselines using point cloud data from the KITTI Vision Benchmark Suite [16] which were collected using a 360° Velodyne laser scanner.

Our main contributions are as follows:

- We demonstrate that the point cloud odometry problem can be effectively solved in an end-to-end fashion, and our proposed architecture outperforms existing learning-based approaches by a significant margin, and achieves comparable performance to conventional ones.
- We investigate a variety of point cloud encoding approaches which can be applied to the point cloud odometry task using deep learning.
- We adopt a dual sub-network architecture to infer translation and orientation separately instead of a single network.
- Comprehensive experiments and ablation studies have been done to evaluate our proposed method. Results show that DeepPCO outperforms existing learning-based LiDAR odometry methods.

3.2 Problem Statement

The aim of this work is to design a DNN-based framework for performing deep relative pose regression using point clouds from a LiDAR sensor. For each timestamp t, the agent receives two consecutive point cloud frames $\mathbf{P}_{t-1} = \{\mathbf{x}_{t-1}^i \mid i = 1,...,N_{t-1}\}$ of the previous point cloud and $\mathbf{P}_t = \{\mathbf{x}_t^i \mid i = 1,...,N_t\}$ of the current point cloud from LiDAR, where each point $\mathbf{x}_i$ is a vector (x,y,z) describing the 3D coordinates of the point. Therefore, the shape for each $\mathbf{P}_t$ is $(N_t,3)$. The odometry of the agent is parameterized by a 6-DoF pose $[\mathbf{p},\mathbf{q}]^T$, where $\mathbf{p} \in R^3$ is a 3-D translation vector and $\mathbf{q} \in R^3$ is a 3-D rotation vector (Euler angle). To this end, deep 3-D pose regressors learn a function $\mathscr{F}$ such that $\mathscr{F}(\mathbf{P}_{t-1},\mathbf{P}_t) = (\mathbf{p},\mathbf{q})^T$, where the function $\mathscr{F}$ is usually a neural network for DNN-based methods.

3.3 Method Details

3.3.1 Approaches of Point Cloud Encoding

Neural networks usually require highly structured data as inputs whereas point cloud data is unordered and irregular. It is therefore interesting to explore how to transform point cloud data to feed into a neural network. In the following section, we will discuss different encoding approaches of point cloud data for the odometry task. A sample point cloud data is shown in Fig 3.2. The middle blank part represents the car which is mounted with a sensor to collect point cloud data.

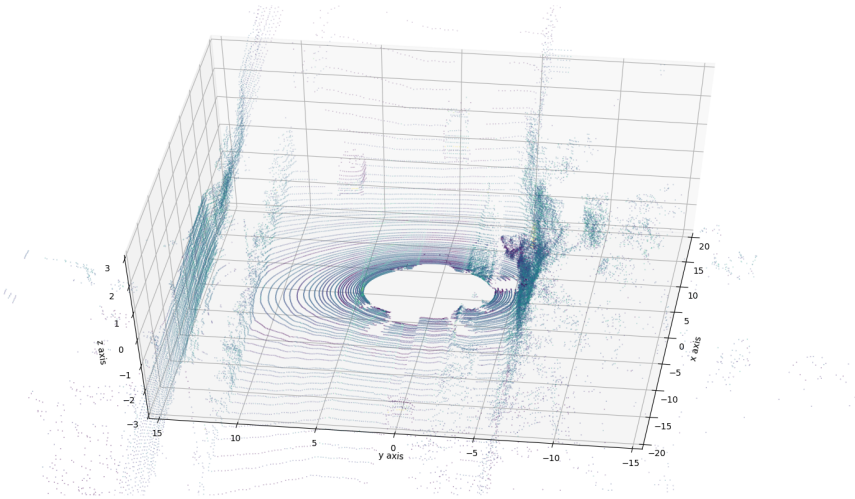


Figure 3.2: A sample point cloud from KITTI dataset.

3.3.1.1 2-D Encoding of a Point Cloud

A straightforward approach to encode a point cloud for the odometry task is to transform it to a 2-D depth image. The projected depth image can be top-view, front-view, or panoramic-view. Among all of our experiments, we find that panoramic-view projection performs the best among all developed models with different parameter settings. Therefore, here we discuss panoramic-view projection equations which we have adopted in our work. Specifically, Li et al. [29] proposed an approach to project a point cloud into a panoramic depth image by the following equations:

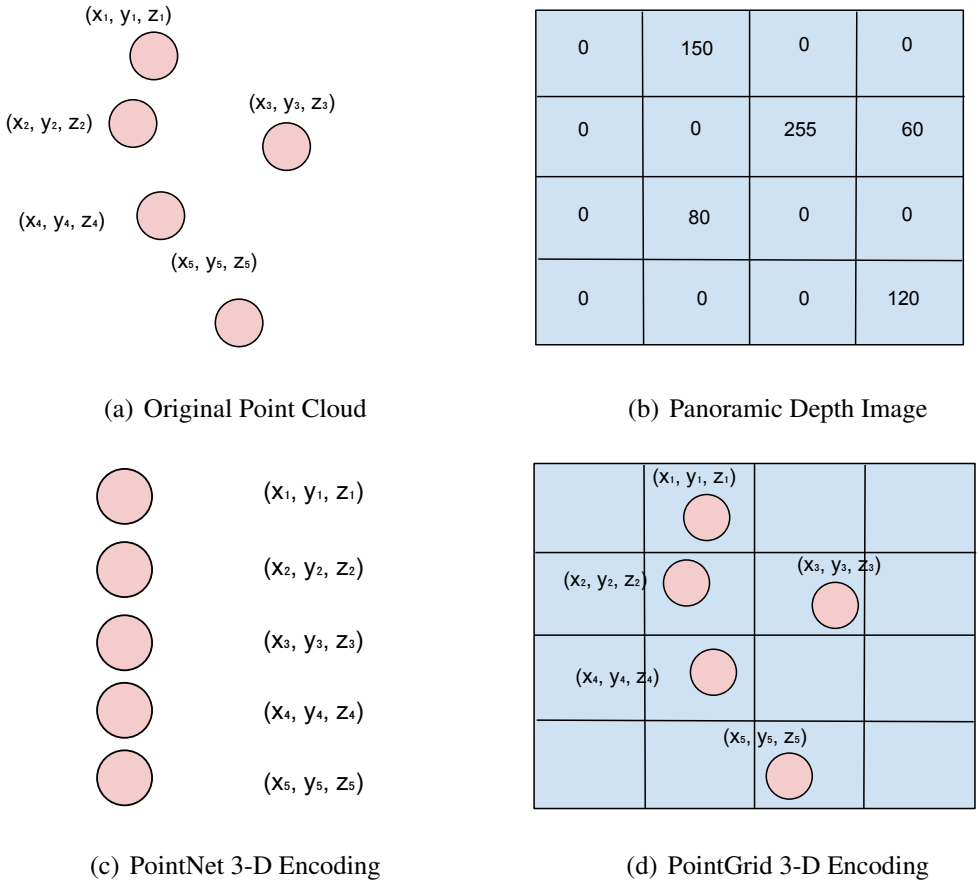
$$\theta = \arctan 2(y, x) \tag{3.1}$$

$$\phi = \arcsin(z/\sqrt{x^2 + y^2 + z^2}) \tag{3.2}$$

$$r = \lfloor \theta / \Delta\theta \rfloor \tag{3.3}$$

$$c = \lfloor \phi / \Delta\phi \rfloor \tag{3.4}$$

where (x,y,z) is 3-D point coordinates vector, θ is the azimuth angle, ϕ is the elevation angle, (r, c) is the 2-D image position of the 3-D point projection, $\Delta\theta$ is the average horizontal angle resolution and $\Delta\phi$ is the vertical angle resolution. For detailed explanation, we refer the reader to the original paper. In our work, we use exactly the same equations above. We then normalize depth values to the range [0, 255]. Points closer to the sensor are assigned higher values. Indeed, such inverse normalization has been proved effective in many 2-D vision tasks.



(a) Original Point Cloud

(b) Panoramic Depth Image

(c) PointNet 3-D Encoding

(d) PointGrid 3-D Encoding

Figure 3.3: Visualization of various point cloud encoding approaches for deep learning.

3.3.1.2 3-D Encoding of a Point Cloud

An alternative representation for 3-D point cloud data is to be discretized as a 3-D voxel grid, an approach that has been employed in VoxNet [36]. Nevertheless, voxelization of point clouds is practically unfeasible. For example, each point cloud data in KITTI [16] contains around 120,000 points, which would require huge memory and computation resources to maintain in a high resolution 3-D grid. Even adopting a sampling strategy to reduce the number of points, voxelization of point cloud data is still difficult to be trained for volumetric CNN. Meanwhile, the sampling can also affect the accuracy of our odometry task. As an alternative to the voxelization approach, Qi *et al.* [41] proposed PointNet, a straightforward approach that treats every single point as a 3 dimensional vector (x, y, z) for 2-D CNN architecture. Thus, the input points are encoded as $n \times 3$, and this approach removes the need for a 3-D CNN architecture which greatly speeds up the training process. Considering the invariance to permutations of points, they developed a spatial transformer network (T-Net) to predict an affine transformation matrix and then apply it to the coordi-

nates of input points. However, they focus mainly on classification and segmentation tasks. The introduced transformation sub-network makes it challenging to predict translation in our PCO task. Therefore, while we adopt most parts of the PointNet architecture, we remove the T-Net. More recently, Le *et al.* [27] proposed a 3-D CNN architecture named PointGrid to combine the benefits of points and grid for better representation of the local geometry shape given the limits of PointNet's ability to capture contextual neighborhood structure. Specifically, they employed *Point Quantization* to a fixed number of K points in each cell. If there are more than K points located inside a grid cell, they sample K points randomly. If there are less than K points within a grid cell, they sample K points with replacement. If there are no points existing, they fill the grid cell with zeros.

The detailed explanation of all point cloud encoding approaches for deep learning in this work have been summarized in Fig.3.3. We used 2-D encoding for our proposed optical neural network architecture, termed DeepPCO, in which the point cloud data is projected to panoramic depth images. In the experiment, we compare it with the 3-D encoding approaches described above.

3.3.2 DeepPCO Architecture

Our DeepPCO is composed of two sub-networks, Translation Sub-Network and FlowNet Orientation Sub-Network, both of which forms a deep parallel neural network architecture. The whole DeepPCO architecture is illustrated in Fig. 3.4.

3.3.2.1 End-to-End Network Architecture

Our insight comes from the idea of *Divide and Conquer* that instead of jointly learning and predicting position and rotation vectors using a single neural network, we design two separate neural networks for inferring each transformation. One is adept at predicting translation while the other specializes in inferring rotation. The inputs of DeepPCO are two consecutive 2-D panoramic-view projection images which are stacked together. The Translation Sub-Network is in charge of prediction of translation between two consecutive point clouds, while the FlowNet Sub-Network infers the rotation between them. We used FlowNet as it has been proved to effectively extract geometrical features useful for the odometry task. In the Translation Sub-Network, fully convolutional layers are utilized to extract features from projected depth images, while in the FlowNet Sub-Network, we adopt the same configuration of convolutional layers from FlowNet [14]. Our intuition is that is is beneficial to take advantage of FlowNet as optical flow is heavily used in VO tasks to attain good rotation performance. Since the input channel of FlowNet is 6, we expand each depth

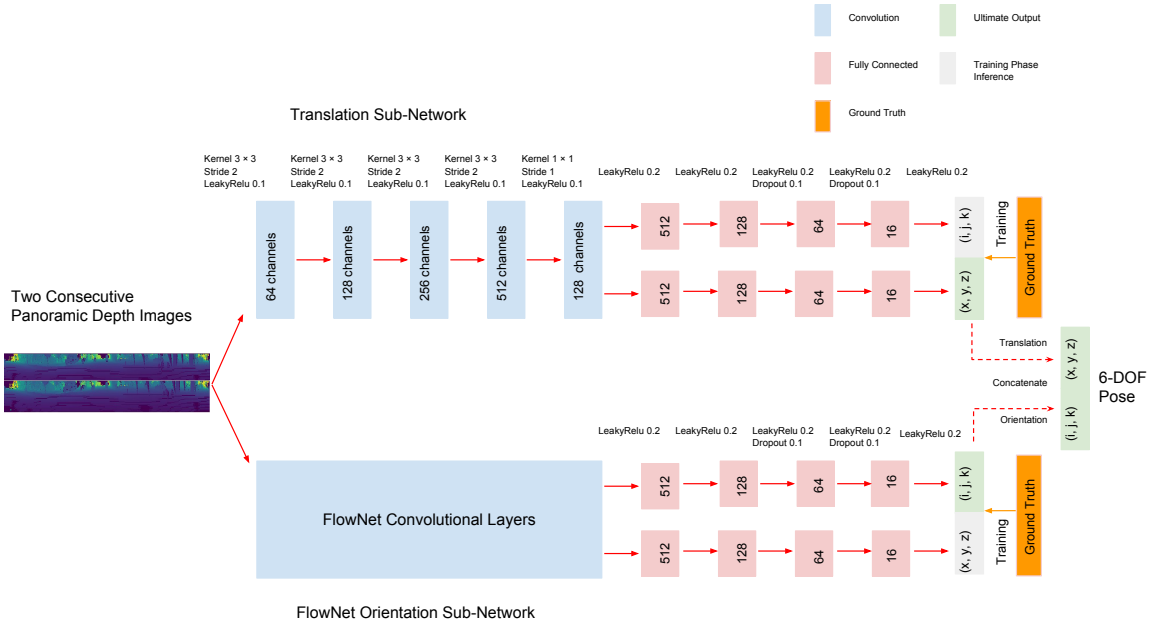


Figure 3.4: The architecture of DeepPCO. Two consecutive point clouds are encoded into panoramic depth images. The two images are sent to two Sub-Networks simultaneously, both of which are trained using 6-DOF pose ground truths. The Translation Sub-Network is used to predict translation vector (x, y, z) while the FlowNet Orientation Sub-Network infers orientation vector (i, j, k). Translation and orientation vectors at the end of the architecture are concatenated to output the desired 3D transformation. All the initial configurations are listed, and we employ Dropout to prevent overfitting during training. Values shown in convolutional layers above stand for the number of channels or depths.

image to 3 channels by replicating it 2 more times directly. Leaky Rectified Linear Units (Leaky ReLU) is applied after every convolutional layer in our architecture with the angle of negative slope set to 0.1 and in-place operation. We employ transfer learning to initialize the weights of convolutinal layers in our FlowNet Sub-Network using pretrained FlowNet convolutional layers' weights. We note that we do not use any max pooling layers or batch normalization layers in our architecture since our comprehensive experiments indicate that adding such layers significantly decreases the prediction accuracy. The reason may be that these feature processing operations lose some significant geometric information which is required for precise odometry estimation. Moreover, both of these two sub-networks have the same fully connected layers settings as in the [39], which consist of two branches jointly trained by 6-DOF ground truth. Although their work used their own collected indoor point cloud dataset to design the network, our experiments empirically prove that such a setting of fully connected layers offer good performance for the PCO task. Another important design is that we jointly learn pose for the two sub-networks respectively. PoseNet [24]

demonstrated that regressing position and orientation separately performed poorly compared to full 6-DOF pose training. Our experiment further proves that the joint training of translation and rotation performs better than training translation or rotation separately. Hence, our neural network architecture is end-to-end and easy to be trained since it only employs few 2-D CNN layers and very small fully connected layers eliminating the use of a large RCNN architecture.

3.3.2.2 Output and Cost Function

Our network outputs a 6-DOF pose vector **v**:

$$\mathbf{v} = \mathbf{[p,\ q]} \tag{3.5}$$

where **p** is a 3-D position and **q** is an Euler angle. We express orientation of pose as Euler angle rather than quaternion based on the reason that quaternion is subject to an extra unit constraint affecting the optimization of our network. Since two sub-networks are trained in a supervised manner simultaneously, our final objective loss function L across two sub-networks is defined as follows:

$$L = \|\mathbf{p} - \hat{\mathbf{p}}\|_2^2 + k * \|\mathbf{q} - \hat{\mathbf{q}}\|_2^2 \tag{3.6}$$

where $\hat{\mathbf{p}}$ is ground truth of translation vector, $\hat{\mathbf{q}}$ is ground truth of orientation vector, $\| * \|_2^2$ measures the mean squared error (squared L2 norm) between each element in the estimation vector and ground truth vector, and k is a scale factor to balance the errors of position and orientation to be approximately equal.

3.4 Experiments

In this section, we introduce our experimental set up and provide details on training. We compare our network to 5 baseline approaches, which are designed and tested on their best variations for our task. Austin's two-stream network performs the same task as ours and is set as the benchmark for PCO task. DeepVO and ResNet18 [18] are chosen for 2-D encoding of point clouds since they perform well in visual odometry tasks. We select PointNet and PointGrid as comparisons for 3-D encoding of point cloud given their excellent performance in 3-D point cloud tasks like semantic segmentation.

3.4.1 Dataset

The KITTI VO/SLAM benchmark dataset is chosen for the experiment. The dataset consists of 11 sequences (Sequences 00-10), which are associated with ground truths. Each sequence contains consecutive point cloud data collected by laser sensor for trajectory estimation. Since the dataset is relatively small compared to other benchmark datasets, we selected Sequence 00-03, 05-09 as training datasets, and Sequence 04 and 10 as testing datasets which are chosen by most VO work since they can represent the major scenarios for agent or sensor movements in real world. Considering our loss function adopts Euler angle while the orientation of ground truth in KITTI is quaternion, we transformed orientation inside our experimental datasets to Euler angle. All poses in KITTI are based on absolute transformation, so we converted them to relative transformation for the purpose of training the network.

3.4.2 Implementation and Settings

Our architecture is implemented with PyTorch framework. For evaluating our results, we choose Root Mean Square Error (RMSE) as our metric. Adam [26] optimizer was employed to train our neural network with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We set learning rate from 0.0001 and then we applied 50% weight decay for every 10 epochs. Our network was trained for 30 epochs according to different encoding approaches and architectures. The batch size was set to 8. The scale factor k of the objective function is 100. We use 8-fold cross-validation for choosing hyperparameters. During the training phase, we also used data shuffling to prevent overfitting.

3.4.3 Testing Results

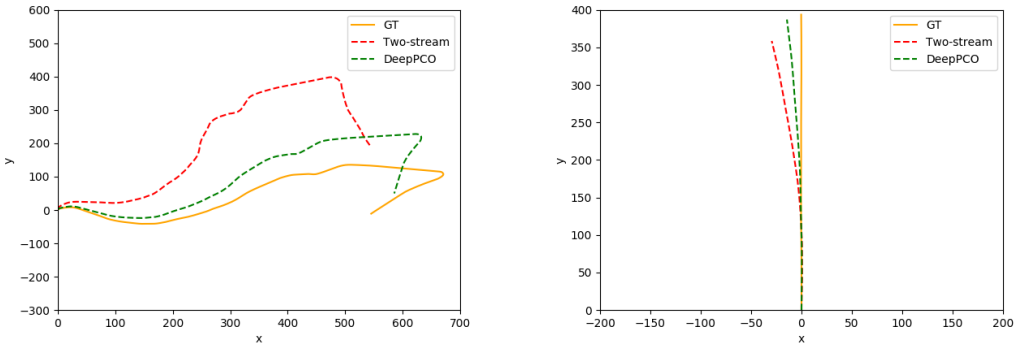
The test results are presented in Table 3.1. Our experiment indicates that the proposed architecture outperforms all the baselines in both test sequences. A deeper convolutional neural network architecture, like ResNet, is not necessary for achieving excellent odometry results. Among 3-D encoding approaches, PointGrid is more suitable for performing 6-DOF tasks than PointNet.

3.4.4 Trajectory Evaluation

In this part, we plotted predicted trajectories of all methods in our experiment for direct comparisons. All the visualization results are shown in Fig. 3.5.

Table 3.1: Test results on KITTI sequences: t_{rel} is averaged translation RMSE between prediction and ground truth, and r_{rel} is averaged orientation RMSE between prediction and ground truth. The measurement unit is meter (**m**).

	Sequence 04		**Sequence 10**	
Model	t_{rel}	r_{rel}	t_{rel}	r_{rel}
Two-stream [39]	0.0554	0.0830	0.0870	0.1592
ResNet18 [18]	0.1094	0.0602	0.1443	0.1327
DeepVO [51]	0.2157	0.0709	0.2153	0.3311
PointNet [41]	0.0946	0.0442	0.1381	0.1360
PointGrid [27]	0.0550	0.0690	0.0842	0.1523
DeepPCO (Ours)	**0.0263**	**0.0305**	**0.0247**	**0.0659**



(a) Sequence 04 (b) Sequence 10

Figure 3.5: Trajectories of testing results on Sequence 04 and 10.

For a clear representation, we only plot our work together with existing baseline approach [39]. The proposed DeepPCO system can produce accurate pose estimation compared to the ground truth. Compared to the baseline approach in the previous work, DeepPCO improves significantly the accuracy of the odometry task. As seen in Fig. 3.5, during the first 200 meters of Sequence 04, DeepPCO has very low drift. However, after 200 meters, the drift gradually increases. In order to investigate the reason, we plotted the projected depth images as shown in Fig. 3.6. The plot suggests that large open areas can degrade the accuracy of pose prediction to some degree. Feature-rich environments on the other hand, which contain multiple objects, can improve the prediction performance.

3.4.5 Comparison to Conventional Approach

In our experiments, we also compared our work to the open-sourced LOAM algorithm [62]. For a fair comparison, we did not use any IMU data. We employed the official evaluation

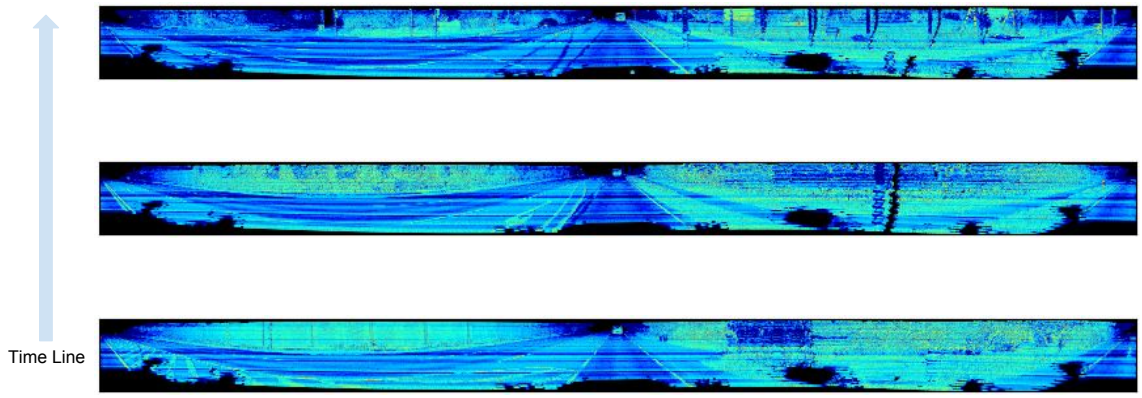


Figure 3.6: Sample projected depth images from sequence 04 in the latter time line. Images are plotted using depth values from range [0, 255]. The surroundings are vacant and lack features.

tool released for KITTI to examine these two approaches. We take sequence 04 as an example. The percentage of translation error and deg/m of rotation error of LOAM are 2.3245% and 0.0108, while results of DeepPCO are 3.1012% and 0.0177 respectively. We notice that the conventional approach slightly outperforms DeepPCO. One reason may be that training datasets are limited, especially for rotation-related data in the KITTI dataset. The other reason may be that we did not utilize any geometric awareness in our network, which can be further explored in the future.

3.4.6 Ablation Study

In order to explore the impact of various components of our DeepPCO, we conduct the following ablation experiments. In the first, we split each sub-network to individually infer 6-DOF poses and examine whether our parallel architecture can outperform the individual ones. In the second experiment, for each sub-network, we keep the convolutional layers the same but using combined fully connected layers instead of two separate branches of fully connected layers as in DeepPCO. Our purpose is to check whether the design of two branches for predicting translation and orientation separately is better than a single branch to jointly train and predict transformation. We note that all variants of our ablation architectures are trained from scratch.

a) ***Individual Sub-Network***: We want to investigate the performance of utilizing each individual sub-network of DeepPCO for pose estimations. Hence, we segmented our parallel architecture to two individual networks but kept all the parameter and hyper-parameter settings unchanged. In order to ensure the fairness of our experiments, we

Table 3.2: Testing results of individual sub-network on KITTI sequences: t_{rel} is averaged translation RMSE between prediction and ground truth, and r_{rel} is averaged orientation RMSE between prediction and ground truth. The measurement unit is meter (**m**). Each sub-network was trained using 6-DOF poses, and we conducted 5 times independently for each one.

	Translation Sub-Network Only				**FlowNet Orientation Sub-Network Only**			
	Sequence 04		**Sequence 10**		**Sequence 04**		**Sequence 10**	
Times	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}	t_{rel}	r_{rel}
1	**0.0288**	0.0421	**0.0243**	0.0735	0.0474	**0.0315**	0.0352	**0.0648**
2	**0.0282**	0.0392	**0.0235**	0.0731	0.0471	**0.0312**	0.0373	**0.0675**
3	**0.0262**	0.0429	**0.0248**	0.0737	0.0469	**0.0297**	0.0377	**0.0682**
4	**0.0307**	0.0417	**0.0234**	0.0755	0.0469	**0.0302**	0.0342	**0.0676**
5	**0.0281**	0.0426	**0.0247**	0.0740	0.0580	**0.0309**	0.0447	**0.0669**
Mean	**0.0284**	0.0417	**0.0241**	0.0740	0.0493	**0.0307**	0.0378	**0.0670**

Table 3.3: Testing results of various branches of fully connected (FC) layers on KITTI sequences: t_{rel} is averaged translation RMSE between prediction and ground truth, and r_{rel} is averaged orientation RMSE between prediction and ground truth. The measurement unit is meter (**m**). For single branch FC layers, each sub-network was trained using full 6-DOF pose while for DeepVO, each sub-network was learned by two branches, one for translation and the other for orientation.

	Sequence 04		**Sequence 10**	
Model	t_{rel}	r_{rel}	t_{rel}	r_{rel}
Single Branch FC	0.0559	0.0843	0.0327	0.0729
Two Branches FC	**0.0284**	**0.0307**	**0.0241**	**0.0670**

randomly run 5 times for each sub-network to evaluate translation and orientation on sequence 04 and 10 respectively. Meanwhile, we calculated average of RMSE for the 5 experiments. All the results are shown in Table 3.2. Results suggest that all translation predictions on Sequence 04 and 10 using Translation Sub-Network are better than FlowNet Orientation Sub-Network, while all orientation predictions based on FlowNet Orientation Sub-Network are better than those generating from Translation Sub-Network. In other words, this ablation experiment demonstrated that Translation Sub-Network performs better for inferring translation whereas achieving worse orientation prediction than FlowNet Orientation Sub-Network. Conversely, FlowNet Orientation Sub-Network outperforms Translation Sub-Network for orientation inference. Therefore, it is a desirable choice for designing dual sub-networks to predict translation and orientation separately.

b) ***Single Branch Fully Connected Layers*** In our proposed architecture, we utilized two branches for predicting translation and orientation. However, we may ask whether

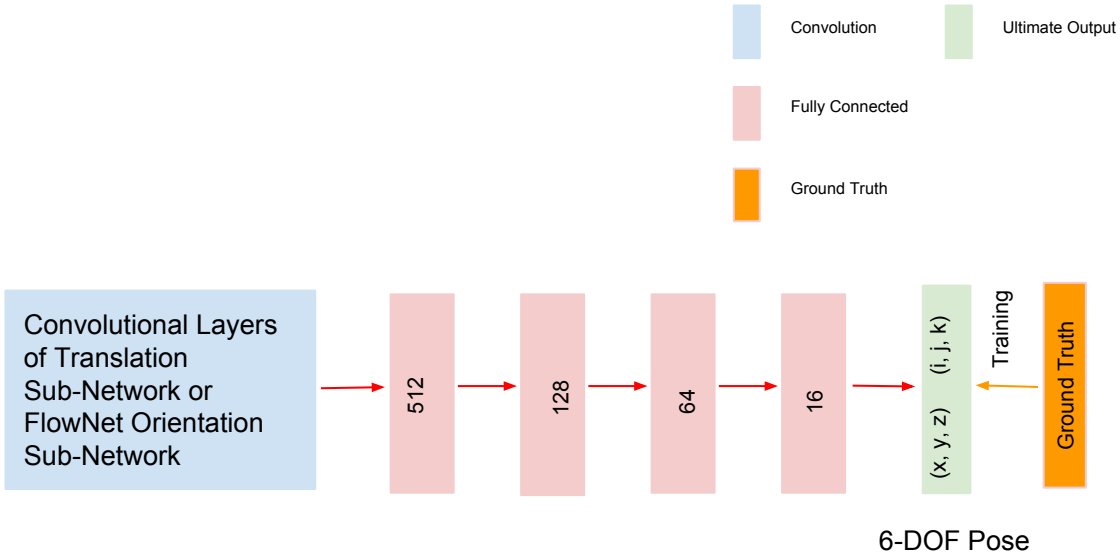


Figure 3.7: Ablation experiment of single branch fully connected layers. All the parameter configurations of convolutional layers and fully connected layers are the same as DeepPCO. Different from DeepPCO in which transformation vector is trained using two branches, 3-D translation (x, y, z) and orientation (i, j, k) are jointly trained and inferred by just one branch here.

only using a single branch of fully connected layers, each sub-network can produce a better result. Thus, we design the ablation study for this situation. All the parameter configurations of layers remain same. The only difference is that we use one branch to jointly learn transformation. The results are presented in Table 3.3, and the visualization of the experiment design is depicted in Fig. 3.7. While the variation of single branch FC layers are capable of attaining reasonable estimations, they are less accurate than our parallel architecture. Results indicate that adopting the design of two branches of FC layers in DeepPCO tends to have better transformation inference for PCO task.

3.5 Summary

In this chapter, we first introduce the background and motivation of LiDAR odometry, and then define the deep LiDAR odometry task. We also discuss the data encoding approaches of LiDAR point clouds. We propose an end-to-end deep LiDAR odometry system DeepPCO, which outperforms state-of-the-art approaches based on comprehensive experiments. Initially, we develop neural networks to predict the 6-DoF poses simultaneously. However, the designed neural network is only suitable for predicting either translations or rotations.

Inspired by the experiments, we propose to develop different neural networks to predict them separately. Thus, this work indicates that designing two sub-networks to predict translation and rotation respectively is a promising strategy to improve the accuracy of odometry estimation. Meanwhile, different LiDAR point clouds encoding approaches can generate different results, so it is important to choose the appropriate encoding approach for the task. The intuitive way to learn features from point clouds is to utilize 3-D CNN for point clouds. However, there are around 120,000 points of one point cloud in our experiments, so it is difficult to train the 3-D CNNs for our point clouds. Therefore, we choose the projected depth images as inputs to feed into neural networks, and we observe that the training speed is very fast. The advantages of the projected encoding method of point clouds are that it can significantly facilitate the training speed compared to 3-D CNN approaches, and the pre-trained CNN on large-scale image datasets can be employed to improve the performance of localization. The disadvantage of this method is that it may lose the spatial information since point clouds are transformed from 3-D to 2-D. For point clouds with the large number of points, our work reveals that the projected depth images from point clouds can effectively perform odometry estimation. In the next chapter, we will discuss the deep LiDAR relocalization approach.

Chapter 4

LiDAR Relocalization with Deep Neural Networks

4.1 Introduction

LiDAR-based relocalization methods have achieved impressive accuracy [25, 28]. 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 [34]. 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 [4, 34, 57]. 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.

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.* [48] proposed a point cloud retrieval-based localization method to deal with the situation when the

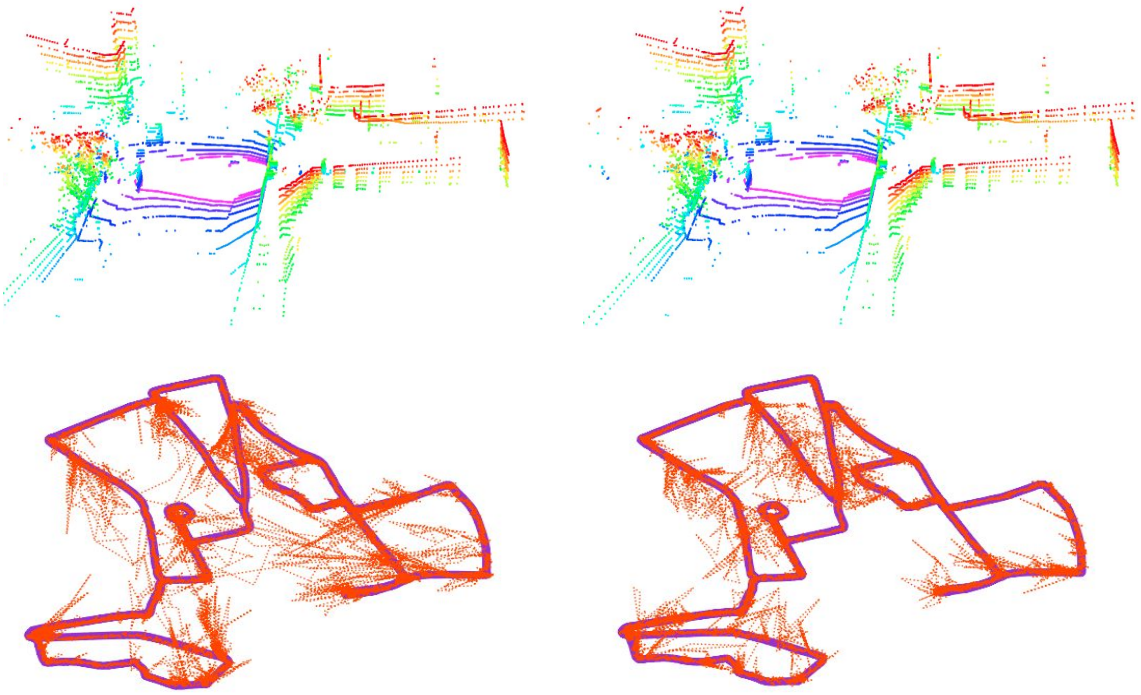


Figure 4.1: PointLoc results for the challenging Oxford Radar RobotCar dataset [1, 35]. We directly feed the LiDAR point cloud from a single timestamp to the neural network for predicting the 6-DoF pose without the requirement of pre-built maps. The estimations of PointLoc are robust regardless of weather, and outperform the state-of-the-art DNN-based LiDAR and visual relocalization methods significantly.

GNSS is absent. It obtains a 6-DoF pose with respect to the pre-built map in the form of reference database. Dube *et al.* [13] further proposed SegMap to improve the storage efficiency of the reference database by storing data-driven descriptors of individual objects in a point cloud. 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(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(n) storage space and cannot be deployed on many mobile robots. Third, the recall rate of it is often not good enough [48].

Recently, learning-based approaches have emerged as a promising tool to build up a completely end-to-end localization system. In Chapter 3, the proposed DeepPCO demonstrates the ability of applying deep learning to develop the end-to-end localization system, which inspires us to explore the application of deep neural networks for relocalization.

For the relocalization task, learning-based 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, 23, 24, 45]. However, all the current pose regression approaches utilize RGB images as inputs, which have several problems. Visual inputs 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, point clouds, acquired by LiDAR, 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 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 LiDAR point clouds. 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 point cloud and camera relocalization methods. Fig.4.1 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 point cloud-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 will release the collected indoor LiDAR-image dataset dubbed vReLoc 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 point cloud and visual relocalization methods by a large margin.

4.2 Problem Statement

We design a DNN-based framework for performing deep global pose regression using point cloud data from a LiDAR sensor. 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 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 $\mathbf{x}_i$ 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 $\mathscr{F}$ such that $\mathscr{F}(\mathbf{P}_t) = (\mathbf{t}, \mathbf{r})^T$, where the function $\mathscr{F}$ is usually a neural network for DNN-based methods.

4.3 Method Details

4.3.1 Deep Point Cloud Relocalization

This section introduces our proposed PointLoc, a deep 3-D pose regressor for predicting the global pose from point cloud data. The overall architecture is illustrated in Fig. 4.2. 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.

4.3.1.1 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 [1, 35]in our experiments is around 21,000 and we want to keep the information as much as possible.

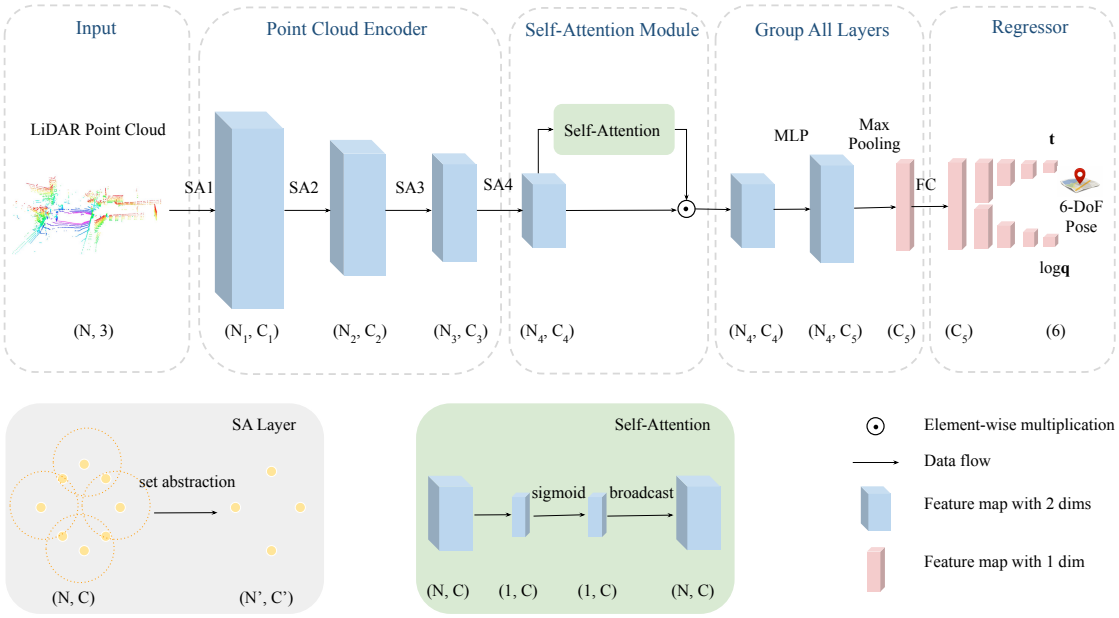


Figure 4.2: The architecture of PointLoc. It consists of a point cloud encoder, a self-attention module, a grouping all (GA) layers module and a pose regressor. Numbers in the parentheses represent the dimensions of feature tensors. (N, 3) represents the 3-D coordinates x, y and z of a point cloud. The encoder is composed of 4 consecutive set abstraction (SA) layers. Each SA layer shown in the diagram [32] consists of a sampling layer, a grouping layer and a PointNet layer [41]. For more details about the SA layer, please refer to PointNet++ [41] and FlowNet3D [32]. 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.

4.3.1.2 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 [41, 65] 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 transforma-

tion 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:

$$f(x_1,...,x_N) = \phi(\mathbf{MAX}\{h(x_i) \mid x_i \in \mathbf{P}\}) \tag{4.1}$$

where ϕ and h are two continuous functions (they are usually instantiated to be an MLP), and **MAX** denotes the max pooling layer [42]. PointNet++ extends PointNet by recursively capturing the hierarchical features on point sets in a metric space [42]. From the aforementioned Eq. 4.1, 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 [41,65]). 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++ [40, 42]. 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 [41]. 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. 4.2). We also a leverage multi-scale grouping strategy [42] 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 [32]:

$$\mathbf{F_j^{'}} = \mathbf{MAX}_{\{i \mid ||x_i - x_j|| \leq r\}}\{h(\mathbf{F}_i, x_i - x_j)\} \tag{4.2}$$

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 **MAX** is the max pooling layer.

4.3.2 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 [21,50] 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 [21, 50, 58], 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 $M \in R^{1 \times C}$ 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 M. After that, we broadcast and mask the features $\mathbf{F}$ by 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:

$$\hat{\mathbf{F}} = \mathbf{F} \cdot M \tag{4.3}$$

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

4.3.3 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. 4.2, 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.

4.3.4 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.

4.3.5 Loss Function

Our goal is to estimate the 6-DoF pose $[\mathbf{t},\mathbf{r}]^T$. Prior works [10, 11, 23, 24] 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 [6]. Odometry tasks with DNNs [51, 53] 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 [6] for training our neural network, which is adapted from [23]. Given K training samples $\mathscr{G} = \{\mathbf{P}_t \mid t = 1,...,K\}$ 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:

$$\mathscr{L}(\mathscr{G}) = \|\mathbf{t}-\hat{\mathbf{t}}\|_1 e^{-\beta} + \beta + \|\log\mathbf{q} - \log\hat{\mathbf{q}}\|_1 e^{-\gamma} + \gamma \tag{4.4}$$

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:

$$\log\mathbf{q} = \begin{cases} \frac{\mathbf{v}}{\|\mathbf{v}\|}\cos^{-1}u, & \text{if } \|\mathbf{v}\| \neq 0 \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{4.5}$$

4.4 New Dataset

There is a lack of public datasets in the indoor environment with point cloud data. In order to boost the research in this area, we collected a new dataset dubbed vReLoc with rich sensor modalities, e.g. vision and LiDAR data on a mobile robot platform. Our dataset has been released online to benefit future researchers.

The experimental robot is Turtlebot 2, mounted with a Velodyne HDL-32E LiDAR 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 ~60,000 points. The camera was employed to capture RGB images, and the size of each image is $640\times480\times3$. A Vicon Motion Tracker system is leveraged for acquiring accurate ground truth 6-DoF poses. 10

Table 4.1: 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		✓

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 and LiDAR 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 4.1.

4.5 Experiments

In this section, we evaluate our proposed approach on the recently released outdoor Oxford Radar RobotCar [1, 35] dataset and our proposed indoor vReLoc dataset and compare to state-of-the-art methods.

4.5.1 Implementation Details

Adam [26] 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 [6] and AtLoc [50].

Table 4.2: 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 4.3: Parameter Setting of the Self-Attention Module.

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

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 [6, 23, 24, 50], 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 4.2. 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 4.3, we present the parameter settings of the self-attention module. The parameter setting of the group all layers is shown in Table 4.4. We show the parameter setting of the pose regressor in Table 4.5. 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.

4.5.2 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 re-

Table 4.4: 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 4.5: 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

localization approaches. For LiDAR sensor localization approaches, we choose PointNetVLAD [48] and Deep Closest Point (DCP) [55]. 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 [41] and DGCNN [56] 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 [23] since it outperforms PoseNet and Bayesian PoseNet in previous works. AtLoc [50] is selected for comparison since it is the state-of-the-art single image-based learning approach. We also choose LSTM-Pose [49] as the sequential baseline. Moreover, we also compare with MapNet [6] 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.

4.5.3 Results on the Oxford Radar RobotCar

The Oxford Radar RobotCar dataset [1] is a radar extension to the Oxford RobotCar dataset [35], 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).

Table 4.6: 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			✓

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 5.1.

Results The test results of the Radar RobotCar are presented in Table 4.7. Following the plotting style of relocalization work [21], the trajectories of FULL7 and FULL8 of DCP, MapNet, AtLoc, and PointLoc are shown in Fig. 4.3. PointLoc is much better than the LiDAR point cloud retrieval-based approach PointNetVLAD in terms of accuracy, which proves the effectiveness of our proposed network. As seen from Table 4.8, 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, PointLoc outperforms DCP with PointNet significantly, which reveals that the proposed embedding neural network can effectively learn meaningful features for relocalization. For DCP with DGCNN, the whole neural network

Table 4.7: 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 [48]	DCP [55]	PoseNet17 [23]	LSTM-Pose [49]	MapNet [6]	AtLoc [50]	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°**

Table 4.8: Comparisons of the computational time and storage space of PointLoc and PointNetVLAD.

	Computational Time	Storage Space
PointLoc	0.0985 sec	13 MB
PointNetVLAD	38.0242 sec	283 MB (76 MB model size and 207 MB reference db size)

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. The 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, PointLoc can have stable estimation across the test dataset, which indicates that the point cloud relocalization method is more robust than the visual relocalization.

4.5.4 Results on a Real-World Indoor Robot

We also validate our proposed PointLoc on the real-world indoor robot dataset. Our experimental design 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. vReLoc will be released to help support future work on relocalization.

The median errors and trajectories of test results using our PointLoc are plotted in Fig. 4.4. The results demonstrated that PointLoc can be successfully applied to the real-

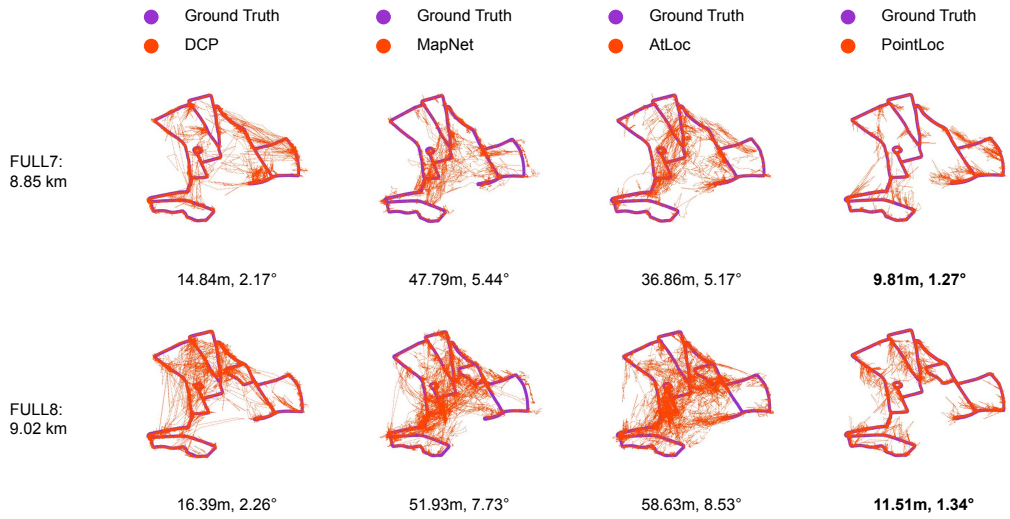
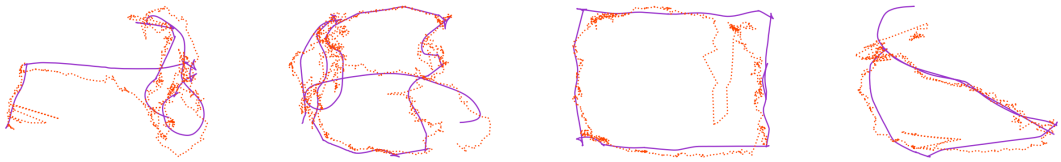


Figure 4.3: Trajectories of DCP, MapNet, AtLoc and the proposed PointLoc on FULL7 and FULL8 with mean translation error (m) and rotation error (°). The darkorchid line is the ground truth poses, and the orange-red dot line shows the estimated poses. Our PointLoc outperforms the existing LiDAR and camera relocalization approaches by a significant margin.

world indoor scenarios for robot localization.

4.5.5 Ablation Study

To explore the impact of different components of PointLoc, we conduct the ablation studies in Table 4.9. 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.



(a) Seq-05 (0.12m, 3.00°) (b) Seq-06 (0.10m, 2.97°) (c) Seq-07 (0.13m, 3.47°) (d) Seq-14 (0.11m, 2.84°)

Figure 4.4: 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 4.9: Ablation study 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**°

4.6 Summary

In this chapter, we first introduce the background and motivation of LiDAR relocalization, and then define the deep LiDAR relocalization task. We propose an end-to-end deep LiDAR relocalization system PointLoc, which consists of point cloud pre-processing, point cloud encoder, self-attention module, group all layers module and pose regressor. It outperforms state-of-the-art approaches based on comprehensive experiments. The method of feature extraction is crucial to the performance of localization. We first choose the projected depth images as inputs to learn features for relocalization as shown in DeepPCO. However, the projected depth image is too sparse, which leads to poor relocalization performance. Considering the number of points in our experiments is around 20,000, it is not suitable for applying Voxel-based neural networks. Therefore, we choose the PointNet-style architecture for relocalization, which can effectively and efficiently learn features. PointNet-style neural networks have benefits of training speed of neural networks, training storage space of GPUs, and prediction accuracy of our task. Meanwhile, outliers like moving objects can significantly affect the relocalization performance, and it is difficult to remove them. Inspired by the self-attention mechanism, we design the self-attention module to learn to mask outliers. In other words, we expect the neural network to learn to remove outliers from training datasets by itself. Furthermore, our work reveals the potential to apply deep

learning for relocalization, which is time and storage effective since we only need to store the neural network itself. In the next chapter, we will discuss the deep radar relocalization approach.

Chapter 5

Radar Relocalization with Deep Neural Networks

5.1 Introduction

Relocalization is a fundamental problem in robotics and computer vision. A robot has to localize itself when moving in urban or indoor environments to achieve competent autonomy. There is a significant body of knowledge in visual localization, as it has been studied for decades. Conventional geometry-based visual localization systems mainly utilize handcrafted features and descriptors, which are typically sensitive to illumination variation, dynamic objects and viewpoint change [21]. Recently, learning-based visual localization methods such as PoseNet and variants [6, 22–24] have been proposed to solve these challenges, which leverage either a single image or a sequence of images to predict 6-Degree-of-Freedom (6-DoF) poses directly. In Chapter 4, we also introduce the learning-based LiDAR relocalization approach.

Essentially, these deep learning methods attempt to perform a one-to-one mapping between images and 6-DoF poses. Location-related information is implicitly encoded within the parameters of these deep neural networks, and therefore these methods require agents that have previously traversed the same environment. However, vision sensors inherently suffer from several drawbacks which restrict their ability to be used in scenarios where reliability is highly desirable, such as self-driving cars. Visual inputs are easily impacted by ambient environmental conditions e.g. sunshine, rain, fog; and further by their narrow Field-of-View (FoV). Meanwhile, LiDAR sensors are also vulnerable in environments like fog, which restrict their ability to be applied in real world such as LiDAR relocalization discussed in Chapter 4.

Emerging Frequency-Modulated Continuous Wave (FMCW) radar sensors can effectively solve many of the shortcomings of cameras and LiDAR sensors. They can provide a

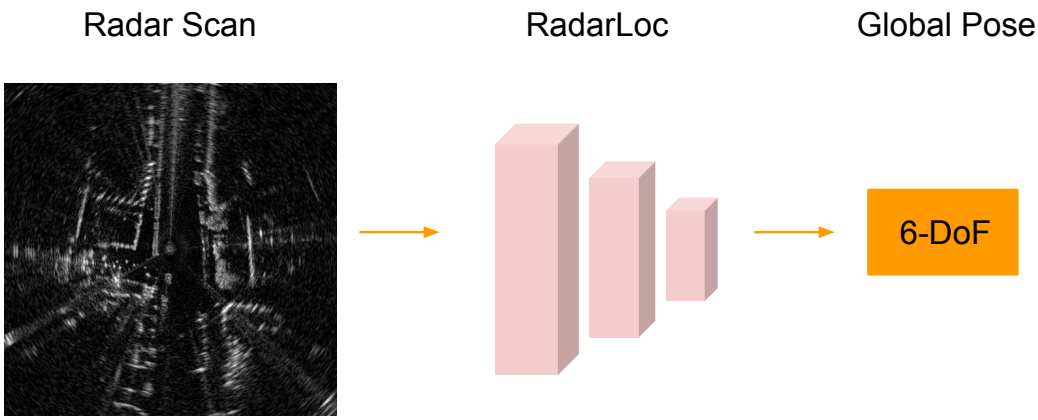


Figure 5.1: System overview of the proposed RadarLoc relocalization framework. A raw FMCW radar scan is first transformed into a Cartesian Radar Image. The radar image is then fed to RadarLoc, which directly estimates the 6-DoF pose in an end-to-end manner.

360° view of the scene and range objects hundreds of meters away. Meanwhile, they can function reliably in unstructured environments in different conditions e.g. snow, darkness, fog, smoke, direct sunlight [1] without impact. These characteristics of radar make it suitable for robot localization, especially for autonomous agents which operate in large-scale urban scenes. Inspired by the aforementioned deep pose regression methods that use images and LiDAR point clouds, the aim of this work is to investigate and provide a robust radar localization system, allowing robots to relocalize themselves under previously visited scenes.

Specifically, we propose a novel geometry-aware neural network architecture, termed RadarLoc, which can estimate the 6-DoF pose using a single radar scan. The proposed self-attention module of a nested encoder-decoder architecture further improves the localization performance. During the training phase, RadarLoc takes as input a sequence of radar scans, and predicts poses optimized and constrained by both absolute and relative (geometric) pose losses. At inference time the 6-DoF pose is regressed from a single input scan. Fig. 5.1 illustrates the overview of the proposed fully differentiable relocalization system.

Our contributions are summarized as follows: We demonstrate that radar scans can be employed to estimate absolute 6-DoF poses in an end-to-end fashion. We further refine pose estimations by leveraging geometric constraints between radar pairs as one component of the loss function. Comprehensive experiments and ablation study have been done to demonstrate the effectiveness of RadarLoc, which outperforms state-of-the-art radar-based localization, DNN-based camera relocalization methods by a significant margin.

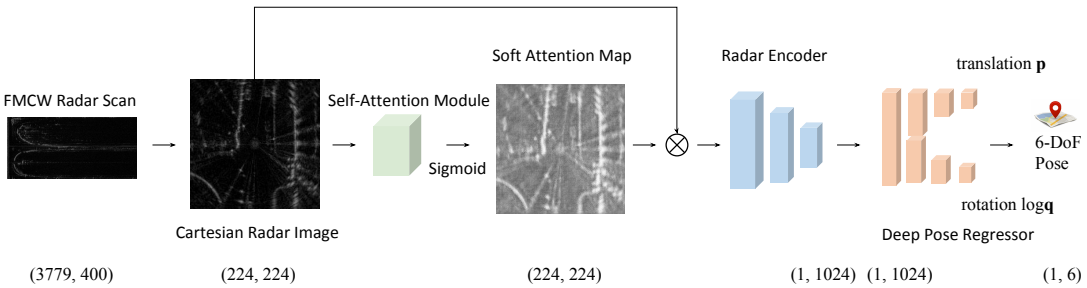


Figure 5.2: The architecture of RadarLoc. RadarLoc consists of a self-attention module, a radar encoder and a deep pose regressor. A raw FMCW radar scan is transformed into a Cartesian radar image, and then it is fed into a self-attention module to learn a soft attention map. DenseNet [20] is employed as the radar encoder to extract useful features for relocalization. The deep pose regressor predicts the parameterized translation $\mathbf{p} \in R^3$ and rotation $log\mathbf{q} \in R^3$ [6].

5.2 Problem Statement

The scope of this work is to predict absolute 6-DoF poses of the mobile agent given radar scans as inputs. The scene has been visited by the agent before, in which the agent can relocalize itself. The relocalization of the agent is parameterized by a 6-DoF pose $\mathbf{P} = [\mathbf{p}, \mathbf{q}]$ with respect to the world coordinate, where $\mathbf{p} \in R^3$ is a 3-D translation vector and $\mathbf{q} \in R^4$ is a 4-D rotation vector. At each timestamp t, the agent receives a Cartesian Radar image $\mathbf{I} \in R^{H \times W}$ from the FMCW scanning radar where H is height and W is width. The deep radar relocalization framework learns a function f so that $f(\mathbf{I}) = [\mathbf{p}, \mathbf{q}]$, where f is a deep neural network.

5.3 Method Details

In this section, we introduce the proposed deep radar relocalization framework in detail. The overall architecture of RadarLoc is illustrated in Fig. 5.2, which consists of a self-attention module, a radar encoder, and a deep pose regressor. Since the original output of the FMCW scanning radar is a polar image, we transform it into the Cartesian space as a grey-scale image for better representation. During training phase, the neural network is optimized by the geometry-aware loss function which employs a sequence of radar scans to learn global 6-DoF poses and relative transformations simultaneously. During test phase, the RadarLoc estimates the 6-DoF pose of a single Radar input each time.

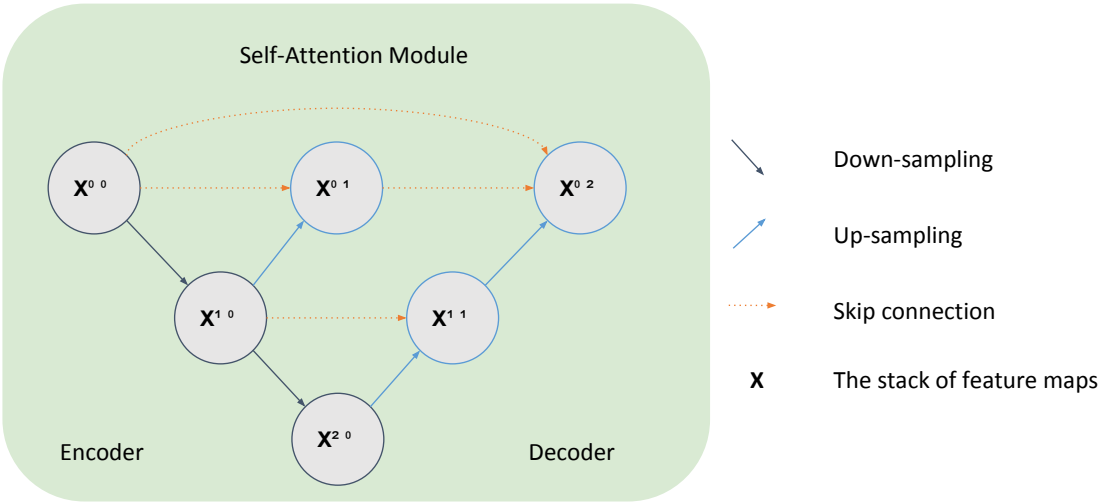


Figure 5.3: The architecture of self-attention module. The self-attention module is a nested encoder-decoder structure [67]. For better visualization, we only depict 3 levels in the figure, and in our implementation, the nested structure has 6 levels. We adopt a soft attention mechanism, and fuse output features at the upper level to have a fine-grained attention map.

5.3.1 Self-Attention for Robust Relocalization

For the radar relocalization task, there are two categories of noises which can significantly affect the accuracy of pose predictions. One is noises from the radar sensor itself. The current FMCW scanning radar is affected by multiple noises, e.g. range error, angular error, and false positive and false negative detection which make the radar scans noisier than camera images. The other is the foreground moving objects in dynamic environments. There are several types of dynamic outliers e.g. pedestrians, bikes, buses, trucks in the complex urban environments, which have different shapes and sizes. Since the radar can scan more than 150 meters range, it is likely that one radar image can contain these different types of moving objects. Therefore, the aforementioned noises can inevitably bias the neural network, making the radar relocalization quite challenging. Barnes *et al.* [3] proposed a U-Net structure to predict distraction-free radar odometry. Wang *el al.* [50] designed a non-local self-attention module to filter out moving objects for camera relocalization. However, these methods neither learn semantic features in a fine-grained manner [3] nor are designed specifically for radar images [50]. To this end, we propose a novel self-attention module for radar relocalization as shown in Fig. 5.3, which is a nested encoder-decoder style neural network, to mitigate the impact of these noises by filtering them out. Our design intuition is that considering the different shapes and sizes of moving objects, the self-attention module should have the ability to extract fine-grained features and filter out these dynamic

noises. Compared to the U-Net style architecture, the nested encoder-decoder architecture can gradually down-sample, fuse and up-sample features from inputs, which can reduce the semantic gap between the feature maps and extract fine-grained semantic information. We choose the re-designed skip pathways proposed by Zhou *et al.* [67] due to its impressive performance on multiple medical image segmentation tasks, in which the feature maps pass through a dense convolution block whose number of convolution layers depends on the pyramid level, and the stack of the feature maps are calculated as:

$$\mathbf{X}^{i,j} = \begin{cases} g(\mathbf{X}^{i-1,j}), & j = 0 \\ g([[\mathbf{X}^{i,k}]_{k=0}^{j-1}, g^{'}(\mathbf{X}^{i+1,j-1})]), & j > 0 \end{cases} \tag{5.1}$$

where $\mathbf{X}^{i,j}$ is the extracted feature of each node in Fig. 5.3 , $g(*)$ denotes a convolution layer with an activation function, $g^{'}(*)$ is an up-sampling layer, $[*]$ indicates a concatenation layer, $i \in [0,1,..,n-1]$ is the index of the down-sampling layer along the encoder, $j \in [0,1,...,n-1]$ is the index of convolution layer of the dense block along the skip pathway, n is the number of pyramid levels. In order to learn features at different scales, we fuse node outputs on the uppermost level to generate the output features $\mathbf{I}_{node}$ by averaging them:

$$\mathbf{I}_{node} = \frac{1}{n}\sum_{j=0}^{n-1}\mathbf{X}^{0,j} \tag{5.2}$$

Thus, our self-attention module is an encoder-decoder pyramid structure with densely skip pathways followed by an activation function. We adopt a soft attention mechanism to learn the mask, so the activation function we use is *Sigmoid*. Given a Cartesian radar image $\mathbf{I} \in R^{H\times W}$, the self-attention module serves to learn a noise-free feature map $\mathbf{I}^{'} \in R^{H\times W}$:

$$\mathbf{I}^{'} = \sigma(\mathbf{I}_{node}) \cdot \mathbf{I} \tag{5.3}$$

where σ is the *Sigmoid* function, and $\cdot$ represents the dot product.

5.3.2 Radar Encoder

The radar encoder extracts features from a radar image for relocalization. Existing state-of-the-art camera relocalization approaches [6, 21, 50] employ ResNet [18] as the visual encoder considering the residual neural networks can learn deeper and alleviate the gradient vanishing problem. DenseNet [20], which consists of densely connected convolutional networks, has been proved better performance on four object recognition tasks than ResNet. Hence, RadarLoc adopts pre-trained DenseNet as the radar encoder for feature extraction of the relocalization. We broadcast feature map $I^{'}$ to 3 channels, and replace the last 1000-dimensional fully connected layer with a M-dimensional fully connected layer. Formally,

given the $\mathbf{I}^{'}$ from the self-attention module, the feature encoder $f_{encoder}$ extracts the feature vector $\mathbf{z} \in R^{M\times 1}$ from $\mathbf{I}^{'}$, which can be presented as:

$$\mathbf{z} = f_{encoder}(\mathbf{I}^{'}) \tag{5.4}$$

5.3.3 Deep Pose Regressor

The deep pose regressor receives the feature vector $\mathbf{z}$ from the Radar Encoder, and predicts the position $\mathbf{p}$ and the rotation $\mathbf{q}$ respectively. It consists of Multi-Layer Perceptrons (MLPs) of two branches. An activation function is applied to each layer of the MLPs except the last one. The pose regressor which ultimately estimates the global pose $\mathbf{P} = [\mathbf{p}, \mathbf{q}]$ is defined as:

$$\mathbf{P} = f_{MLPs}(\mathbf{z}) \tag{5.5}$$

5.3.4 Loss Function with Geometric Constraints

For the loss function, we employ the definition in [6] as it has been shown to be effective in existing image-based global pose regression tasks. The vanilla loss function h is defined as:

$$h(\mathbf{P}, \hat{\mathbf{P}}) = \|\mathbf{p} - \hat{\mathbf{p}}\|_1 e^{-\beta} + \beta + \|\log \mathbf{q} - \log \hat{\mathbf{q}}\|_1 e^{-\gamma} + \gamma \tag{5.6}$$

where $\mathbf{p}$ and $\log \mathbf{q}$ are translation and orientation of the predicted global pose $\mathbf{P}$, $\hat{\mathbf{p}}$ and $\log \hat{\mathbf{q}}$ are translation and orientation of the ground-truth global pose $\hat{\mathbf{P}}$, $\| * \|_1$ denotes the L_1 loss function, β and γ are learnable balance factors which are initiated 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, which is defined as:

$$\log \mathbf{q} = \begin{cases} \frac{\mathbf{v}}{\|\mathbf{v}\|} \cos^{-1} u, & \text{if } \|\mathbf{v}\| \neq 0 \\ \mathbf{0}, & \text{otherwise} \end{cases} \tag{5.7}$$

Since a 2-D radar image can provide metric information within a wide range, we further improve the performance of relocalization by leveraging geometric constraints to optimize parameters of the neural network. During training, we choose N radar images, consisting of the current radar image I_0 as well as N-1 sequential radar images $\{I_1, ..., I_{N-1}\}$ close to I_0. Consequently, RadarLoc learns both global poses ($\mathscr{L}_{gp}$) and relative pose transformations ($\mathscr{L}_{rp}$) between radar image pairs. The improved loss functions are defined as:

$$\mathscr{L}_{gp} = \sum_{i=0}^{N-1} h(\mathbf{P}_i, \hat{\mathbf{P}}_i) \tag{5.8}$$

Table 5.1: Dataset Descriptions on the Oxford Radar RobotCar.

Scene	Time	Tag	Training	Test
Seq-01	2019-01-11-14-02-26	sun	✓	
Seq-02	2019-01-14-12-05-52	overcast	✓	
Seq-03	2019-01-14-14-48-55	overcast	✓	
Seq-04	2019-01-15-14-24-38	overcast	✓	
Seq-05	2019-01-18-15-20-12	overcast	✓	
Seq-06	2019-01-10-11-46-21	rain		✓
Seq-07	2019-01-14-12-41-28	overcast		✓
Seq-08	2019-01-15-13-06-37	overcast		✓
Seq-09	2019-01-17-14-03-00	sun		✓
Seq-10	2019-01-18-14-14-42	overcast		✓

$$\mathscr{L}_{rp} = \sum_{i=0}^{N-2} h(\mathbf{Q}_i, \hat{\mathbf{Q}}_i) \tag{5.9}$$

where $\mathbf{P}_i, \mathbf{Q}_i$ are predicted global poses and relative pose transformations while $\hat{\mathbf{P}}_i, \hat{\mathbf{Q}}_i$ are ground-truth global poses and relative pose transformations respectively, and h is the distance function defined in Eq. 5.6. Therefore, the ultimate loss function for RadarLoc is formulated as:

$$\mathscr{L}_{total} = \mathscr{L}_{gp} + \mathscr{L}_{rp} \tag{5.10}$$

Importantly, note that we employ multiple images in the training phase, and only a single radar image in the test phase.

5.4 Experiments

In this section, we evaluate our proposed RadarLoc on the recently released Oxford Radar RobotCar Dataset [1, 35], and compare it with state-of-the-art radar-based localization and deep camera and LiDAR relocalization methods.

5.4.1 Dataset

The Dataset provides Navtech CTS350-X FMCW scanning radar data, RGB images and corresponding ground truth poses. It was collected in January 2019 over thirty-two traversals of a central Oxford route spanning a total of 280 km of urban driving, and covered different kinds of lighting, weather and traffic conditions [1]. Therefore, the dataset is large-scale and complex. For the relocalization task, it is quite challenging since the urban scenes encompass a variety of foreground objects e.g. people, car, bus, which significantly influence the performance of relocalization. The descriptions of our training sequences and test sequences from the Oxford Radar RobotCar Dataset are illustrated in Table 5.1.

Table 5.2: Results showing the mean translation error (m) and rotation error (°) for state-of-the-art radar-based localization methods and deep camera and LiDAR relocalization methods on the Oxford Radar RobotCar Dataset. For RadarSLAM and Adapted methods, the sensory data is FMCW radar scan. The sensory data of AtLoc and PointLoc are camera RGB image and LiDAR point cloud respectively.

Sequence	RadarSLAM [19] [Radar]	Adapted Masking [3] [Radar]	Adapted PoseNet17 [23] [Radar]	Adapted AtLoc [50] [Radar]	Adapted LSTM [49] [Radar]	AtLoc [50] [RGB]	PointLoc [54] [LiDAR]	RadarLoc (Ours) [Radar]
Seq-06	49.81m, 5.22°	12.54m, 3.93°	15.12m, 4.08°	15.85m, 4.20°	15.86m, 4.28°	15.36m, 3.37°	14.42m, **2.77**°	**8.43m**, 3.44°
Seq-07	24.73m, 3.36°	8.11m, 3.04°	13.59m, 3.54°	13.23m, 3.82°	13.33m, 2.47°	39.76m, 8.31°	8.46m, **1.82** °	**5.12m**, 2.87°
Seq-08	26.09m, 1.57°	11.32m, 4.18°	14.81m, 3.46°	14.17m, 2.94°	14.86m, 2.88°	31.68m, 4.34°	9.52m, **2.14**°	**6.56m**, 3.06°
Seq-09	39.84m, 5.67°	11.53m, 2.76°	14.44m, 3.04°	15.71m, 3.23°	13.86m, 2.71°	47.06m, 9.38°	11.52m, **1.98**°	**6.51m**, 2.91°
Seq-10	17.83m, 1.71°	9.42m, 1.81°	13.21m, 2.02°	13.22m, 1.94°	14.65m, 1.89°	10.35m, **1.26**°	8.43m, 1.40°	**5.34m**, 1.78°
Average	31.66m, 3.50°	10.58m, 3.15°	14.23m, 3.23°	14.44m, 3.22°	14.51m, 2.85°	28.84m, 5.33°	10.47m, **2.02**°	**6.39m**, 2.81°

5.4.2 Implementation

The spatial dimensions of the self-attention module of RadarLoc are 8, 16, 32, 64, 128 and 256 respectively. The size of a Cartesian radar image is set to 224×224 in order to utilize the pre-trained DenseNet on ImageNet. For all experiments, the number of training epochs is set to 100, and we tune all baseline methods for the best performance. The learning rate is set to 1×10^{-4}, and we set the initial values of $\beta_0 = 0.0$ and $\gamma_0 = -3.0$. Furthermore, we retrieve a sequence of $N = 4$ radar images each time. For all methods, Adam [26] optimizer is applied to the neural networks.

5.4.3 Baselines

We compare RadarLoc with both radar-based methods and state-of-the-art RGB and Lidar techniques. We also adapt learning-based visual relocalization pipelines to use radar images as input. RadarSLAM is a full radar-based graph SLAM system for reliable localization in large-scale scenarios. Masking by Moving [3] is the state-of-the-art deep learning-based radar odometry approach, and we adapt the feature extraction module for relocalization. PoseNet17 [23], LSTM-Pose [49], and AtLoc [50] are state-of-the-art camera image-based deep relocalization methods, and since our radar scan can be seen as a 2-D 224×224 grey-scale image, we want to examine the performance of these architectures on the radar inputs. We apply these neural networks to radar images for adapted radar relocalization. AtLoc (RGB) is the state-of-the-art deep camera relocalization method, and PointLoc (LiDAR) is the state-of-the-art DNN-based LiDAR point cloud relocalization method.

5.4.4 Results

The experimental results are illustrated in Table 5.2, and the qualitative comparisons are depicted in Fig. 5.4. From Table 5.2, the proposed RadarLoc outperforms radar-based

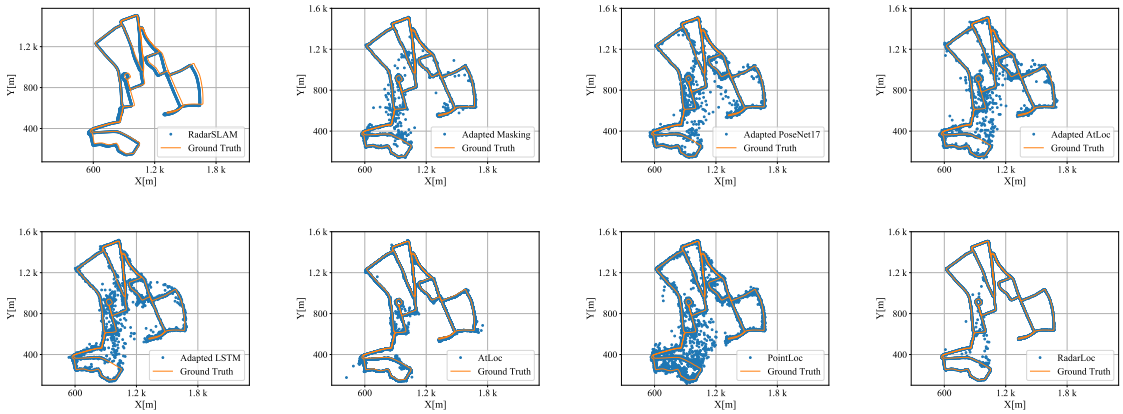
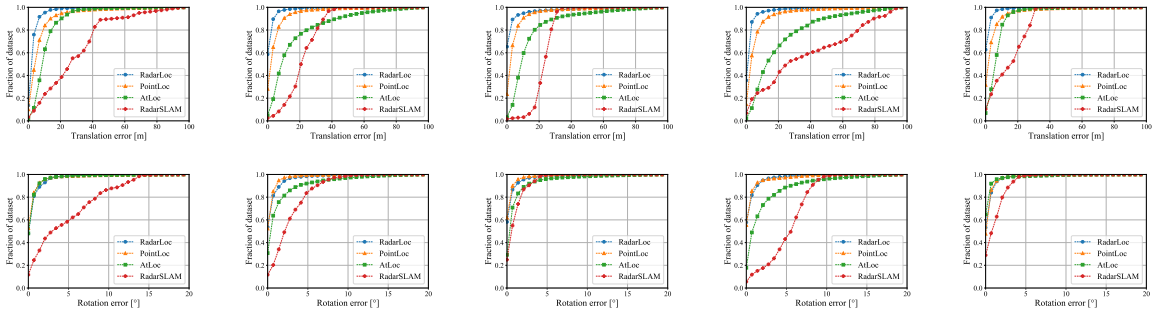


Figure 5.4: Visual comparisons of all localization approaches for Sequence 10. Poses were projected from 6-DoF to 3-DoF with exception to RadarSLAM, which outputs 3-DoF poses originally.



(a) Sequence 06 (b) Sequence 07 (c) Sequence 08 (d) Sequence 09 (e) Sequence 10

Figure 5.5: Cumulative distributions of translation and rotation errors.

methods by a significant margin. RadarSLAM can predict consecutive poses but accumulates drifts with the increasing distance, which leads to large localization errors as shown in Table 5.2. Note also that RadarSLAM is a continuous localization technique, while RadarLoc is single-shot. For adapted Masking by Moving, adapted PoseNet17, adapted AtLoc and adapted LSTM, the results indicate that our proposed neural network architecture is superior than previously proposed architectures for both deep radar odometry and deep camera relocalization.

We also compare with camera-based and LiDAR-based deep relocalization methods to examine the differences among different sensory data for deep pose regression. The results in Table 5.2 demonstrate that radar-based deep relocalization method is much better than camera-based method in terms of accuracy. The probable reasons are radar sensors can provide broader FoV of scenes and are less sensitive to environmental conditions than

cameras. Interestingly, RadarLoc significantly outperforms PointLoc in translation while remaining comparable in rotation performance. This is most likely due to LiDAR providing full 3-D metric depth, rather than the 2-D bird's eye view of the FMCW radar scanner, which aids in full 6-DoF pose estimation.

Fig. 5.5 shows the cumulative distribution function (CDF) for both translation and orientation errors for the above mentioned approaches. RadarLoc consistently produces low errors in all sequences and it is closely followed by PointLoc. Pose accuracy for RadarSLAM and AtLoc, on the other hand, was highly dependent on the sequence being considered. Most noticeably, in Sequence 09, over 40% of poses estimated with RadarSLAM showed errors beyond 40m.

5.4.5 Ablation Study

In order to study the impact of different components of the proposed RadarLoc system, we conduct the ablation study as shown in Table 5.3. For ablation experiments, we keep all the architecture designs the same as RadarLoc except that we do not contain self-attention module (w/o SA), use the UNet as the self-attention module (SA w/ UNet), use the ResNet as the radar encoder (ResNet) and do not use the geometric constraints as one component of the loss function (w/o GC) respectively. The RadarLoc improves the w/o SA by 39.77% in translation and 5.70% in rotation, which proves that our self-attention module is very effective in improving the radar localization performance. To delve into the reasons behind the improvement of our self-attention module, we visualize the soft attention map as depicted in Fig. 5.2. The self-attention module helps RadarLoc focus more on the static objects like streets and buildings rather than feature-less regions of a radar image. Moreover, it improves the SA w/ UNet by 12.71% and the ResNet by 31.51% in translation whilst retaining comparable performance in rotation (less than 0.1°). Note that the performance of translation is very crucial in application scenarios like indoor parking lot or outdoor autonomous robots. Furthermore, the RadarLoc improves the w/o GC by 36.63% in translation and 5.39% in rotation, which demonstrates that the geometric constraints can greatly improve the performance of radar relocalization. Meanwhile, RadarLoc without geometric constraints (w/o GC) also outperforms all the baselines in Table 5.2 except the rotation of PointLoc, which reveals the effectiveness of the proposed neural network architecture considering the only difference between RadarLoc and the w/o GC is the loss function.

Table 5.3: Results showing the mean translation error (m) and rotation error (°) of ablation studies on the Oxford Radar RobotCar Dataset.

Sequence	w/o SA	SA w/ UNet	ResNet	w/o GC	RadarLoc
Seq-06	12.56m, 3.89°	9.96m, 3.62°	11.51m, 3.62°	11.13m, 3.80°	**8.43m**, **3.44**°
Seq-07	10.26m, 3.16°	6.74m, 2.76°	8.09m, **2.75**°	8.04m, 2.95 °	**5.12m**, 2.87°
Seq-08	10.91m, 3.38°	**6.46m**, 2.72°	9.42m, **2.60**°	10.74m, 3.47°	6.56m, 3.06°
Seq-09	10.36m, 2.82°	7.77m, 3.13°	9.73m, 2.86°	10.34m, **2.77**°	**6.51m**, 2.91°
Seq-10	8.94m, 1.65°	5.64m, **1.61**°	7.91m, 1.81°	10.15m, 1.88°	**5.34m**, 1.78°
Average	10.61m, 2.98°	7.32m, 2.77°	9.33m, **2.73**°	10.08m, 2.97°	**6.39m**, 2.81°

5.5 Summary

In this chapter, we first introduce the background and motivation of radar relocalization, and then define the deep radar relocalization task. We propose an end-to-end deep radar relocalization system RadarLoc. We also propose a loss function with geometric constraints and self-attention module to further improve the performance of relocalization. It outperforms state-of-the-art approaches based on comprehensive experiments. This work is inspired by PointLoc introduced in the previous section. Radar sensors can perceive environments in different conditions like fog, so we study how to utilize it for relocalization. From this work, we observe that it is important to choose the feature extraction neural network to improve the overall performance. We select ResNet and DenseNet respectively for our experiments, and find that the results are significantly different. Meanwhile, the neural network of removing outliers is also significant to enhance the localization accuracy. Radar images generated by FMCW radar sensors contain multiple outliers like moving objects, which can affect the prediction accuracy. Thus, we also employ self-attention mechanism, which the neural network learn to mask outliers by itself from training datasets. Different from PointLoc, RadarLoc learns a mask which has the same shape as the radar image. Furthermore, we find that the loss function can affect the relocalization performance. In our RadarLoc, we add the relative constraint between two radar scans to the regression loss function as a term, leading to the significant improvement of accuracy. It indicates that when dealing with the relocalization problem, we can always consider adding a relative constraint term in the loss function. In the next chapter, we will conclude the whole thesis.

Chapter 6

Conclusions

This thesis aims to study deep learning for localization with LiDAR point clouds and radar scans as sensory inputs. To achieve this goal, we propose three approaches to build localization systems respectively.

In Chapter 3, we propose a novel end-to-end deep parallel neural network, termed DeepPCO, to predict LiDAR odometry. In Chapter 4, a novel PointNet-style architecture with self-attention, called PointLoc, is developed to efficiently solve LiDAR relocalization from 360° LiDAR input frames. In Chapter 5, we introduce a novel end-to-end neural network with self-attention, named RadarLoc, to estimate 6-DoF global poses directly for radar data. Extensive experiments and ablation studies have been conducted on the KITTI, Oxford Radar Robotcar and vReLoc datasets to validate the effectiveness of the proposed approaches.

6.1 Key Contributions

We summarize three key contributions to LiDAR and Radar based localization with deep neural networks as follows.

In Chapter 3, we present an end-to-end deep parallel neural network named DeepPCO for the point cloud odometry task. To design deep neural networks to learn effective features from LiDAR point clouds, we explored several data encoding approaches. Our experiments indicate that the projected panoramic depth images can lead to the best performance for the LiDAR odometry task. Two consecutive point clouds are processed into panoramic depth images, which are stacked and sent simultaneously to dual sub-networks for transformation estimation. These two sub-networks are designed so that one sub-network is trained to estimate position, and the other one is trained to estimate orientation. We also employ a pre-trained FlowNet convolutional neural network to enhance the performance of orientation estimation. Our method improves the state-of-the-art Two-stream [39] method

by 52.53% in odometry prediction. We hope that our design choices and experiments on 2-D and 3-D encoding approaches and corresponding architectures provide an initial basis for and inspiration for future research in the area of robust point cloud-based odometry.

In Chapter 4, we present a novel LiDAR relocalization approach, PointLoc, based on deep learning. Conventional geometric LiDAR localization systems require the careful design of sub-systems and heavy hyper-parameter tuning procedures. Retrieval-based approaches consume time and storage, which restricts their applications in real-world scenarios. Therefore, we study the learning-based approach to regress global poses directly. Inspired by the deep camera relocalization methods, we study how to predict global poses from the raw 3-D LiDAR point clouds. However, how to effectively extract features for relocalization is challenging. We study several deep neural networks for point clouds. Leveraging a PointNet-style neural network, the proposed system achieves superior relocalization accuracy compared to state-of-the-art LiDAR point cloud and image-based relocalization approaches. We also need to consider the impact of outliers which can significantly affect the overall performance of relocalization. Thus, we introduce the self-attention mechanism into the neural networks to learn to remove outliers from data. PointLoc can predict global poses with limited calculation time and small storage space compared to existing LiDAR localization systems. It can therefore be applied to large-scale relocalization and robot navigation scenarios for satisfying meter-level localization requirements. It can also be leveraged in indoor environments or urban areas with high-rise buildings as a complement when GNSS-based location is absent. For centimeter-level requirement scenarios like autonomous driving, it can serve as a coarse localization method to provide good initial pose estimates where point cloud registration and RANSAC can be applied to further improve the accuracy of relocalization. In conclusion, PointLoc is the first LiDAR point cloud-based method for LiDAR relocalization in an end-to-end manner. We validate our approach by conducting real-world robot experiments. We also release an indoor dataset for LiDAR and image related localization tasks. PointLoc outperforms the state-of-the-art DNN-based LiDAR point cloud and camera relocalization approaches.

In Chapter 5, we propose a novel radar-based relocalization system, RadarLoc, based on deep learning. Radar sensors can mitigate the shortcomings of camera and LiDAR sensors since they can function well in unstructured environments in different conditions such as fog. The raw FMCW radar scan is in the type of polar representation. To enable better learning of features from radar scans, we transform a raw scan into a Cartesian image. In this way, we can utilize existing neural network architectures from camera images to extract features. However, radar images contain multiple outliers such as moving objects,

which we need to remove. Inspired by the deep radar odometry works, we design self-attention with a nested encoder-decoder architecture to mask out outliers. Furthermore, we adopt DenseNet instead of ResNet as feature encoder for radar images, and the experiments demontrate that DenseNet is better than ResNet for our relocalization task. To further improve the performance of radar relocalization, we also design a novel loss function to leverage the geometric relationship between radar images. The proposed RadarLoc can directly predict 6-DoF global poses in an end-to-end fashion. Experiments on city-scale open datasets demonstrate the feasibility of employing radar sensors for the relocalization task. The system can be leveraged in urban areas like Oxford for localization or as a component of the existing radar localization system to redeem the accumulative drifts of radar odometry. In conclusion, we design an end-to-end learning-based radar relocalization system. We propose a geometric loss function and a self-attention neural network module to further enhance the relocalization performance. RadarLoc outperforms the geometry-based RadarSLAM and deep camera and LiDAR relocalization methods on the challenging Oxford Radar RobotCar dataset.

6.2 Discussion

This work studies how to exploit deep learning for the LiDAR and radar based localization task. We discuss implications from three aspects in this thesis: encoding, sensor types and scalability.

- **Encoding.** In DeepPCO, we projected raw 3-D point clouds into depth images for the LiDAR odometry estimation task considering the limitations of the computational complexity and performance. On one hand, this encoding approach may lose spatial information which is significant for pose predictions. Thus, when computational resources are enough, we can study the voxelization of point clouds for LiDAR odometry. On the other hand, we conducted the PointNet-like neural network for raw 3-D point clouds, but the performance of it was worse than the approach of projected depth images. With the development of 3-D neural networks for point clouds, we can study novel neural networks to directly learn features from raw 3-D point clouds for LiDAR odometry. In PointLoc, we adopted PointNet-like neural network architecture to estimate global poses from raw 3-D point clouds. We also tried to project point clouds into depth images, but the projected depth images are too sparse which leads to poor performance. Therefore, the projected encoding approach of point clouds is not suitable for sparse point clouds. If there are a large number of points, we can employ a projected encoding approach to perform global relocalization experiments.

- **Sensor Types.** Different sensor types can lead to different approaches for localization. The number of points is varying according to different types of sensors. For DeepPCO, the type of LiDAR sensor is Velodyne-64, which can generate around 120,000 points. Therefore, it is difficult to adopt voxelization approaches for localization because the number of points is large which can consume significant computational resources, and the neural networks will be hard to train and converge. For PointLoc, the type of LiDAR sensor is Velodyne-32, which can produce around 20,000 points. If we project them into depth images, they will be too sparse, and the performance will be degraded since the lack of enough depth information. For RadarLoc, we utilized the FMCW radar sensor to study, which generates polar images. Several radar sensors can also generate point clouds, which can be further studied in the field of radar relocalization. Meanwhile, in our experiments, RadarLoc outperforms PointLoc in translation and has comparable performance in orientation. However, the LiDAR sensor type in the experiments is Velodyne-32. If the type of the LiDAR sensor is changed, the results could be different. Hence, the sensor types of LiDAR and radar play an important role in the performance of localization. Comprehensive datasets of different types of LiDAR and radar sensors are needed to compare and study for localization.

- **Scalability.** It is important to discuss the scalability of the proposed methods. Learning-based localization approaches heavily rely on the collected datasets. For DeepPCO, PointLoc, and RadarLoc, all of them can be applied on a large scale. However, they have to be re-trained to have an excellent localization performance. For LiDAR odometry, the scenes in the test datasets can be different from the training datasets. Thus, DeepPCO can be trained in small datasets and extended to a large scale. However, two problems should be further studied. One is whether the sensor types and settings should be the same for all datasets. The other is how to determine the size of the training datasets if we want to have a good localization performance on a large scale. For PointLoc and RadarLoc, it is required that agents have visited the scenes before. Thus, to be scalable, they should be re-trained on the whole environment. We also note that for the relocalization problem in the real world, if the scenes become large, it may require coarse localization as initial poses first. For example, if we want to perform relocalization in the UK, we apply multiple PointLoc or RadarLoc to datasets collected from every city in the UK and group them. When we want to relocalize an agent, we first employ GPS or retrieval-based methods to generate an initial pose. Next, we send the current point cloud or radar scan of the query agent

to the corresponding PointLoc or RadarLoc for accurate global pose estimations. To conclude, the proposed LiDAR and radar based deep localization approaches can be scalable.

6.3 Future Work

The work in this thesis opens up a number of research directions for future work, which can potentially improve the localization performance of LiDAR and radar based localization systems.

For DeepPCO, several interesting extensions could be considered from our work. It would be interesting to consider how to integrate our system into a full learning-based SLAM system and evaluate its performance. Furthermore, it is worth evaluating our approach in more challenging environments like severe weather conditions, and leverage more datasets to improve the accuracy of odometry estimation. Another direction is to expand on learning-based sensor fusion methods for LiDAR odometry. One failure case is that we tested VoxelNet for raw point clouds, and it was difficult to train. Thus, one research direction is how to design a deep neural network and an encoding approach that can facilitate the training speed, reduce the computational complexity and keep the 3-D information from point clouds. Future work could also focus on extending the proposed architecture to leverage geometry-based optimizations, in order to further improve the overall performance.

For PointLoc, future research is required to improve the relocalization accuracy by more effectively eliminating the noisy point features from the point cloud, since outliers such as moving objects can significantly affect the performance of LiDAR relocalization. We can also explore how to utilize the intensity information of LiDAR sensors for better relocalization performance. One failure case is that we projected point clouds into depth images and perform global localization. However, the projected depth images are too sparse and the performance is not good enough. One research direction is to collect more datasets using LiDAR sensors with more dense point clouds, and we project point clouds into depth images and study how to design deep neural networks for relocalization with depth images. We can also research whether deep camera relocalization methods can be applied to these depth images for excellent performance. Moreover, how to leverage multiple point clouds to jointly improve the localization accuracy can also be a promising direction to consider.

For RadarLoc, one important direction for future work is on reducing the prediction outliers, which significantly influences the performance of the large-scale localization. Another direction is to integrate the deep radar relocalization system with deep radar odometry.

One failure case is that we extract features from LiDAR point clouds and radar scans, and fuse them in the late stage for relocalization. However, the relocalization performance of the fusion approach is worse than the LiDAR or radar relocalization only. Several existing research tasks have demonstrated that sensor fusion can improve the overall performance. Consequently, one research direction is how to fuse LiDAR and radar features to improve the relocalization accuracy. We can also consider fusing other sensor modalities like IMU to improve the overall performance. In the future, collecting more radar sensory data to supplement the shortage of open datasets is also significant. Moreover, different neural architectures can be designed and validated on these datasets to achieve more accurate localization results.

6.4 Conclusions

In conclusion, this thesis studies the LiDAR and radar based localization with deep neural networks. We first review the advantages and drawbacks of existing LiDAR and radar based localization systems, and then introduce the deep learning-based odometry estimation and relocalization methods with camera, LiDAR and radar sensors. Based on the downsides of existing localization systems, we propose three approaches for LiDAR and radar based localization. DeepPCO utilizes projected depth images from LiDAR point clouds to predict poses, which is a deep parallel neural network. PointLoc exploits 3-D point clouds to estimate global poses, which is a PointNet-style neural network. RadarLoc employs transfer learning and self-attention mechanisms for radar relocalization, and is based on DenseNet and UNet. The proposed three methods reveal the potential to solve localization problems with deep neural networks. The end-to-end learning-based localization systems address some of the shortcomings of conventional geometry-based localization approaches, demonstrating promising results and opening up several directions for future work. We believe that this thesis can contribute to the development and research of robust and accurate localization systems, which can be applied to a variety of emerging application scenarios in indoor and outdoor settings.

Bibliography

[1] Dan Barnes, Matthew Gadd, Paul Murcutt, Paul Newman, and Ingmar Posner. The Oxford Radar RobotCar Dataset: A Radar Extension to the Oxford RobotCar Dataset. *ICRA*, pages 6433–6438, 2020.

[2] Dan Barnes and Ingmar Posner. Under the Radar: Learning to Predict Robust Keypoints for Odometry Estimation and Metric Localisation in Radar. *ICRA*, 2020.

[3] Dan Barnes, Rob Weston, and Ingmar Posner. Masking by Moving : Learning Distraction-Free Radar Odometry from Pose Information. *CoRL*, 2019.

[4] Ioan Andrei Barsan, Shenlong Wang, Andrei Pokrovsky, and Raquel Urtasun. Learning to Localize Using a LiDAR Intensity Map. *CoRL*, pages 605–616, 2018.

[5] Paul J Besl and D. Nerl McKay. A Method for Registration of 3-D Shapes. *TPAMI*, 1992.

[6] Samarth Brahmbhatt, Jinwei Gu, Kihwan Kim, James Hays, and Jan Kautz. Geometry-Aware Learning of Maps for Camera Localization. *CVPR*, pages 2616–2625, 2018.

[7] Cesar Cadena, Luca Carlone, Carrillo Henry, Latif Yasir, Scaramuzza Davide, Neira Jose, Reid Ian, and J. Leonard John. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics*, 2016.

[8] Sarah H. Cen and Paul Newman. Precise Ego-Motion Estimation with Millimeter-Wave Radar under Diverse and Challenging Conditions. *ICRA*, 2018.

[9] Changhao Chen, Bing Wang, Chris Xiaoxuan Lu, Niki Trigoni, and Andrew Markham. A Survey on Deep Learning for Localization and Mapping: Towards the Age of Spatial Machine Intelligence. *arXiv*, 2020.

[10] Ronald Clark, Sen Wang, Andrew Markham, Niki Trigoni, and Hongkai Wen. VidLoc: A Deep Spatio-Temporal Model for 6-DoF Video-Clip Relocalization. *CVPR*, pages 2652–2660, 2017.

[11] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. VINet: Visual-Inertial Odometry as a Sequence-to-Sequence Learning Problem. *AAAI*, 31(1), 2017.

[12] Gabriele Costante and Thomas A. Ciarfuglia. LS-VO: Learning Dense Optical Subspace for Robust Visual Odometry Estimation. *ICRA*, 2018.

[13] Renaud Dubé, Andrei Cramariuc, Daniel Dugas, Juan Nieto, Roland Siegwart, and Cesar Cadena. SegMap: 3D Segment Mapping using Data-Driven Descriptors. *RSS*, 2018.

[14] Philipp Fischer, Eddy Ilg, H Philip, Caner Hazırbas, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning Optical Flow with Convolutional Networks. *ICCV*, 2015.

[15] Matthew Gadd, Daniele De Martini, and Paul Newman. Contrastive Learning for Unsupervised Radar Place Recognition. *ICAR*, 2021.

[16] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. *CVPR*, 2012.

[17] Ruben Gomez-Ojeda, Zichao Zhang, Javier Gonzalez-Jimenez, and Davide Scaramuzza. Learning-based Image Enhancement for Visual Odometry in Challenging HDR Environments. *ICRA*, 2018.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CVPR*, 2016.

[19] Ziyang Hong, Yvan Petillot, and Sen Wang. RadarSLAM: Radar based Large-Scale SLAM in All Weathers. *IROS*, 2020.

[20] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. *CVPR*, 2017.

[21] Zhaoyang Huang, Yan Xu, Jianping Shi, and Xiaowei Zhou. Prior Guided Dropout for Robust Visual Localization in Dynamic Environments. *ICCV*, pages 2791–2800, 2019.

[22] Alex Kendall and Roberto Cipolla. Modelling Uncertainty in Deep Learning for Camera Relocalization. *ICRA*, pages 4762–4769, 2016.

[23] Alex Kendall and Roberto Cipolla. Geometric Loss Functions for Camera Pose Regression with Deep Learning. *CVPR*, pages 6555–6564, 2017.

[24] Alex Kendall, Matthew Grimes, and Robert Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. *ICCV*, pages 2938–2946, 2015.

[25] Hyungjin Kim, Bingbing Liu, Chi Yuan Goh, Serin Lee, and Hyun 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*, 2(3):1518–1524, 2017.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *ICLR*, 2015.

[27] Truc Le and Ye Duan. PointGrid: A Deep Network for 3D Shape Understanding. *CVPR*, pages 9204–9214, 2018.

[28] Jesse Levinson and Sebastian Thrun. Robust Vehicle Localization in Urban Environments Using Probabilistic Maps. *ICRA*, pages 4372–4378, 2010.

[29] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle Detection from 3D Lidar Using Fully Convolutional Network. *RSS 2016*.

[30] Ruihao Li, Sen Wang, Zhiqiang Long, and Dongbing Gu. UnDeepVO: Monocular Visual Odometry through Unsupervised Deep Learning. *ICRA*, 2018.

[31] Weiping Liu, Jia Sun, Wanyi Li, Ting Hu, and Peng Wang. Deep Learning on Point Clouds and Its Application: A Survey. *Sensors*, 19(19), 2019.

[32] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. FlowNet3D : Learning Scene Flow in 3D Point Clouds. *CVPR*, pages 529–537, 2019.

[33] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. DeepVCP: An End-to-End Deep Neural Network for Point Cloud Registration. *ICCV*, pages 12–21, 2019.

[34] Weixin Lu, Yao Zhou, Guowei Wan, Shenhua Hou, and Shiyu Song. L3-Net : Towards Learning based LiDAR Localization for Autonomous Driving. *CVPR*, pages 6382–6391, 2019.

[35] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *IJRR*, 36(1):3–15, 2016.

[36] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. *IROS*, 2015.

[37] Iaroslav Melekhov, Juha Ylioinas, Juho Kannala, and Esa Rahtu. Image-Based Localization Using Hourglass Networks. *ICCV-W*, pages 870–877, 2017.

[38] Peter Muller and Andreas Savakis. Flowdometry: An Optical Flow and Deep Learning Based Approach to Visual Odometry. *WACV*, 2017.

[39] Austin Nicolai, Ryan Skeele, Christopher Eriksen, and Geoffrey A Hollinger. Deep Learning for Laser Based Odometry Estimation. *RSS-W*, 2016.

[40] Charles R. Qi, Or Litany, Kaiming He, and Leonidas J. Guibas. Deep Hough Voting for 3D Object Detection in Point Clouds. *ICCV*, pages 9276–9285, 2019.

[41] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *CVPR*, pages 77–85, 2017.

[42] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *NeurIPS*, pages 5099–5108, 2017.

[43] Stefan Saftescu, Matthew Gadd, Daniele De Martini, Dan Barnes, and Paul Newman. Kidnapped Radar: Topological Radar Localisation using Rotationally-Invariant Metric Learning. *ICRA*, 2020.

[44] Muhamad Risqi U Saputra, Andrew Markham, and Niki Trigoni. Visual SLAM and Structure from Motion in Dynamic Environments: A Survey. *CSUR*, 2018.

[45] Torsten Sattler, Qunjie Zhou, Marc Pollefeys, and Laura Leal-taix. Understanding the Limitations of CNN-based Absolute Camera Pose Regression. *CVPR*, pages 3297–3307, 2019.

[46] Aleksandr V Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-ICP. *RSS*, 2009.

[47] Tim Yuqing Tang, Daniele De Martini, Dan Barnes, and Paul Newman. RSL-Net: Localising in Satellite Images from a Radar on the Ground. *RA-L*, 5(2):1087–1094, 2020.

[48] Mikaela Angelina Uy and Gim Hee Lee. PointNetVLAD: Deep Point Cloud Based Retrieval for Large-Scale Place Recognition. *CVPR*, pages 4470–4479, 2018.

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

[50] Bing Wang, Changhao Chen, Chris Xiaoxuan Lu, Peijun Zhao, Niki Trigoni, and Andrew Markham. AtLoc: Attention Guided Camera Localization. *AAAI*, 34(06):10393–10401, 2020.

[51] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. DeepVO: Towards End-to-End Visual Odometry with Deep Recurrent Convolutional Neural Networks. *ICRA*, pages 2043–2050, 2017.

[52] Wei Wang, Pedro P B De Gusm, Bo Yang, Andrew Markham, and Niki Trigoni. RadarLoc : Learning to Relocalize in FMCW Radar. *ICRA*, 2021.

[53] Wei Wang, Muhamad Risqi U. Saputra, Peijun Zhao, Pedro Gusmao, Bo Yang, Changhao Chen, Andrew Markham, and Niki Trigoni. DeepPCO: End-to-End Point Cloud Odometry through Deep Parallel Neural Network. *IROS*, 2019.

[54] Wei Wang, Bing Wang, Peijun Zhao, Changhao Chen, Ronald Clark, Bo Yang, Andrew Markham, and Niki Trigoni. PointLoc: Deep Pose Regressor for LiDAR Point Cloud Localization. *IEEE Sensors Journal*, 22(1):959 – 968, 2022.

[55] Yue Wang and Justin Solomon. Deep Closest Point: Learning Representations for Point Cloud Registration. *ICCV*, pages 3522–3531, 2019.

[56] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic Graph CNN for Learning on Point Clouds. *ACM TOG*, 38(5):1–12, 2019.

[57] Xinkai Wei, Ioan Andrei Barsan, Shenlong Wang, Julieta Martinez, and Raquel Urtasun. Learning to Localize Through Compressed Binary Maps. *CVPR*, pages 10308–10316, 2019.

[58] Bo Yang, Sen Wang, Andrew Markham, and Niki Trigoni. Robust Attentional Aggregation of Deep Feature Sets for Multi-view 3D Reconstruction. *IJCV*, pages 53–73, 2019.

[59] Nan Yang, Rui Wang, Jörg Stückler, and Daniel Cremers. Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry. *ECCV*, 2018.

[60] Zi Jian Yew and Gim Hee Lee. 3DFeat-net: Weakly Supervised Local 3D Features for Point Cloud Registration. *ECCV*, pages 607–623, 2018.

[61] Huan Yin, Li Tang, Xiaqing DIng, Yue Wang, and Rong Xiong. LocNet: Global Localization in 3D Point Clouds for Mobile Vehicles. *IV*, pages 728–733, 2018.

[62] Ji Zhang and Sanjiv Singh. LOAM: Lidar Odometry and Mapping in Real-time. *RSS*, 2014.

[63] Ji Zhang and Sanjiv Singh. Visual-lidar Odometry and Mapping: Low-rift, Robust, and Fast Localization. *ICRA*, pages 393–398, 2015.

[64] Wenxiao Zhang and Chunxia Xiao. PCAN: 3D Attention Map Learning Using Contextual Information for Point Cloud Based Retrieval. *CVPR*, pages 12428–12437, 2019.

[65] Tianhang Zheng, Changyou Chen, Junsong Yuan, Bo Li, and Kui Ren. PointCloud Saliency Maps. *ICCV*, pages 1598–1606, 2019.

[66] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. *CVPR*, pages 4490–4499, 2018.

[67] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. UNet++: A Nested U-Net Architecture for Medical Image Segmentation. *DLMIA-W*, 2018.