LodoNet: A Deep Neural Network with 2D Keypoint Matching for 3D LiDAR Odometry Estimation

Ce Zheng
czheng6@uncc.edu
The University of North Carolina at Charlotte

Yecheng Lyu
ylyu@wpi.edu
Worcester Polytechnic Institute

Ming Li
mli12@wpi.edu
Worcester Polytechnic Institute

Ziming Zhang*
zzhang15@wpi.edu
Worcester Polytechnic Institute

ABSTRACT

Deep learning based LiDAR odometry (LO) estimation attracts increasing research interests in the field of autonomous driving and robotics. Existing works feed consecutive LiDAR frames into neural networks as point clouds and match pairs in the learned feature space. In contrast, motivated by the success of image based feature extractors, we propose to transfer the LiDAR frames to image space and reformulate the problem as image feature extraction. With the help of scale-invariant feature transform (SIFT) for feature extraction, we are able to generate matched keypoint pairs (MKPs) that can be precisely returned to the 3D space. A convolutional neural network pipeline is designed for LiDAR odometry estimation by extracted MKPs. The proposed scheme, namely LodoNet, is then evaluated in the KITTI odometry estimation benchmark, achieving on par with or even better results than the state-of-the-art.

KEYWORDS

Vehicle localization; LiDAR processing; Deep learning

ACM Reference Format:
Ce Zheng, Yecheng Lyu, Ming Li, and Ziming Zhang. 2020. LodoNet: A Deep Neural Network with 2D Keypoint Matching for 3D LiDAR Odometry Estimation. In *28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3413771

1 INTRODUCTION

Odometry estimation is one of the key components in the automated driving systems and robotics. In recent years, autonomous driving has attracted significant research interests and several works have been proposed to estimate the position and orientation of the autonomous vehicles. Multiple sensors are deployed to collect relative data including monocular cameras, inertial measurement units (IMU), and light detection and ranging (LiDAR). Camera is first selected for odometry estimation because it is cost efficient and there exist well-studied image feature extractors to support. However, camera based methods such as VINS-Mono[22], DVSO[35], and GANVO[1] highly subject to the accuracy of camera calibration, and it is difficult to transfer from image feature pairs to coordinate correspondents between frames. Comparing with cameras, LiDARs take advantage of accurate distance acquisition and insensitive to the light condition. Therefore, LiDAR odometry (LO) attracts increasing research interests. Several works have been proposed to solve the vehicle odometry estimation using LiDAR data. They mainly follow two approaches. The first one performs traditional set-to-set registration that extracts features from the entire point cloud and register one to another. This approach includes LOAM[38] and ICP[6][26][13]. However, those approach only focus on the global features without capturing the local pattern, which limits its accuracy. The other approach use neural networks to extract local and global features and try to estimate the vehicle odometry through matching the feature vectors. However, Estimating odometry directly from large amount of 3D points is not only challenging but also computationally expensive. Motivated by the success of image feature extractors, we intuitively raise the question:

Can we extract features from the LiDAR point clouds using image feature extractors so that the coordinate correspondences can be established through matching points in image feature space?

Fortunately, Lyu *et al.* [17, 18] and RangeNet++ [19] have introduced an algorithm to project the LiDAR data on to a spherical view so that a LiDAR point cloud with geometry features can be transferred to an image-like feature map with minor point losses. By employing this projection, we can efficiently generate image representations of LiDAR frames for feature extraction.

In this paper, we propose a deep neural network architecture to estimate the vehicle odometry from LiDAR frames. we first project the sparse LiDAR point clouds to spherical depth images with depth completion to tackle the sparse issue. We than apply the classic keypoints detection and matching algorithm to the 2D spherical depth images. The extracted matched keypoint pairs(MKPs) on 2D spherical image space can be projected back to the 3D LiDAR space by the inverse projection function. Figure 1 illustrates that the MKPs extracted by our method are accurate and reliable on 3D LiDAR space. Inspired by the recent works on deep learning-based Visual Odometry (VO) estimation and the promising performance of PointNet[21] on point cloud segmentation and classification, we

*Corresponding author

MM '20, October 12–16, 2020, Seattle, WA, USA.

ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413771

construct our deep neural network architecture to estimate LiDAR odometry using extracted MKPs as illustrated in Figure 2.

To summarize, our main contributions are:

- We propose a new approach for extracting matched keypoint pairs(MKPs) of consecutive LiDAR scans by projecting 3D point cloud onto 2D spherical depth images where MKPs can be extracted effectively and efficiently. After projecting back to 3D LiDAR space, the extracted MKPs can be used for LiDAR odometry estimation.
- By utilizing the PointNet structure, which provide strong performance over point cloud tasks, We adopt our convolutional neural network architecture to infer the rotation information and translation information from extracted MKPs of consecutive scans.
- The evaluation of our experiments and ablation studies on KITTI Odometry estimation benchmark[9] demonstrate the effectiveness of the proposed method.

2 RELATED WORK

LiDAR feature extraction in 2D space.

LiDAR feature extraction is one of the essential tasks of LiDAR-based applications. Since the raw data from the LiDAR sensors are not well structuralized and ordered, an intuitive way is to project the LiDAR points onto a 2D space. LoDNN [5] is a pioneering work that projects all LiDAR points onto the ground plane and samples them to an image-like array. In this work, however, the LiDAR points are not uniformly distributed on the ground plane but heavily gathered together near the LiDAR scanner, which results in massive dropped points in the near-range and redundant space in the far-range. Lyu *et al.* [17, 18] and RangeNet++ [19] improve the projection scheme by replacing the target plane with a sphere surface, in which LiDAR points are nearly uniformly distributed. SqueezeSeg V1 [33], V2 [34], and LO-Net [12] also employ this projection scheme and result in a good performance in LiDAR point semantic segmentation. In our work, we follow the projection scheme of RangeNet++ to generate a 2D LiDAR feature map for each LiDAR frame.

Image feature extraction and keypoint detection.

Image feature extraction and keypoint detection have been studied for decades. Scale-invariant feature transform (SIFT) [14] is one of the popular feature extraction algorithms in the image processing domain. By transforming an image into a large group of feature vectors that are invariant to image translation, scaling, and rotation, SIFT generates robust feature vectors on parts captured in different views. Other feature extraction methods include SURF [2] and ORB [24], however, they cannot generate robust feature vectors on LiDAR frames. In our work, we employ the SIFT as our LiDAR feature extractor.

Registration and feature-based vehicle odometry estimation.

Iterative Closet Point (ICP)[3] method and its variants[20][7][23] have been used in the field of LiDAR based pose estimation widely. In the ICP algorithm, a transformation between point clouds of adjacent scans is optimized iteratively by minimizing distance until a specific termination condition is met. Despite its popularity, ICP is sensitive to the initial poses and computationally expensive. Multiple variants of ICP algorithms such as point-to-line ICP[6], point-to-plane ICP[13], and plane-to-pane ICP[20] were developed. GICP[26] was proposed by combining point-to-point ICP and point-to-plane ICP into a single probabilistic framework. A probabilistic model is attached to the minimization step which can reduce the time complexity and increase the robustness to incorrect correspondences. Collar Line Segments (CLS)[29] transforms the LiDAR scans into line could then generates line segments within each bin. This pre-processing method produces better results than GICP. However, due to the high computational cost in line segments, CLS cannot achieve real-time odometry estimation.

The state-of-art LiDAR Odometry estimation method: LiDAR Odometry And Mapping(LOAM)[38] was proposed by Zhang and Singh, which achieves both low-drift in motion estimation and low-computational complexity. The key idea is to divide the complex problem of Simultaneous Localization and Mapping into two parallel algorithms. One algorithm performs odometry at a high frequency but at low fidelity to estimate the velocity of the laser scanner. A second algorithm runs at an order of magnitude lower frequency for fine matching and registration of the point cloud.

Deep Learning-based vehicle odometry estimation.

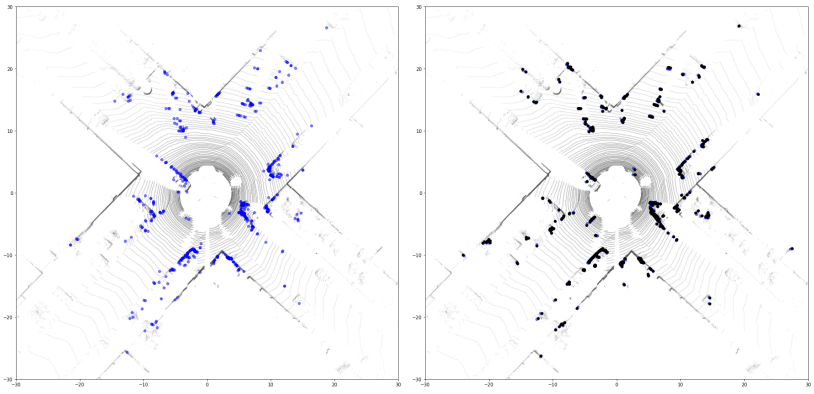
In recent years, several works have been done exploring the use of neural networks in vehicle odometry estimation. DeepVO[31], DVSO[35], Depth-VO-Feat[37], and GANVO[1] have achieved promising results on Visual odometry(VO) estimation. In VO tasks, Camera data are used. However, applying deep learning method to solve 3D LiDAR odometry problem still remains challenges. DeepICP[15] detects the keypoints by a point weighting layer, and then generates a search region for each keypoint. A matched point can be generated by the corresponding point generation layer. The odometry is finally estimated by solving the SVD from the matched keypoint pairs. DeepPCO[32] generates panoramic-view of depth image projection to feed to it neural networks. L3-Net [16] proposes a learning-based LiDAR localization system by comparing the network-based feature vector between the current LiDAR frame and pre-build point cloud map followed by a recurrent neural network based smoothness module. LO-Net[12] is another learning-based odometry estimator. Different from L3-Net that only extracts point-wise features, LO-Net projects the points onto a sphere surface and builds an image-like feature map for each LiDAR frame. For better training the odometry estimator, LO-Net introduces an attention branch to predict if the geometric consistency of an area in the feature map can be modeled or not.

3 METHOD

3.1 Problem Setup

We are given a collection of training samples $\{(\mathcal{X}_t, \mathcal{X}_{t+1}, \mathbf{y}_t)\}_{t\in\mathcal{T}}$ where $\mathcal{X}_t \subseteq \mathbb{R}^3$ denotes the set of keypoints from the t-th LiDAR scans, $\mathbf{y}_t \in \mathcal{Y} \subseteq \mathbb{R}^6$ denotes the ground-truth odometry between the t-th and $(t+1)$-th LiDAR scans, and $\mathcal{T}$ denotes the scan index set. Our goal is to learn an odometry prediction function $f : \mathcal{X} \times \mathcal{X} \rightarrow \mathcal{Y} \in \mathcal{F}$ by minimizing certain loss function $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$, *i.e.,*

$$\min_{f\in\mathcal{F}} \sum_{t\in\mathcal{T}} \ell(f(\mathcal{X}_t, \mathcal{X}_{t+1}), \mathbf{y}_t). \tag{1}$$



(a) (b)

Figure 1: **Illustration of MKPs detected by our methods. (a): The point cloud of i+1's LiDAR scan, blue points are the keypoints detected by SIFT on the coordinate frame of $i+1$'s scan. (b): Black points are the MKPs of the blue points from the $i's$ scan and projected to the $i+1's$ coordinate frame. The black points are expected to overlap to their paired blue points since they are match points projected to the same coordinate frame. Lines are aligned for each matched pair (blue point and paired black point). Here these lines are invisible because of overlapping.**

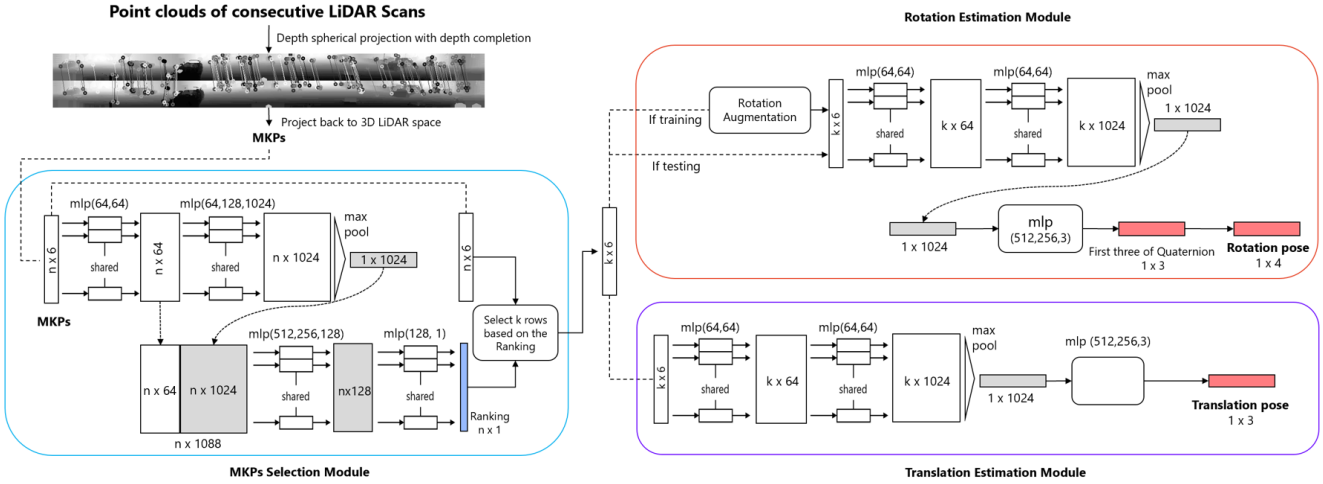


Figure 2: **LodoNet Architecture:Depth spherical images of the LiDAR point clouds can be obtained by the projection procedure. The matching keypoint pairs (MKPs) is extracted by the feature extraction algorithm:SIFT on the depth images than projected back to 3D LiDAR space. The MKPs Selection Module takes n MKPs of consecutive scans as input. It can generate a Ranking Matrix $[n, 1]$ of the input n MKPs, then we choose the best k MKPs by their ranking as the input for rotation and translation estimation. The Rotation Estimation Module and Translation Estimation Module predict the rotation pose and translation pose respectively.Dotted line indicate two blocks are identical.**

Note that Eqn. 1 holds in general for all the LiDAR odometry estimation algorithms. To simplify our explanation later, here we assume that the feasible space $\mathcal{F}$ is proper, closed, and convex (PCC) that covers all the constraints on f such as regularization. At test time, given two sets of keypoints $\mathcal{X}_{t'}, \mathcal{X}_{t'+1}$, we can predict their odometry as $f(\mathcal{X}_{t'}, \mathcal{X}_{t'+1})$.

3.2 Formulation

As we know, odometry has 6 degrees of freedom (DOF). This leads to the fact that odometry can be estimated given at least 3 matched keypoint pairs (MKPs). Therefore, instead of learning the general function f in Eqn. 1 directly, in the literature it will be more plausible to decompose it as two functions, *i.e.*, $f = h \circ g$ where $\circ$ denotes the function composition. The keypoint matching function, $g : \mathcal{X}_t \times \mathcal{X}_{t-1} \rightarrow \mathcal{P}_t \times \mathcal{P}_{t-1} \subseteq \mathbb{R}^3 \times \mathbb{R}^3$, generates MKPs from the input keypoint sets, and the odometry regression function, $h : \mathcal{P}_t \times \mathcal{P}_{t-1} \rightarrow \mathcal{Y}$, predicts the odometry based on the MKPs. Odometry estimation algorithms are all about how to design or learn such functions g, h to determine function f.

ICP-based learning approaches. Iterative Closest Point (ICP) [4] matches two sets of points iteratively as well as estimating the pose transformation by minimizing distances between the corresponding matched points until it converges. Although ICP is well-known for point registration, the high computation and sensitivity to initial poses significantly limit its applications. The key idea behind ICP-based learning approaches for LiDAR odometry estimation is to use ICP to locate MKPs (given the current odometry regression function

h) that are used to update h further. In general, such approaches can be formulated as follows:

$$\min_{h\in\mathcal{H}} \sum_{t\in\mathcal{T}} \ell(h(\mathcal{P}_t, \mathcal{P}_{t+1}), \mathbf{y}_t), s.t.\ \mathcal{P}_t, \mathcal{P}_{t+1} = \tilde{g}(\mathcal{X}_t, \mathcal{X}_{t+1}, h(\mathcal{P}_t, \mathcal{P}_{t+1})) \tag{2}$$

where $\tilde{g} : \mathcal{X} \times \mathcal{X} \times \mathcal{Y} \to \mathcal{P} \times \mathcal{P}$ denotes a variant of the keypoint matching function, and the feasible space $\mathcal{H}$ is PCC.

The constraint here models the MKPs as the stationary solution given current h, which can be viewed as a generalization of ICP. Then such solutions are used to update h in the objective by minimizing the loss. At training time this procedure is repeated until it converges. At test time, given $\tilde{g}$ and learned h we use the constraint to locate the MKPs and then output the odometry estimation as $h(\mathcal{P}_{t'}, \mathcal{P}_{t'+1})$.

$$\mathbf{Q}_i = \begin{bmatrix} x_{i,1}^2 + x_{i,2}^2 & -x_{i,2}x_{i,3} & -x_{i,1}x_{i,3} & x_{i,2} & -x_{i,1} & 0 \\ -x_{i,2}x_{i,3} & x_{i,1}^2 + x_{i,3}^2 & -x_{i,1}x_{i,2} & -x_{i,3} & 0 & x_{i,1} \\ -x_{i,1}x_{i,3} & -x_{i,1}x_{i,2} & x_{i,2}^2 + x_{i,3}^2 & 0 & x_{i,3} & -x_{i,2} \\ x_{i,2} & -x_{i,3} & 0 & 1 & 0 & 0 \\ -x_{i,1} & 0 & x_{i,3} & 0 & 1 & 0 \\ 0 & x_{i,1} & -x_{i,2} & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$\mathbf{q}_i = \begin{bmatrix} y_{i,1}x_{i,2} - y_{i,2}x_{i,1} \\ -y_{i,1}x_{i,3} + y_{i,3}x_{i,1} \\ y_{i,2}x_{i,3} - y_{i,3}x_{i,2} \\ x_{i,1} - y_{i,1} \\ x_{i,2} - y_{i,2} \\ x_{i,3} - y_{i,3} \end{bmatrix} \tag{4}$$

$$\left[\alpha, \beta, \gamma, b_1, b_2, b_3\right]^\top = -\left[\sum_i \mathbf{Q}_i\right]^{-1} \sum_i \mathbf{q}_i \tag{5}$$

Our LodoNet. As we see in Eqn. 2, the odometry regression function h and the keypoint matching function g in ICP-based learning approaches are essentially *coupled*. This potentially can lead to two serious problems, at least, in training, *i.e.,* high computation and non-convergence of the training loss.

To address such problems, our methodology in LodoNet is to decouple functions g, h to avoid the loop as well as significantly improve the convergence. To this end, we propose the following optimization problem:

$$\min_{z\in\mathcal{Z}, \tilde{h}\in\tilde{\mathcal{H}}} \sum_{t\in\mathcal{T}} \ell(\tilde{h}(\mathcal{P}_t, \mathcal{P}_{t+1}, z(\mathcal{P}_t, \mathcal{P}_{t+1})), \mathbf{y}_t), \\ s.t.\ \mathcal{P}_t, \mathcal{P}_{t+1} = g(\mathcal{X}_t, \mathcal{X}_{t+1}), \tag{6}$$

where $z : \mathcal{P} \times \mathcal{P} \to \Pi$ denotes an *attentional* function that returns probability vectors over the MKPs in the simplex space Π, $\tilde{h} : \mathcal{P} \times \mathcal{P} \times \Pi \to \mathcal{Y}$ denotes a variant of the odometry regression function, and both spaces $\mathcal{Z}, \tilde{H}$ are PCC.

Different from ICP-based learning approaches, here we use a predefined keypoint matching function g to extract MKPs from LiDAR data (*i.e.,* constraint), and feed these MKPs as input to the learning algorithm directly to minimize the objective. In this way, there is no loop between g and $\tilde{h}$ (*i.e.,* decoupling). The quality of the MKPs, however, cannot be guaranteed to be good for odometry estimation. Therefore, we deliberately introduce the attentional mechanism to assign weights for estimation. In fact, Eqn. 6 is an unconstrained minimization problem with convergence guarantee using alternating optimization (*i.e.,* learning z while fixing $\tilde{h}$, and then learning $\tilde{h}$ while fixing z), in general, as $(\mathcal{P}_t, \mathcal{P}_{t+1}), \forall t \in \mathcal{T}$ now are the inputs. At test time we predict the odometry as $\tilde{h}(\mathcal{P}_t, \mathcal{P}_{t+1}, z(\mathcal{P}_t, \mathcal{P}_{t+1}))$.

Algorithm 1 Depth Completion

INPUT: valid pixel set $\mathcal{P}_1 = \{\mathbf{p}_1\}$, void pixel set $\mathcal{P}_0 = \{\mathbf{p}_0\}$
OUTPUT: filled pixel set $\mathcal{P}_0 = \{\mathbf{p}_0\}$

1: **For Each** $\mathbf{p}_0$ in $\mathcal{P}_0$:
2: $\quad \mathbf{p}' \in \arg\min_{\mathbf{p}_1\in\mathcal{P}_1} \{||\mathbf{p}_0 - \mathbf{p}_1||\}$,
3: $\quad \mathbf{p}_0 \leftarrow \mathbf{p}^*$
4: **End**
5: Return $\mathcal{P}_0$

3.3 Data prepossessing

Depth spherical image generation from LiDAR point clouds.

The 3D LiDAR point clouds are usually stored by a set of Cartesian coordinates (X, Y, Z). Due to the relative low resolution of LiDAR scanners, the 3D LiDAR point clouds are quite sparse. A scan of the LiDAR point cloud is shown in Figure 3a. However, matching keypoints pairs from two consecutive scans of LiDAR point clouds would be inaccurate and time/memory consuming. Therefore, we project the sparse LiDAR point clouds to spherical projection images that are nearly dense. The projection function is:

$$\alpha = \arcsin(\frac{z}{\sqrt{x^2+y^2+z^2}}),\ \bar{\alpha} = |\frac{\alpha}{\Delta\alpha}| \tag{7}$$

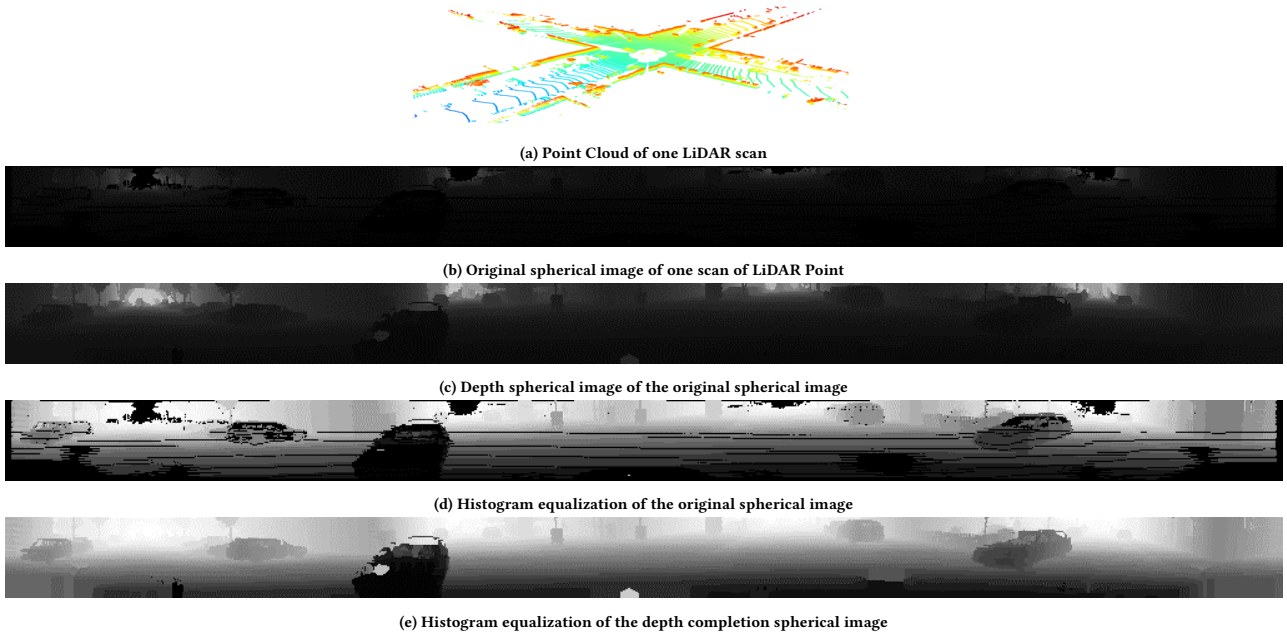
$$\beta = \arcsin(\frac{y}{\sqrt{x^2+y^2}}),\ \bar{\beta} = |\frac{\beta}{\Delta\beta}| \tag{8}$$

Where α and β are the indexes of points' position in the matrix. $\Delta\alpha$ and $\Delta\beta$ are the angular resolutions in the horizontal and vertical directions, respectively. The element at (α, β) of the spherical image is set to be the range value $r = \sqrt{x^2+y^2+z^2}$ of the LiDAR point (x, y, z) [12]. A matrix of size $H \times W \times C$ can be obtained by applying this projection. H is the number of vertical channels from LiDAR scanners, W is the width of the spherical image, and C is the channel of input point cloud matrix. Figure 3b shows the spherical image of the LiDAR scan in Fig 3a.

Depth completion and histogram equalization.

In depth completion we aim to fill the void pixels in our projected spherical image, so that every pixel can be utilized in the following algorithms. Traditional image inpainting algorithms try to interpolate the target pixels using surrounding pixel color values, which is computational expensive for real-time processing. In this paper, based on the assumption that depth value does not change much in local regions, we speed up the depth completion by filling the void pixels with the depth value of its nearest valid pixels. The algorithm is described in algorithm 1.

As shown in the Figure 3b, the value in each pixel represents the distance of the original LiDAR detected point to the LiDAR that causes the most pixels in the spherical image to have a relatively small value. To improve the spherical image's visual quality, we apply the histogram equalization technique in order to enhance the contrast as shown in Figure 3c. Histogram equalization is the most popular contrast enhancement technique due to its simplicity



(a) Point Cloud of one LiDAR scan

(b) Original spherical image of one scan of LiDAR Point

(c) Depth spherical image of the original spherical image

(d) Histogram equalization of the original spherical image

(e) Histogram equalization of the depth completion spherical image

Figure 3: The spherical image of one LiDAR scan

and effectiveness [10]. This widely used technique is achieved by flattening the dynamic range of an image's histogram of grey value based on the probability density function [27]. After we generate the depth completion spherical image in Figure 3d , we apply the Histogram equalization. Therefore, the brightness of the depth completion spherical image is improved significantly as shown in the Figure 3e.

Keypoints detection and matching.

After the depth completion and histogram equalization step, we want to detect a group of MKPs from consecutive frames of spherical images f_i and f_{i+1}. For example, one keypoint is shown in f_i at the location (x_1, y_1) and its matching point is located at (x_1', y_1') in f_{i+1}. Since this MKP represents the same object in consecutive frames, the potential odometry information between these consecutive frames is related to their location in spherical images. Thus, we apply the SIFT algorithm to detect keypoints in consecutive frames of spherical images. This local feature extraction method can extract comprehensive keypoints which are invariant to the object translation and rotation. Each detected keypoint will be represented as a 128−element feature vector, called by descriptors [8]. The feature vectors will be used for keypoint matching.

Given a keypoint from depth completion image f_i, we want to find the best matching point in f_{i+1} to form the MKP. The similarity will be measured by the Euclidean distance between keypoints in the spherical image [28]. In Figure 4a, two consecutive scans are concatenated vertically. The Figure 4b shows the top 400 MKPs detected from these consecutive scans. However, some MKPs are mismatched or not suitable for odometry estimation. For example, we do not want to use any point from dynamic objects. Figure 4c illustrates the 100 MKPs selected by MKP Selection module among the 1000 MKPs detected in this section. We will discuss this issue in section 3.4.

Projecting back to 3D point cloud.

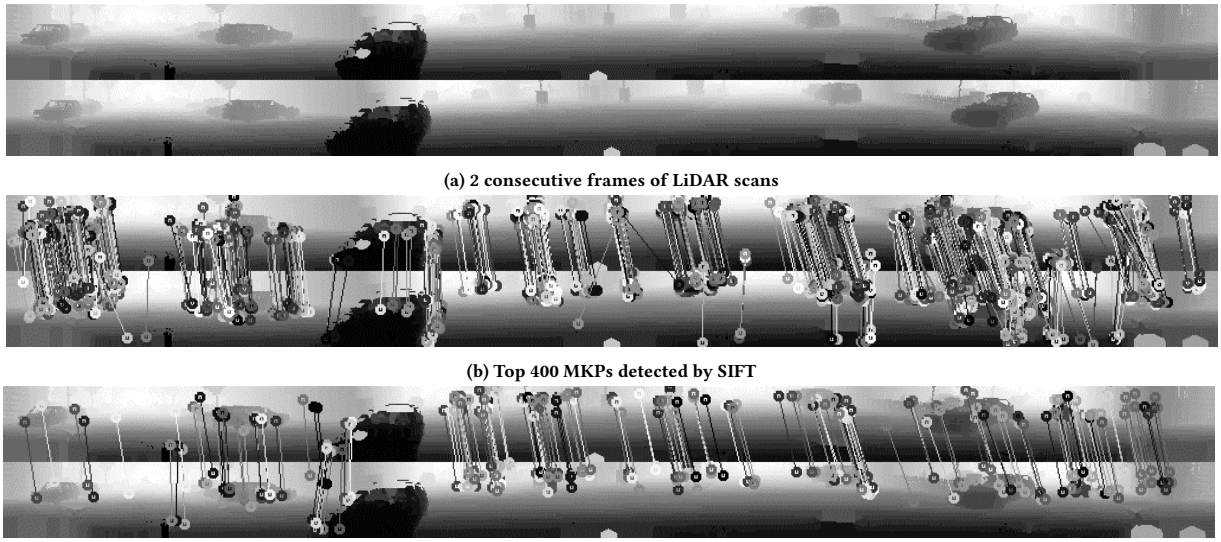
It is simple to project MKPs in the spherical image back to 3D point clouds. Noticed that the MKPs were detected from depth completion spherical image, some MKPs may not have corresponding points in the original 3D space. In other words, some "fake" MKPs are generated by depth completion, not from original LiDAR points. These "fake" MKPs are removed and only real MKPs return the points coordinate in 3D space. The MKPs which related to the odometry rotation and translation are extracted from entire LiDAR points of consecutive frames. Then we aligned the points in frame f_i with their matching points in frame f_{i+1} to construct a matrix with the size of $[m, 6]$ where m is the number of MKPs we find, and 6 features represent the x, y, z coordinates of matching points in consecutive frames.

3.4 LiDAR odometry regression

MKPs Selection module.

In Section 3.3, we extract MKPs of consecutive frames which contain the odometry rotation and translation information. However, some of MKPs are inappropriate for odometry estimation. We want to exclude those MKPs from dynamic objects such as moving cars. Even though matching are good, these MKPs may inhibit the odometry estimation due to the inconsistent odometry information of dynamic objects with LiDAR device. Hence a selection module is needed to determine whether a matching point should be used for odometry estimation.

we deploy a MKPs Selection module to solve this segmentation problem in order to improve the effectiveness and robustness of the network. Here we use the PointNet segmentation structure to



(a) 2 consecutive frames of LiDAR scans

(b) Top 400 MKPs detected by SIFT

(c) 100 MKPs selected by MKPs Selection Module

Figure 4: MKPs extracted from consecutive LiDAR scans

Algorithm 2 Data Augmentation on Odometry Rotation

INPUT: Point cloud pair $\mathcal{P}_1, \mathcal{P}_2$, odometry Matrix Ground Truth Tr, augmentation rotation upper-bound $beta_{max}$
OUTPUT: Augmented point cloud pair $\mathcal{P}'_1, \mathcal{P}'_2$, augmented odometry Matrix Ground Truth Tr'

$\beta \leftarrow Random(-\beta_{max}, \beta_{max})$
$Tr1 \leftarrow Yaw(\beta)$
$Tr' \leftarrow [Tr]^{-1}\mathcal{P}_2$
$\mathcal{P}'_1 \leftarrow \mathcal{P}_1$
$\mathcal{P}'_2 \leftarrow Tr1\mathcal{P}_1$
Return $\mathcal{P}'_1, \mathcal{P}'_2, Tr'$

achieve the goal. The PointNet consists of symmetry function for unordered input, a local and global information aggregation, and a joint alignment network. In our MKPs Selection module, we modify several parts to fit our scenario. T-Net structure has been removed since we don't want to preserve the rotation invariant. The input are MKPs of consecutive frames, not original unordered points. For the output $M_S(P_i) \in [0, 1]$, ground truth odometry which containing rotation R and translation T information are used to calculate the distance of the point in frame i with its matching point's projection in frame i by the following equation:

$$d = \| \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \mathcal{X}_i - \mathcal{X}_{i+1} \|_2 \tag{9}$$

The output is set as 1 if the distance is small while large distance indicates the output is 0. MKPs with label 1 will be use in the odometry estimation.

Rotation Estimation module.

After selecting a fixed number of MKPs, we aim to infer the 3-DoF relative odometry rotation information by constructing a regression model. By observing popular autonomous driving datasets with LiDAR odometry task, recording cars usually go straight line while turning frames are not enough. Hence, we apply a data augmentation procedure for rotation estimation. Because each matching pair associate with the 3-DoF odometry rotation information R, the Eqn. 3 represents the relation between matching pairs with a rotation matrix. We can generate the augmented MKPs with corresponding rotation matrix by given the existing MKPs with their rotation matrix based on the Algorithm 2.

The structure of rotation estimation module is based on the PointNet classification model while we change it to the regression model. We remove the T-net structure and add our rotation augmentation structure. The input of the Rotation Estimation module is selected MKPs from the MKPs Selection module concatenate with rotation augmentation of selected MKPs. The output is the Unit Quaternion format of the odometry rotation information. The Unit Quaternion of two consecutive frames can be represented by a vector $[a, bi, cj, dk]$, where a, b, c, and d are real numbers, and i, j, and k are the fundamental Quaternion units [25]. Based on the norm of the Unit Quaternion vector is equal to 1, here we only predict the first three dimension of Unit Quaternion and the last dimension can be calculated afterwards. The rotation loss function of consecutive frames is defined as:

$$\mathcal{L}_R = \|Q - \hat{Q}\|_l \tag{10}$$

Where Q is the ground truth first three dimension of Unit Quaternion, $\hat{Q}$ is the predicted first three dimension of Unit Quaternion by the network, and l refers to the Euclidean distance norm. Here we choose the l_2 norm in this module.

Translation Estimation module.

The network structure of Translation estimation module is quite similar to the Rotation Estimation module. In contrast, we directly apply the selected MKPs as input to feed the Translation Estimation module since the translation augmentation is not required. The output is the 3-DoF odometry translation information and the loss function is defined as:

$$\mathcal{L}_T = \|T - \hat{T}\|_l \tag{11}$$

Where T is the ground truth translation matrix, $\hat{T}$ is the predicted translation array by the network, and l refers to the Euclidean distance norm. Here we choose the l_2 norm in this module.

4 EXPERIMENTS

4.1 Implementation detail

In our experiment, we use the point cloud data which is collected by the Velodyne HDL-64 3D LiDAR sensor to estimate the odometry. When converting the LiDAR point clouds to spherical images, we set the height of the image to 64 and the width to 1024. For each two consecutive scans f_i and f_{i+1}, we detect MKPs on spherical images then convert back to 3D space. In a sequence of point clouds with $n+1$ scans, we can collect 1000 MKPs for n consecutive frames to form an input matrix $[n, 1000, 6]$. LodoNet predicts the odometry between n consecutive frames and give an output as $[n, 7]$, then we can calculate the odometry matrix $[n, 12]$ indicating each scan's odometry by the relation of rotation and translation matrix. The whole framework is implemented with the Tensorflow, we choose the Adam optimizer[11] for optimization. The batch size and learning rate are set to 128 and 0.0001. MKPs Selection module select 100 MKPs among 1000 MKPs for each consecutive scans. One GeForce RTX 2080 Ti GPU is used for training.

4.2 Dataset

KITTI. The KITTI odometry dataset[9] is a popular and widely used dataset in the autonomous driving tasks. It provides 22 independent sequences with stereo gray-scale camera images, color camera images, and point clouds captured by a Velodyne HDL-32 LiDAR sensor among urban, highway and countryside scenes. For our work, we use Velodyne LiDAR data only to estimate odometry. For sequence 0 to sequence 10, KITTI dataset provides the ground truth odometry, while for sequence 11 to 21 the ground truth is preserved for online benchmark testing. Hence, we use sequence 0, 1, 2, 3, 4, 5, 6, 9, and 10 to train our model while leaving sequence 7 and sequence 8 for validation.

As we state in section 3.4 that odometry has 6 degrees of freedom which can be represented by rotation and translation components. In the ground-truth odometry pose files provided by KITTI dataset, a 12 by 1 vector is assigned for the odometry pose of i_{th} frame to the 0 frame while 9 of them indicating the rotation and 3 of them present the translation. This 9-dimension rotation vector usually reshapes to a 3 by 3 rotation matrix. In our experiments, we convert this 3 by 3 rotation matrix to a 1 by 4 Unit Quaternion representation due to less dimension for network training. Noted here we only predict first three dimension of Unit Quaternion and the last dimension can be calculated directly since the norm of the Unit Quaternion is equal to 1. Thus, given the input of the MKPs of two consecutive frames, the output of our network is a 1 by 7 vector which is concatenated by a 1 by 4 rotation vector with a 1 by 3 translation vector.

4.3 Evaluation

We compared our method with the ground truth trajectory and several LiDAR odometry estimation methods: ICP-point2point(ICP-po2po), ICP-point2plane (ICP-po2pl), GICP[26], CLS[29], LOAM[38], Velas et al.[30], LO-Net[12].

Table 1 shows the evaluation results of the mentioned methods on the KITTI dataset. We use t_{rel}: the Average Transnational RMSE(%) and r_{rel}: Average Rotational RMSE(°/100m) to evaluate the results of different methods. There are few deep learning-based approaches for LiDAR odometry estimation that have comparable results. DeepPCO[32] only reports the results on its validation sequences. However they did not specify the unit of t_{rel} and r_{rel}. Based on the trajectories they provide, we determine that their t_{rel} is 2.63 and r_{rel} is 3.05 for sequence 04, and t_{rel} is 2.47 and r_{rel} is 6.59 for sequence 10 with the same unit as in Table 1. CAE-LO[36] did not provide the results on KITTI Seq 00-10. LO-Net[12] is one of the best deep learning methods for LiDAR based odometry estimation. From Table 1, The Seq 07 and 08 are not used to train LodoNet. The bold number indicates the best performance among all the methods, and the blue number indicates the runner-up. In some sequences, our results are even better than LOAM. However, LOAM still remains the best option which is the state-of-art Geometry based approach. Until now there is no deep learning method can beat the LOAM algorithm, but it is clear that deep learning methods become more and more accurate.

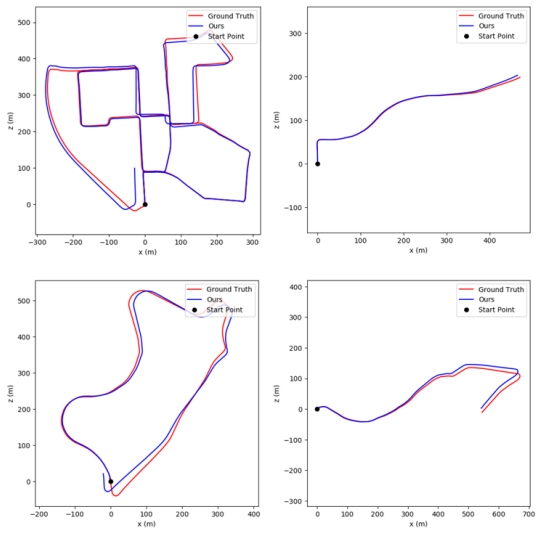


Figure 5: 2D estimate trajectory of our training sequences: KITTI Seq.00 (upper left), Seq.03 (upper right), Seq.09 (lower left), Seq.10 (lower right) with ground truth.

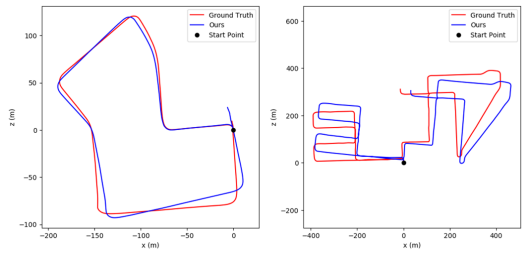
Figure 5 shows our estimated 2D trajectory plots of our training sequences: KITTI sequence 00, 03, 09, and 10 with ground-truth. Figure 6a shows our estimated 2D trajectory plots of our testing sequences: KITTI sequence 07 and 08 with ground-truth. The blue line is our estimated trajectory and the red line is the ground truth trajectory. Our LodoNet can produce accurate pose estimation with respect to ground truth. The average errors of translation and rotation with respect to path length interval of KITTI sequence 07 and 08 are shown in the Figure 6b and 6c respectively.

4.4 Ablation study

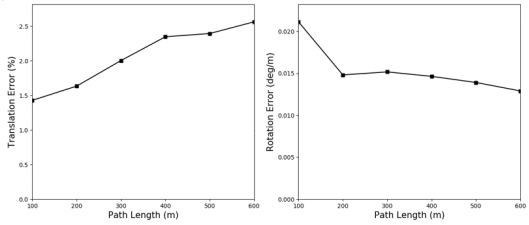
In this section, we investigate the effects of different factors of our odometry estimation on the KITTI dataset. We change the

Table 1: Odometry results on KITTI dataset

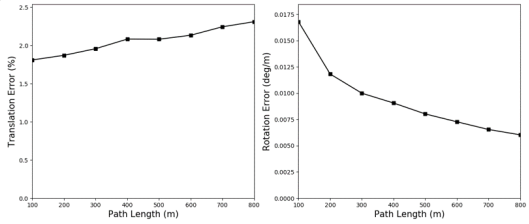
seq	ICP-po2po		ICP-po2pl		GICP[26]		CLS[29]		LOAM[38]		Velas et al[30]		DeepPCO[32]		LO-Net[12]		ours	
	t_rel	r_rel	t_rel	r_rel	t_rel	r_rel	t_rel	r_rel	t_rel	r_rel	t_rel	r_rel	t_rel	r_rel	t_rel	r_rel	t_rel	r_rel
00	6.88	2.99	3.80	1.73	1.29	0.64	2.11	0.95	**0.78**	**0.53**	3.02	/	/	/	1.47	0.72	1.43	0.69
01	11.21	2.58	13.53	2.58	4.39	0.91	4.22	1.05	1.43	0.55	4.44	/	/	/	1.36	0.47	**0.96**	**0.28**
02	8.21	3.39	9.00	2.74	2.53	0.77	2.29	0.86	**0.92**	**0.55**	3.42	/	/	/	1.52	0.71	1.46	0.57
03	11.07	5.05	2.72	1.63	1.68	1.08	1.63	1.09	**0.86**	**0.65**	4.94	/	/	/	1.03	0.66	2.12	0.98
04	6.64	4.02	2.96	2.58	3.76	1.07	1.59	0.71	0.71	0.50	1.77	/	2.84	3.07	**0.51**	0.65	0.65	**0.45**
05	3.97	1.93	2.29	1.08	1.02	0.54	1.98	0.92	**0.57**	**0.38**	2.35	/	/	/	1.04	0.69	1.07	0.59
06	1.95	1.59	1.77	1.00	0.92	0.46	0.92	0.46	0.65	0.39	1.88	/	/	/	0.71	0.50	**0.62**	**0.34**
07	5.17	3.35	1.55	1.42	0.64	**0.45**	1.04	0.73	**0.63**	0.50	1.77	/	/	/	1.70	0.89	1.86	1.64
08	10.04	4.93	4.42	2.14	1.58	0.75	2.14	1.05	**1.12**	**0.44**	2.89	/	/	/	2.12	0.77	2.04	0.97
09	6.93	2.89	3.95	1.71	1.97	0.77	1.95	0.92	0.77	0.48	4.94	/	/	/	1.37	0.58	**0.63**	**0.35**
10	8.91	4.74	6.13	2.60	1.31	0.62	3.46	1.28	**0.79**	0.57	3.27	/	2.41	6.70	1.80	0.93	1.18	**0.45**
aveage	7.36	3.41	4.74	1.93	1.92	0.73	2.12	0.91	**0.84**	**0.46**	3.15	/	/	/	1.33	0.69	1.27	0.66



(a) Estimated trajectory plots of KITTI Seq. 07(Left) and 08(Right) with ground truth.



(b) The average errors of translation and rotation with respect to path length interval of KITTI Seq. 07



(c) The average errors of translation and rotation with respect to path length interval of KITTI Seq. 08

Figure 6: Evaluation of our estimation on our testing set:KITTI Seq. 07 and Seq. 08

observation parameter while others remain as default parameters to evaluate our network.

Rotation augmentation in Rotation Estimation Module

In section 3.4, we clam that our rotation data augmentation procedure contribute to estimate 3-DoF relative odometry rotation information. We list the results of KITTI Seq. 07 and 08 in Table 2 with different rotation augmentation ratio. It proves that our rotation data augmentation procedure can improve our network performance and when r = 0.05 achieves the best performance.

Table 2: Comparison of different combinations of the rotation augmentation ratio.

	with augmentation				w/o
	ratio r	0.03	0.04	0.05	0.06	augmentation
Seq 07	t_rel	2.99	2.48	**1.86**	2.09	3.77
	r_rel	1.79	1.58	**1.64**	1.46	2.18
Seq 08	t_rel	3.46	2.09	**2.04**	4.59	4.33
	r_rel	1.47	0.98	**0.97**	2.01	1.80

Table 3: Comparison of top k MKPs choose by MKPs selection module.

	Top k	50	100	200	300
Seq 07	t_rel	3.70	**1.86**	2.97	4.22
	r_rel	2.84	**1.64**	1.89	2.28
Seq 08	t_rel	3.20	**2.04**	5.38	4.09
	r_rel	1.65	**0.97**	2.04	2.00

Number of MKPs selected by MKPs Selection Module

As we state in section 4.1, we extract 1000 MKPs on consecutive spherical images. We compare the results on KITTI Seq. 07 and 08 with different numbers as shown in the Table 3. When choosing 100 MKPs, the network achieves the best performance.

5 CONCLUSIONS

In this paper,we present a novel deep learning-based LiDAR odometry estimation framework named LodoNet. Within the framework we propose a new approach that extract the matched keypoint pairs(MKPs) by applying conventional image-based feature describer from projected LiDAR images. With the help of PointNet, we adopt the MKPs to estimate the movements between the LiDAR frames, which finally result in the LiDAR odometry estimation. Experiments on KITTI dataset demonstrate the effectiveness of our framework compared with existing deep learning approaches. Moreover, since our framework is mainly integrated by a conventional feature describer and a light-weighed neural network, which can be easily deployed to the automated driving systems without allocating many computational resources. In our future work, we are going to explore how to further integrate the MKPs extraction and odometry estimation steps for a more accurate and efficient LiDAR odometry estimator.

REFERENCES

[1] Yasin Almalioglu, Muhamad Risqi U. Saputra, Pedro P. B. de Gusmao, Andrew Markham, and Niki Trigoni. 2018. GANVO: Unsupervised Deep Monocular Visual Odometry and Depth Estimation with Generative Adversarial Networks. *CoRR* abs/1809.05786 (2018). arXiv:1809.05786 http://arxiv.org/abs/1809.05786

[2] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. Surf: Speeded up robust features. In *European conference on computer vision*. Springer, 404–417.

[3] P. J. Besl and N. D. McKay. 1992. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14, 2 (1992), 239–256.

[4] Paul J Besl and Neil D McKay. 1992. Method for registration of 3-D shapes. In *Sensor fusion IV: control paradigms and data structures*, Vol. 1611. International Society for Optics and Photonics, 586–606.

[5] Luca Caltagirone, Samuel Scheidegger, Lennart Svensson, and Mattias Wahde. 2017. Fast LIDAR-based road detection using fully convolutional neural networks. In *2017 ieee intelligent vehicles symposium (iv)*. IEEE, 1019–1024.

[6] A. Censi. 2008. An ICP variant using a point-to-line metric. In *2008 IEEE International Conference on Robotics and Automation*. 19–25.

[7] Y. Chen and G. Medioni. 1991. Object modeling by registration of multiple range images. In *Proceedings. 1991 IEEE International Conference on Robotics and Automation*. 2724–2729 vol.3.

[8] Johnny Chien, Chen-Chi Chuang, Chia-Yen Chen, and Reinhard Klette. 2016. When to use what feature? SIFT, SURF, ORB, or A-KAZE features for monocular visual odometry. 1–6.

[9] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[10] Rafael C. Gonzalez and Richard E. Woods. 2008. *Digital image processing*. Prentice Hall, Upper Saddle River, N.J.

[11] Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (12 2014).

[12] Qing Li, Shaoyang Chen, Cheng Wang, Xin Li, Chenglu Wen, Ming Cheng, and Jonathan Li. 2019. LO-Net: Deep Real-time Lidar Odometry. *CoRR* abs/1904.08242 (2019). arXiv:1904.08242 http://arxiv.org/abs/1904.08242

[13] Kok-Lim Low. 2004. Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration. (01 2004).

[14] David G Lowe. 1999. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, Vol. 2. Ieee, 1150–1157.

[15] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. 2019. DeepICP: An End-to-End Deep Neural Network for 3D Point Cloud Registration. *CoRR* abs/1905.04153 (2019). arXiv:1905.04153 http://arxiv.org/abs/1905.04153

[16] Weixin Lu, Yao Zhou, Guowei Wan, Shenhua Hou, and Shiyu Song. 2019. L3-net: Towards learning based lidar localization for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 6389–6398.

[17] Yecheng Lyu, Lin Bai, and Xinming Huang. 2018. Chipnet: Real-time lidar processing for drivable region segmentation on an fpga. *IEEE Transactions on Circuits and Systems I: Regular Papers* 66, 5 (2018), 1769–1779.

[18] Yecheng Lyu, Lin Bai, and Xinming Huang. 2018. Real-time road segmentation using lidar data processing on an fpga. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.

[19] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. 2019. Rangenet++: Fast and accurate lidar semantic segmentation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*.

[20] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. 2013. Comparing ICP variants on real-world data sets. *Autonomous Robots* (04 2013). https://doi.org/10.1007/s10514-013-9327-2

[21] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE* (2017).

[22] T. Qin, P. Li, and S. Shen. 2018. VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Transactions on Robotics* 34, 4 (2018), 1004–1020.

[23] E Recherche, Et Automatique, Sophia Antipolis, and Zhengyou Zhang. 1992. Iterative Point Matching for Registration of Free-Form Curves. *Int. J. Comput. Vision* 13 (07 1992).

[24] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. 2011. ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision*. Ieee, 2564–2571.

[25] M. Schwaab, M. Romanovas, D. Plaia, T. Schwarze, and Y. Manoli. 2016. Fusion of visual odometry and inertial sensors using dual quaternions and stochastic cloning. In *2016 19th International Conference on Information Fusion (FUSION)*. 573–580.

[26] Aleksandr Segal, Dirk Hähnel, and Sebastian Thrun. 2009. Generalized-ICP. *Proc. of Robotics: Science and Systems*. https://doi.org/10.15607/RSS.2009.V.021

[27] Kuldeep Singh, Dinesh K. Vishwakarma, Gurjit Singh Walia, and Rajiv Kapoor. 2016. Contrast enhancement via texture region based histogram equalization. *Journal of Modern Optics* 63, 15 (2016), 1444–1450. arXiv:https://doi.org/10.1080/09500340.2016.1154194

[28] Norhayati Suaib, Mohammad Hamiruce Marhaban, M Iqbal Saripan, and Siti Ahmad. 2014. Performance evaluation of feature detection and feature matching for stereo visual odometry using SIFT and SURF. *IEEE TENSYMP 2014 - 2014 IEEE Region 10 Symposium*, 200–203.

[29] Martin Velas, Michal Spanel, and Adam Herout. 2016. Collar Line Segments for fast odometry estimation from Velodyne point clouds. 4486–4495. https://doi.org/10.1109/ICRA.2016.7487648

[30] Martin Velas, Michal Spanel, Michal Hradis, and Adam Herout. 2017. CNN for IMU Assisted Odometry Estimation using Velodyne LiDAR. *CoRR* abs/1712.06352 (2017). arXiv:1712.06352 http://arxiv.org/abs/1712.06352

[31] S. Wang, R. Clark, H. Wen, and N. Trigoni. 2017. DeepVO: Towards end-to-end visual odometry with deep Recurrent Convolutional Neural Networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2043–2050.

[32] Wei Wang, Muhamad Risqi Utama Saputra, Peijun Zhao, Pedro Gusmao, Bo Yang, Changhao Chen, Andrew Markham, and Niki Trigoni. 2019. DeepPCO: End-to-End Point Cloud Odometry through Deep Parallel Neural Network. 3248–3254. https://doi.org/10.1109/IROS40897.2019.8967756

[33] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. 2018. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1887–1893.

[34] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. 2019. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 4376–4382.

[35] Nan Yang, Rui Wang, Jörg Stückler, and Daniel Cremers. 2018. Deep Virtual Stereo Odometry: Leveraging Deep Depth Prediction for Monocular Direct Sparse Odometry. *CoRR* abs/1807.02570 (2018). arXiv:1807.02570 http://arxiv.org/abs/1807.02570

[36] Deyu Yin, Qian Zhang, Jingbin Liu, Xinlian Liang, Yunsheng Wang, Jyri Maanpää, Hao Ma, Juha Hyyppä, and Ruizhi Chen. 2020. CAE-LO: LiDAR Odometry Leveraging Fully Unsupervised Convolutional Auto-Encoder for Interest Point Detection and Feature Description.

[37] Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian D. Reid. 2018. Unsupervised Learning of Monocular Depth Estimation and Visual Odometry with Deep Feature Reconstruction. *CoRR* abs/1803.03893 (2018). arXiv:1803.03893 http://arxiv.org/abs/1803.03893

[38] Ji Zhang and Sanjiv Singh. 2017. Low-drift and Real-time Lidar Odometry and Mapping. *Autonomous Robots* 41 (02 2017), 401–416. https://doi.org/10.1007/s10514-016-9548-2