Pseudo-LiDAR for Visual Odometry

Yanzi Miao, Member, IEEE, Huviying Deng, Chaokang Jiang, Zhiheng Feng, Xinrui Wu, Guangming Wang, Member, IEEE, and Hesheng Wang, Senior Member, IEEE

Abstract—As one of the important tasks in the field of robotics and machine vision, visual odometry provides tremendous help for various applications such as navigation, location, and so on. Conventionally, the task of visual odometry mainly relies on the input of continuous images. However, it is very complicated for the odometry network to learn the epipolar geometry information provided by the images. Since the 6-degree-of-freedom (DoF) pose transformation occurs in 3-D space and learning poses from 3-D point clouds are more straightforward, this article introduces the concept of pseudo-LiDAR to the odometry task. The pseudo-LiDAR point cloud is formed by backprojecting the depth map generated from the image into 3-D space. Due to the limitation of calculation power, most current algorithms based on the point cloud need to sample 8192 points from the point cloud as input, but such an approach makes the rich point cloud information in the pseudo-LiDAR point cloud not fully utilized. To address this problem, a projection-aware algorithm is adopted, which achieves efficient point cloud learning and improves the accuracy of the network while preserving the 3-D structure information in the pseudo-LiDAR point cloud. Finally, an image-only 2-D–3-D fusion module is proposed to enhance the pseudo-LiDAR point features using information such as the texture and color of the images. Through multimodal fusion, the network achieves a deeper understanding of the environment. Experiments on the KITTI dataset prove the effectiveness of our method. The source code will be open-sourced at https://github.com/IRMVLab/Pseudo-LiDAR-for-Visual-Odometry.

Index Terms—Deep learning, feature fusion, projection-aware, pseudo-LiDAR, visual odometry.

I. INTRODUCTION

ODOMETRY uses an incremental method to estimate the relative pose of the mobile robot. Traditional visual odometry works [1], [2] feed images directly into the odometry network. The performance of such methods is limited by the fact that the images do not provide explicit 3-D information to help the odometry network learn the 6-degree-of-freedom (DoF) pose in space. With the development of odometry, more and more research combine odometry with other tasks, such as depth estimation [3], [4], [5]. Multitask learning is utilized to achieve a deeper understanding of the scene. The addition of depth brings rich spatial information to the odometry task. This is in line with our idea. However, unlike other methods, a different approach is taken to combining depth with odometry, that is, pseudo-LiDAR. By learning geometry, shape, and scale information contained in 3-D point clouds, the surrounding environment can be better understood. The introduction of pseudo-LiDAR allows the visual odometry to learn from explicit 3-D coordinates. For the generation of pseudo-LiDAR point clouds, specifically, a stereo matching network is used to estimate the depth map and then the generated depth map is utilized to backproject each pixel in the image into 3-D space. Compared with the LiDAR point cloud, the pseudo-LiDAR point cloud is denser. Even with 64-beam LiDAR, the point cloud it provides is very sparse (cf. Fig. 1). Due to the limitation of calculation power, most of the previous LiDAR-based odometry methods [6] use 8192 points as input to the odometry network. If such a method is adopted in the pseudo-LiDAR-based odometry, a large amount of 3-D geometric information in the pseudo-LiDAR is discarded. Therefore, the projection-aware method is used to learn the features of dense pseudo-LiDAR point cloud efficiently. Many projection-based methods lose depth information when projecting point clouds to the 2-D plane. Wang et al. [7] assign the 3-D coordinates in the point cloud to the projected point so that the information in the point cloud is not lost while learning the point cloud efficiently. In our approach, the 3-D coordinates \((x, y, z)\) of the pseudo-LiDAR point cloud are reprojected into their original 2-D coordinates \((u, v)\), that is, each pixel grid is placed with its corresponding 3-D coordinates.

To achieve a deeper understanding of the point cloud by the network, image texture information is used to enhance the geometric features of the pseudo-LiDAR point cloud. Since there is a one-to-one correspondence between the pseudo-LiDAR and the image itself, the fusion of the image and the point cloud can be performed as long as they are both sampled in the same steps at each layer of the pyramid.

The main contributions of this article are as follows.

1) By introducing the concept of pseudo-LiDAR, a new framework of visual odometry based on the LiDAR algorithm is proposed. The 3-D spatial information provided by the pseudo-LiDAR allows the visual odometry network to learn more directly about the 6-DoF pose in space.
2) A projection-aware method is adopted to take advantage of the dense point cloud in pseudo-LiDAR. This method achieves more efficient point cloud sampling and grouping by projecting the 3-D coordinates of the pseudo-LiDAR point cloud back into its original pixel coordinates.

3) To make full use of the texture and geometric information contained in images, image-only based 2-D–3-D fusion is designed.

4) The experimental results on the KITTI dataset [8] and the ablation experiments show that the idea of introducing projection-aware pseudo-LiDAR and image-only based 2-D–3-D fusion can effectively improve the performance of odometry estimation.

II. RELATED WORK

A. Odometry

Methods of visual odometry can be mainly divided into geometry-based methods [9] and deep-learning-based methods [1], [10]. With the development of deep learning, Wang et al. [1] propose the first end-to-end visual odometry framework based on deep learning to achieve the camera pose. It uses convolutional neural networks (CNNs) and deep recurrent neural networks (RNNs) to learn image features and the relationship between sequences, respectively. To emphasize the importance of different semantic types for ego-motion estimation, DAVO [11] used dynamic attention to weigh different semantic types in images. BeyondTracking [12] proposed the memory component to adaptively save global information. Currently, more and more research combine odometry with other tasks such as optical flow [13] and depth [3], [4], [14], [15], [16] to realize mutual promotion in multitask joint learning. DeepAVO [13] uses the image to compute the optical flow, after which the optical flow is fed into four different CNN networks to learn the pixel motion in different directions. Zhou et al. [3] combine a depth network with a pose network. With the regressed depth and pose, the target frame can be reconstructed from the source frame, so that the photometric consistency can be used to train the networks. Similar to this method, Bian et al. [4] propose to train the networks with a geometric consistency loss between the depth map of the source frame after warping and the depth map of the target frame. For the depth and pose estimated by the two CNN networks, the network proposed by DOC+ [17] uses an online correction module to optimize the pose. Unlike previous approaches, Mahjourian et al. [18] use depth to convert 2-D pixels into 3-D point clouds. Iterative closest point (ICP) is proposed to register two frames of a 3-D point cloud to adjust the ego-motion estimated by the network. These methods all combine depth and pose, but the combination is only connected by loss, and the depth information is not introduced more deeply into the learning of the pose. By contrast, Tiwari et al. [19] integrate the depth map generated by CNNs with the RGB image. SLAM is executed using the generated pseudo-RGB-D features. The visual odometry network proposed by Wang et al. [20] use the connection of image and depth map estimated by the depth network as input to learning the 6-DoF pose. Zou et al. [10] propose a pose network with a two-layer convolutional LSTM module. The pre-extracted depth features are used as the input of the second layer of the ConvLSTM module. Li et al. [21] propose to use a network to estimate pose and depth simultaneously. The network is trained with stereo images to recover scale, but continuous monocular images are used for testing. Yin and Shi [14] explore the combination of depth, pose, and optical flow. The regressed depth and pose are used to recover rigid motion in the scene, while the nonrigid motion of dynamic objects is computed by the residual flow network. While many methods introduce depth information into visual odometry, most current vision-based odometry methods perform slightly inferior to LiDAR-based methods. The LiDAR-based method proposed by Wang et al. [6] uses the structure of the pyramid, warping, and cost volume (PWC) to learn 3-D point cloud directly, with the hierarchical refinement module and the outlier points mask to achieve high-performance pose learning.

B. Pseudo-LiDAR

Recently, the pseudo-LiDAR point cloud has been widely used in the field of object detection, which improves the performance of object detection. Wang et al. [22] believe that the reason for the accuracy difference between image-based methods and LiDAR-based methods lies in the representation of data. Weng and Kitani [23] also utilize the pseudo-LiDAR point cloud for 3-D object detection. The 2-D–3-D bounding box consistency loss is used to deal with the noise in the pseudo-LiDAR point cloud. To solve the point cloud quality problem in the pseudo-LiDAR point cloud interpolation, Liu et al. [24] propose a multimodal feature fusion module to fuse the depth and texture information in RGB. You et al. [25] reduce the error of pseudo-LiDAR point cloud by improving the accuracy of stereo depth estimation. In other fields, pseudo-LiDAR gradually gains attention. For example, Wang et al. [26] and Jiang et al. [27] apply pseudo-LiDAR to 3-D scene flow estimation.

Inspired by these related works, pseudo-LiDAR is introduced into the odometry task. So that the visual odometry can establish the spatial connection of the points through the explicit 3-D coordinates to recover the camera pose.
III. PROBLEM DEFINITION

In this section, the structure of the proposed visual odometry network will be introduced in detail. As shown in Fig. 2, the inputs of our network are two consecutive stereo image pairs \((I_{1R}, I_{1L}), (I_{2R}, I_{2L})\), and the outputs are the quaternion \(q \in \mathbb{R}^4\) and translation vector \(t \in \mathbb{R}^3\), which represent a transformation to the mobile camera.

A. Pseudo-LiDAR Generation

Given a pair of stereo images, the core problem of depth estimation is to obtain the disparity \(d\) of each pixel in the image. To obtain the disparity, the corresponding pixels of the left and right cameras need to be matched. The disparity is the horizontal distance of the corresponding pixels in the left and right images. With the disparity, the depth \(Z\) can be calculated using the camera focal length \(f\) and the baselines \(b\), and the formula is

\[
Z = \frac{f \times b}{d}.
\]

For the depth estimation module, in this article, the stereo depth network GA-Net [28] is used to estimate the disparity of the stereo images. The specific conversion process is as follows. With the estimated depth map and camera intrinsics, the 3-D coordinates \((x, y, z)\) of each pixel \((u, v)\) in images can be calculated

\[
\begin{align*}
z &= Z \\
x &= \frac{(u - c_u) \times z}{f_u} \\
y &= \frac{(v - c_v) \times z}{f_v}
\end{align*}
\]

where \(Z\) is the estimated depth of the pixel in the camera coordinate and \((c_u, c_v)\) is the pixel location of the camera center. \(f_u\) and \(f_v\) are the focal lengths of the camera along the \(x\) - and \(y\) -axes, respectively. Points 2 m above the ground in the generated pseudo-LiDAR are removed as sky points with estimation errors. In addition, points at distances over 30 m are filtered out due to the low accuracy of depth estimation at remote points.

B. Feature Pyramid

In the proposed network, the 3-D coordinates \((x, y, z)\) in the pseudo-LiDAR point cloud will be projected back to their original pixel coordinates \((u, v)\) to generate the PM of \(H \times W \times 3\) as the input of the point stream. Here, \(H\) and \(W\) are the width and height of the input image, respectively. The three channels are the \(x\), \(y\), and \(z\) coordinates of the point cloud, respectively. PM\(_1\) and PM\(_2\) represent the point maps of two consecutive frames.

1) Projection-Aware Point Feature Pyramid: Fig. 3 illustrates the details of the projection-aware set conv layer. Inspired by [7], the stride-based sampling is used to downsample the point map PM\(_1\) and PM\(_2\). Based on the sampling step set for each pyramid level, points at fixed intervals in the 2-D plane are indexed. The stride-based sampling [7] can obtain the index of the sampled points faster on the basis of uniform sampling. Then, each indexed sampled point is used as a center point to aggregate its surrounding points. Specifically, for each center point \(x^c_k\), the corresponding points are found in the projected point map before sampling and a fixed size kernel is set around it. After that, K-nearest neighbor (KNN) is used within the kernel to obtain the points \(x^c_k\) \((k = 1, 2, \ldots, K)\) around the center. After obtaining the center point and its surrounding grouped points, multilayer perceptron (MLP) and max pooling are used to perform feature aggregation to obtain...
the feature of the center point. The formula is

\[ f_i = \text{maxpool}(\text{MLP}((x_i^k - x_i) \oplus f_i^k \oplus f_i^r)) \]  (3)

where \( f_i^r \) and \( f_i^l \) are the original features of \( x_i^r \) and \( x_i^l \). \( f_i \) is the output feature of the center point \( x_i \). \( \oplus \) denotes the concatenation of two vectors, and maxpool(\( i \)) indicates the max pooling operation.

2) Two-Dimensional–Three-Dimensional Fusion: To use the texture information in the images to enhance the network’s understanding of point clouds, the point stream fuses the extracted image features with their corresponding point features. Unlike the traditional 2-D–3-D fusion of image and point cloud, our method does not need to use external parameters to correspond to the image and point cloud, because the image and pseudo-LiDAR point cloud itself are one-to-one correspondence. At each image feature pyramid, two 3 × 3 convolutional layers, a batch normalization layer, and a ReLU activation function are used to extract image features. The convolution step is set to match the sampling step of the corresponding point stream to ensure the correspondence between the image and the point map. After getting the corresponding point and image features, the image features \{G\} \( i \in \mathbb{R}_{i=1}^{l} \times H \times W \}_{l=1}^{L} \) and point features \{F\} \( l \in \mathbb{R}_{i=1}^{l} \times N \}_{l=1}^{l} \) can be fused as shown in Fig. 4. Since the texture information provided by images is susceptible to the outdoor environment, a weight \( w \) is designed, as in many other methods [29], [30], to evaluate the importance of image features on the point features. To obtain the weight \( w \), 2-D convolution and 1-D convolution are applied to image features and point features, respectively, to reduce the feature channels. After that, the convolved features are concatenated and then the information interaction between texture and geometric channels is achieved using 2-D convolution while performing channel reduction. Finally, the generated weights \( w \) of dimension \( H^l \times W^l \times 1 \) are normalized using a sigmoid function. The formula is

\[ w = \sigma \left( C_2(C_2(G^l) + C_1(F^l)) \right) \]  (4)

where \( C_2(\cdot) \) and \( C_1(\cdot) \) represent the 2-D and 1-D convolution, respectively, and \( \sigma(\cdot) \) is the sigmoid activation function. Finally, the fused point cloud features can be computed by

\[ F^l_{\text{fused}} = C_1(w \odot G^l + F^l) \]  (5)

where \( \odot \) denotes the multiplication between elements.

3) Attentive Cost Volume: After obtaining the point feature from the pyramid module, we use cost volume [7], [31] to compute the embedding feature \( E \) between \( PM_1 \) and \( PM_2 \). First, for each point in \( PM_1 \), a fixed-size kernel is set in \( PM_2 \), and its neighboring points are found in the kernel using KNN, followed by the first embedded features obtained by attention-based feature aggregation. Then, similarly, for each point in \( PM_1 \), a fixed-size kernel is set around it, and the features of its neighbors are aggregated within the kernel in \( PM_2 \) to obtain the feature \( E \). The mask \( M \) that masks the dynamic object is obtained by using sharedMLP and softmax on the embedding features \( E \).

For \( E = \{e_i \mid e_i \in \mathbb{R} \}_{i=1}^{n} \) and \( M = \{m_i \mid m_i \in \mathbb{R} \}_{i=1}^{n} \), the pose can be regressed by the following calculation:

\[ q = \frac{\text{FC}(\sum_{i=1}^{n} e_i \odot m_i)}{\text{FC}(\sum_{i=1}^{n} e_i)} \]  (6)

\[ t = \text{FC}(\sum_{i=1}^{n} e_i \odot m_i) \]  (7)

where FC is the fully connected layer.

C. Hierarchical Pose Refinement

1) Pose Refinement: To achieve the hierarchical refinement of the pose in the pyramid structure, both the embedding feature and the mask need to be up-sampled from coarse to fine. Inspired by [7], the set upconv layer has similar three steps as the set conv layer, except that in the set upconv layer, we use points in the dense point map as center points to aggregate sparse points. With the set upconv layer, we can get the \( l \)-layer coarse embedding feature \( CE^l \) and coarse mark \( CM^l \). The re-embedding feature \( RE^l \) is obtained by calculating the cost volume between the point cloud after warp and the original point cloud. To calculate the projected point cloud after warp, taking the \( l \)th layer as an example, two frames of the projected point cloud \( PM_1 \) and \( PM_2 \) are converted to the 3-D point cloud form \( PC_1 \) and \( PC_2 \). \( PC_1^l \) is warped by the pose \( T^{l+1} \) estimated from the \((l+1)\)th layer to get \( PC_{1,\text{warp}}^l \), where \( T^{l+1} \) is the corresponding transformation matrix of the estimated quaternions \( q^{l+1} \) and translation vector \( t^{l+1} \):

\[ PC_{1,\text{warp}}^l = T^{l+1}PC_1^l. \]  (8)
TABLE I
RESULT ON THE KITTI DATASET. USE SEQUENCES 00–08 AS THE TRAINING SET AND SEQUENCES 09 AND 10 AS THE TESTING SET. \( t_{rel} \) DENOTES THE TRANSLATION ERROR (%) AND \( r_{rel} \) DENOTES THE ROTATION ERROR (DEG/M). THE BEST RESULTS ARE ShOWN IN BOLD

| Methods                      | Seq.09 | Seq.10 | Mean |
|------------------------------|--------|--------|------|
|                              | \( t_{rel} \) | \( r_{rel} \) | \( t_{rel} \) | \( r_{rel} \) |
| Geometry-based VO            |        |        |      |
| ORB SLAM2 (w/ LC) [9]        | 2.88   | 0.0025 | 3.30 | 0.0030 | 3.09 | 0.0028 |
| ORB SLAM2 (w/o LC) [9]       | 9.30   | 0.0026 | 2.57 | 0.0032 | 5.93 | 0.0016 |
| SOFT2 [32]                   | 0.75   | 0.0022 | 0.74 | 0.0022 | 0.75 | 0.0022 |
| Learning + Geometry VO       |        |        |      |
| vid2depth [18]               | --     | --     | 21.54 | 0.125 | 21.54 | 0.125 |
| DF-Vo [33]                   | 2.07   | 0.0023 | 2.06 | 0.0036 | 2.07 | 0.0030 |
| Zhu et al. [34]              | 4.66   | 0.017  | 6.30 | 0.016 | 5.48 | 0.016 |
| DeepMatchVo [35]             | 9.91   | 0.038  | 12.18 | 0.059 | 11.05 | 0.049 |
| Gordon et al. [36]           | 3.10   | --     | 5.40 | --     | 4.25 | --     |
| pRGBD-Refined [19]           | 4.20   | 0.0041 | 4.40 | 0.0016 | 4.30 | 0.0013 |
| Jin et al. [37]              | 3.67   | 0.0038 | 2.86 | 0.0054 | 3.27 | 0.0046 |
| Cao et al. [38]              | 3.32   | 0.0026 | 2.96 | 0.0029 | 3.14 | 0.0036 |
| self-Supervised              |        |        |      |
| SfMLearner [3]               | 8.28   | 0.031  | 12.20 | 0.030 | 10.24 | 0.031 |
| GeoNet [14]                  | 28.72  | 0.098  | 23.90 | 0.090 | 26.31 | 0.094 |
| UnDeepVO [21]                | 7.01   | 0.036  | 10.63 | 0.047 | 8.82 | 0.042 |
| depth-vo-feat [39]           | 11.92  | 0.036  | 12.62 | 0.034 | 12.27 | 0.035 |
| SAVO [40]                    | 9.52   | 0.036  | 6.45  | 0.024 | 7.99 | 0.030 |
| Monodepth2-M [41]            | 11.47  | 0.032  | 7.73  | 0.034 | 9.6  | 0.033 |
| SC-SfMLearner [4]            | 11.2   | 0.034  | 10.1  | 0.050 | 10.65 | 0.042 |
| Wang et al. [20]             | 9.88   | 0.034  | 12.24 | 0.052 | 11.06 | 0.043 |
| Li et al. [42]               | 5.89   | 0.033  | 4.79  | 0.0083 | 5.34 | 0.021 |
| CC [43]                      | 6.92   | 0.018  | 7.97  | 0.031 | 7.45 | 0.025 |
| Zou et al. [10]              | 3.49   | 0.010  | 5.81  | 0.018 | 6.45 | 0.014 |
| CM-Vo [44]                   | 9.69   | 0.034  | 10.01 | 0.049 | 9.85 | 0.042 |
| DOC+ [17]                    | 2.02   | 0.0061 | 2.29  | 0.011 | 2.16 | 0.0086 |
| Xu et al. [45]               | 6.30   | 0.025  | 7.05  | 0.029 | 6.68 | 0.027 |
| Dai et al. [46]              | 3.24   | 0.0087 | 1.03  | 0.0065 | 2.14 | 0.0076 |
| Supervised                   |        |        |      |
| DeepV2D [13]                 | 8.71   | 0.037  | 12.81 | 0.083 | 10.76 | 0.037 |
| Ours                         | 0.98   | 0.0043 | 1.66  | 0.0079 | 1.32 | 0.0061 |
| Ours-IGEVE [13]              | 0.90   | 0.0036 | 1.16  | 0.0070 | 0.77 | 0.0053 |

For the generated PC\(_1\), warp, to get the point cloud in the projected form, it is necessary to reproject it and obtain the new coordinates of each point cloud in PC\(_1\) on the 2-D plane. Unlike [7], which uses cylindrical projection to get the point cloud projection after warp, we can project the point cloud directly to the 2-D image plane. It is worth noting that not all points will be reprojected onto the image plane due to errors in the regressed pose. Those points that are projected outside the image plane will be discarded. Using coarse embedding feature CE\(_l\), coarse embedding mask CM\(_l\), re-embedding feature RE\(_l\), and point feature F\(_l\)\(_{fused}\), l-layer embedding feature E\(_l\) and embedding mask M\(_l\) can be obtained by MLP

\[
E_l = MLP(CE_l \oplus RE_l \oplus F_{l_{fused}}) \\
M_l = \text{softmax}(MLP(E_l \oplus CM_l \oplus F_{l_{fused}})).
\]

Finally, using the embedding feature and the embedding mask of the l-layer, the residual poses \( q_{res} \) and \( t_{res} \) can be calculated by (6) and (7). The pose of the l-layer is

\[
q_l = q_{res} \cdot q_{l+1}^{-1} \quad [0, t_l] = q_{res} \cdot (0, t_{l+1}) \cdot (q_{res})^{-1} + [0, t_{res}].
\]

D. Loss

In this article, the ground-truth translation \( t_{gt} \) and quaternion \( q_{gt} \) of the camera pose is used as a supervisory signal to train the odometry network. As in [50] and [51], the translation \( t^l \) and quaternion \( q^l \) output by the network should occupy different positions in the loss function due to their different scales and units. For this reason, two learnable parameters \( s_x \) and \( s_q \) are introduced. Here, the loss function of the l-th layer is

\[
\ell_l = \|t_{gt} - t^l\| \exp(-s_x) + s_x + \|q_{gt} - q^l\|_2 \exp(-s_q) + s_q
\]

where \( \|\cdot\| \) and \( \|\cdot\|_2 \) represent \( \ell_1 \)-norm and \( \ell_2 \)-norm, respectively. Then, for the four layers of the pyramid, the total training loss is

\[
\ell = \sum_{l=1}^{4} \ell_l
\]
To quantitatively show the effect of our method and make it easier to compare with other methods, the evaluation standard proposed by the KITTI odometry benchmark is used. For each test sequence, the average translational and rotational errors on all possible subsequences in the length of 100, 200, \ldots, 800 m are evaluated, and errors are measured in percent (for translation) and degrees per meter (for rotation).

### B. Network Details

Our model is trained and supervised on the KITTI dataset [8]. For depth network, the pretrained model provided by GA-Net [28] is used. During the training process, the depth network will not be trained. The pseudo-LiDAR point cloud generated by the depth map will be saved to reduce unnecessary training time. A Quadro RTX 3090 is used to train our complete model. The Adam optimizer is adopted with \( \beta_1 = 0.9 \) and \( \beta_2 = 0.999 \). The initial learning rate is set to 0.001, which decays to 0.00001 during training. Our decay step is set to attenuate the learning rate by 0.7 every 13 epochs.

### V. Experimental Results

In this section, the quantitative and qualitative results of the visual odometry network performance are shown in tables and pose visualizations.

#### A. Performance Evaluation

In the existing related methods, there are mainly two modes to divide the training/testing sets. To make a fair comparison with the related methods, our model is trained/tested in these two modes, respectively.

1) **Using Sequences 00-08/09 and 10 as Training/Testing Sets**: Quantitative results are summarized in Table II. ORB-SLAM2 [9] with loop closing (w/ LC)/without loop closing (w/o LC) is a classical feature point-based geometric method. vid2depth [18], DeepMatchVO [35], SfMLearner [3], GeoNet [14], UnDeepVO [21], depth-vo-feat [39], Monodepth2-M [41], SC-SfMLearner [4], and CC [43] are all combined depth estimation with odometry, but these methods only use the loss to connect the depth network with the pose network. In the methods Gordon et al. [36], pRGBD-Refined [19], SAVO [40], Wang et al. [20], Zhu et al. [34], Li et al. [42], and DeepV2D [52], depth information is used in the pose network in different ways to optimize the pose. But unlike our approach where depth is directly used to convert a 2-D image to a 3-D point cloud, pRGBD-Refined [19] concatenates the regressed depth map with the image into pseudo-RGB-D to estimate the pose. While these methods all exploit depth information in images, our method acts directly on the 3-D coordinates, where scale information between points is revealed. So, points that are far apart in real 3-D space are not put together for feature aggregation. Previous methods do not exploit the explicit 3-D structure, and depth is used in an indirect form. The comparison results also show the superiority of our method.

2) **Using Sequences 00, 02, 08, 09/03, 04, 05, 06, 07, and 10 as Training/Testing Sets**: Quantitative results are summarized in Table II. DeepVO [1] proposes a visual odometry framework based on deep learning, which realizes the end-to-end estimation of the camera pose. On the basis of DeepVO [1], the FC layer and SE(3) composition layer are added to ESP-VO [2] to directly estimate a range of poses and uncertainties. Both DeepVO [1] and ESP-VO [2] do not provide explicit depth information to the odometry network, and the pose network needs to learn indirectly from images, which is much more complicated than directly learning the input 3-D information to estimate pose. Yang et al. [49] combined the visual odometry and IMU. Here, we have chosen to compare the results of 100% utilization of a visual encoder, which is the same as most visual odometry methods.

#### B. Ablation Study

Table III shows the average running time of models in KITTI sequence 00. The operation of denoising and sampling...
Fig. 5. Two-dimensional trajectory results on KITTI sequences 09 and 10 with ground truth. Trajectory results of depth-vo-feat, ORB-SLAM2 (w/ LC), and ours on KITTI. (a) Two-dimensional trajectory plots of Seq.09. (b) Two-dimensional trajectory plots of Seq.10.

Fig. 6. Three-dimensional trajectory results on KITTI sequences 09 and 10 with ground truth. Trajectory results of depth-vo-feat, ORB-SLAM2 (w/ LC), and ours on KITTI. (a) Three-dimensional trajectory plots of Seq.09. (b) Three-dimensional trajectory plots of Seq.10.

Fig. 7. Translational and rotational error on KITTI sequences. Average translational and rotational error on KITTI sequence 09 on all possible subsequences in the length of 100, 200, ..., 800 m.

8192 point clouds in a pseudo-LiDAR point cloud takes a lot of time. The projection-aware approach avoids large amount of time required to sample the point clouds and the rich point cloud information can be fully utilized. Since the pseudo-LiDAR point cloud generation will take 2.5 s, we consider saving the pseudo-LiDAR point cloud from the depth estimation and enabling the pose estimator to learn the pose from the saved point cloud.

To verify the effectiveness of our proposed method, ablation experiments in Table IV are performed in the same experimental setting. The projection-aware approach has a better performance compared to the network [6] that uses 8192 points as input. The projection-aware network with the addition of image texture information also shows better performance. This shows that richer features can provide the network with a more comprehensive scene understanding. As shown in Table IV, all our proposed methods improve the performance of the network.

In addition, by using a depth network IGEV [13] with higher accuracy than GA-Net [28], comparison with Ours and Ours-IGEV in Table I reveals that higher depth estimation results lead to better odometry results.

VI. DISCUSSION AND CONCLUSION

In this article, a projection-aware pseudo-LiDAR-based visual odometry network is presented. Considering that conventional visual odometry does not make full use of the geometric structure information in images, pseudo-LiDAR is introduced. On this basis, the projection-aware approach is proposed to efficiently utilize the rich point cloud information in pseudo-LiDAR. Also, the texture features of the images are utilized to enhance the features of the pseudo-LiDAR point cloud to help the odometry network better understand the environment. The results of the KITTI dataset, as well as the ablation study, show the effectiveness of our method.

REFERENCES

[1] S. Wang, R. Clark, H. Wen, and N. Trigoni, “DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks,” in Proc. ICRA, May 2017, pp. 2043–2050.
[2] S. Wang, R. Clark, H. Wen, and N. Trigoni, “End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks,” Int. J. Robot. Res., vol. 37, nos. 4–5, pp. 513–542, Apr. 2018.
[3] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, “Unsupervised learning of depth and ego-motion from video,” in Proc. CVPR, Jul. 2017, pp. 1851–1858.
[4] J. Bian et al., “Unsupervised scale-consistent depth and ego-motion learning from monocular video,” in Proc. NIPS, vol. 32, 2019, pp. 35–45.
[5] X. Ban, H. Wang, T. Chen, Y. Wang, and Y. Xiao, “Monocular visual odometry based on depth and optical flow using deep learning,” IEEE Trans. Instrum. Meas., vol. 70, pp. 1–19, 2021.
[6] G. Wang, X. Wu, Z. Liu, and H. Wang, “PWCL-Net: Deep LiDAR odometry in 3D point clouds using hierarchical embedding mask optimization,” in Proc. CVPR, Jun. 2021, pp. 15910–15919.
[7] G. Wang, X. Wu, S. Jiang, Z. Liu, and H. Wang, “Efficient 3D deep LiDAR odometry,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 45, no. 5, pp. 5749–5765, May 2023.
[8] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” Int. J. Robot. Res., vol. 32, no. 11, pp. 1231–1237, Sep. 2013.
Huiying Deng received the B.S. degree from the Department of Automation, Astronautics Jincheng College, Nanjing University of Aeronautics, Nanjing, China, in 2018, and the M.S. degree in electronic information from the School of Information and Control Engineering, China University of Mining and Technology, Xuzhou, China, in 2022. Her current research interests include visual odometry and 3-D LiDAR odometry.

Chaokang Jiang received the B.S. degree from the Department of Engineering, Jingdezhen Ceramic Institute College of Technology and Art, Jingdezhen, China, in 2020. He is currently pursuing the M.E. degree in control science and engineering with the China University of Mining and Technology, Xuzhou, China. His current research interests include SLAM and computer vision.

Zhiheng Feng is currently pursuing the B.S. degree with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China. His research interests include SLAM and computer vision.

Xinrui Wu received the B.S. degree from the Department of Automation, Shanghai Jiao Tong University, Shanghai, China, in 2021, where he is currently pursuing the M.S. degree in control science and engineering. His research interests include SLAM and computer vision.

Guangming Wang (Member, IEEE) received the B.S. degree from the Department of Automation, Central South University, Changsha, China, in 2018. He is currently pursuing the Ph.D. degree in control science and engineering with Shanghai Jiao Tong University, Shanghai, China. His current research interests include SLAM and computer vision, in particular, visual odometry and 3-D LiDAR odometry.

Hesheng Wang (Senior Member, IEEE) received the B.Eng. degree in electrical engineering from the Harbin Institute of Technology, Harbin, China, in 2002, and the M.Phil. and Ph.D. degrees in automation and computer-aided engineering from The Chinese University of Hong Kong, Hong Kong, in 2004 and 2007, respectively. He is currently a Professor with the Department of Automation, Shanghai Jiao Tong University, Shanghai, China. His current research interests include visual servoing, service robots, computer vision, and autonomous driving. Dr. Wang was the General Chair of IEEE ROBIO 2022 and the IEEE RCAR 2016, and the Program Chair of the IEEE ROBIO 2014 and the IEEE/ASME AIM 2019. He is the General Chair of the IEEE/RSJ IROS 2025. He is an Associate Editor of IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, IEEE ROBOTICS AND AUTOMATION LETTERS, Assembly Automation, and International Journal of Humanoid Robotics, a Technical Editor of the IEEE/ASME TRANSACTIONS ON MECHATRONICS, and an Editor of the Conference Editorial Board (CEB) of IEEE Robotics and Automation Society. He served as an Associate Editor for IEEE TRANSACTIONS ON ROBOTICS, from 2015 to 2019.