2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

CalibFormer: A Transformer-based Automatic LiDAR-Camera Calibration Network

Yuxuan Xiao[1], Yao Li[1], Chengzhen Meng[1], Xingchen Li[1], Jianmin Ji[1] and Yanyong Zhang[2]

***Abstract*— The fusion of LiDARs and cameras has been increasingly adopted in autonomous driving for perception tasks. The performance of such fusion-based algorithms largely depends on the accuracy of sensor calibration, which is challenging due to the difficulty of identifying common features across different data modalities. Previously, many calibration methods involved specific targets and/or manual intervention, which has proven to be cumbersome and costly. Learning-based online calibration methods have been proposed, but their performance is barely satisfactory in most cases. These methods usually suffer from issues such as sparse feature maps, unreliable cross-modality association, inaccurate calibration parameter regression, etc. In this paper, to address these issues, we propose CalibFormer, an end-to-end network for automatic LiDAR-camera calibration. We aggregate multiple layers of camera and LiDAR image features to achieve high-resolution representations. A multi-head correlation module is utilized to identify correlations between features more accurately. Lastly, we employ transformer architectures to estimate accurate calibration parameters from the correlation information. Our method achieved a mean translation error of 0.8751cm and a mean rotation error of 0.0562° on the KITTI dataset, surpassing existing state-of-the-art methods and demonstrating strong robustness, accuracy, and generalization capabilities.**

I. Introduction

Nowadays, LiDARs and cameras have played a critical role in robotic systems such as autonomous vehicles. Cameras capture high-resolution images with detailed color and texture information, while LiDARs provide precise 3D point cloud representations of the environment. The fusion of LiDARs and cameras has been widely adopted in many tasks, such as 3D object detection [1–3] and SLAM [4, 5], achieving better performance than relying on a single modality. However, a crucial prerequisite for successful multi-sensor fusion is extrinsic calibration, which involves determining the 6-degree-of-freedom (6-DoF) transformation between the coordinate systems of the sensors.

The main challenge in calibration problems is to effectively identify and correlate common features across different data modalities. Many existing calibration methods [6–8] rely on calibration targets, such as checkerboards or boards with specific patterns. They detect, extract, and match pairwise calibration targets between 2D images and 3D point clouds, which transforms the calibration problem into an optimization problem. These methods typically achieve satisfactory results with predefined external targets. However, calibration parameters can vary irregularly due to factors such as temperature changes or disturbances during vehicle movements. Even a slight relative position offset between LiDAR and camera necessitates a recalibration to correct the extrinsic parameter drift. Consequently, timely target-based methods become prohibitively expensive, if at all possible, which renders targetless calibration approaches necessary.

Some works [9–11] have attempted to obtain the transformation matrix using targetless methods, but they require hand-crafted features and may not work well in different environments. Recently, deep learning techniques have been widely used and demonstrated the superiority of automatic feature engineering. For example, several methods [12–14] use deep learning models to estimate the 6-DoF transformation between camera and LiDAR coordinates. Most deep learning-based methods utilize RGB images and depth images obtained through point cloud projection — often miscalibrated — as inputs. The processing pipeline primarily involves three steps: feature extraction, feature matching, and calibration parameter regression. However, deep learning-based calibration methods face several important design issues. For example, the balance between calibration accuracy and computation efficiency is hard to achieve. It is also challenging to accurately correlate corresponding features from different modalities due to the disparate physical characteristics and operational principles of sensors, yielding data of varying dimensions, qualities, and types. Furthermore, not all correlated features contribute equally to calibration, making it imperative to extract feature correlations with higher contributions.

In this paper, we present CalibFormer, an automatic calibration method for LiDARs and cameras to address these issues. Firstly, calibration usually requires precise alignment between modalities, which in turn demands precise representations of sensor data contributions. Therefore, in the feature extraction phase, we upsample and aggregate multi-layer features [15] from RGB and LiDAR data, obtaining fine-grained representations of the features. In the matching phase, we apply a multi-head correlation module to calculate multi-dimensional correlation representations. In order to capture sufficient correlations from unaligned

*The work is partially supported by the National Natural Science Foundation of China (No.62332016) and Anhui Province Development and Reform Commission 2021 New Energy and Intelligent Connected Vehicle Innovation Project.

[1]Yuxuan Xiao, Yao Li, Chengzhen Meng, Xingchen Li, and Jianmin Ji are with School of Computer Science and Technology, University of Science and Technology of China, Hefei, China. (e-mail: {xiaoyx, zkdly, czmeng, starlet}@mail.ustc.edu.cn, jianmin@ustc.edu.cn)

[2]Yanyong Zhang is the corresponding author, with School of Computer Science and Technology, University of Science and Technology of China, Hefei, China, and also with Institute of Artificial Intelligence, Hefei Comprehensive National Science Center, Hefei, China. (e-mail: yanyongz@ustc.edu.cn)

features, we treat feature correlation as a learnable implicit representation and employ a multi-head computation approach. Subsequently, leveraging the computed correlation features, we employ a Swin Transformer [16] encoder and a Transformer [17] decoder to process the correlation feature effectively. Finally, a feed-forward network is utilized to regress the translation and rotation parameters separately. In summary, our contributions can be summarized as follows:

- CalibFormer is an end-to-end network designed for LiDAR-camera calibration. We extract fine-grained feature maps to achieve precise correlations and take a trade-off between computation and performance.
- We apply a multi-head correlation module to calculate correspondences between misaligned features across different dimensions. Then we utilize the transformer architecture to extract and leverage correlation features with higher contributions.
- The experiment results demonstrate that our approach achieves a translation error of 0.8751cm and a rotation error of 0.0562°, surpassing other deep learning-based methods and exhibiting robust generalization capabilities. Ablation experiments also validate the effectiveness of various modules.

II. Related Works

Existing extrinsic calibration methods can be broadly categorized into target-based and targetless methods. The primary difference between them is the use of specific targets during the calibration process. Target-based methods require artificial calibration objects that offer explicit geometric features in both modalities. In contrast, targetless methods do not rely on any designated targets but instead utilize information from the surrounding environment.

A. Target-based Methods

Zhang and Pless [18] first proposed the target-based extrinsic calibration method using a checkerboard. The algorithm detects the grid corner to estimate its pose relative to the checkerboard in the camera image. It then calculates the extrinsic parameters between the laser scanner and the camera. The transformation is formulated into a nonlinear optimization problem. Geiger et al. [6] further applies a similar optimization problem to calibrate the camera and the 3D LiDAR. Other targets, such as trihedral [7] or spherical targets [8, 19], are also used for calibration besides the checkerboard. Beltrán et al. [20] uses a rectangular plate with 4 holes as the target and affixes 4 ArUco tags [21]. They detect and match the center points of these 4 holes and calculate the transformation between the sensors. In general, these methods extract geometric features, such as points, edges, or planes, from images and point clouds according to the target and then match them. The calibration problem is then formulated into an optimization problem. Despite being simple and effective, target-based methods are still time-consuming and laborious. Moreover, these methods cannot be used everywhere due to the limitations of the targets.

B. Targetless Methods

Li et al. [22] divided targetless calibration methods into four categories: information theory-based, feature-based, ego-motion-based, and learning-based.

Pandey et al. [9] proposed a mutual information-based algorithm that uses the correspondence between the intensity of point clouds and the grayscale of images. Taylor and Nieto [10] utilize the ego-motion of sensors mounted on the moving vehicle to estimate extrinsic parameters. Levinson and Thrun [11] and Yuan et al. [23] respectively extract depth-discontinuous and depth-continuous edge features and then match them to minimize the objective function.

Regnet [12] was the first to adopt a deep learning approach. It extracts and matches features using a network before regressing the calibration parameters. CalibNet [24] incorporates geometric information by introducing a 3D spatial transformer layer into the model. CalibRCNN [13] combines CNN with LSTM and adds pose constraints between consecutive frames to improve the calibration accuracy. LCCNet [14] utilizes the cost volume to compute the correlation between features from different sensors. Moreover, in sensor fusion perception tasks such as object detection [1, 25, 26], the transformer's cross-attention mechanism is used to achieve soft association between different modalities. Nevertheless, the application of the transformer in calibration methods is rare.

In contrast to these methods, our approach employs a deep layer aggregation module to obtain high-resolution feature maps from the LiDAR and camera. Furthermore, we incorporate a multi-head correlation module to compute inter-sensor correlations in a more fine-grained manner. We also utilize a transformer architecture to process correlations and estimate calibration parameters effectively.

III. Methods

Our network takes a pair of misaligned camera images and LiDAR point cloud as inputs and outputs the deviation $\mathbf{T}_{pred}$ of the initial calibration parameters $\mathbf{T}_{init}$ from the ground truth $\mathbf{T}_{LC}$. It first extracts fine-grained features from both modalities separately and calculates their correlations through a multi-head correlation module. Finally, the network regresses the deviations of the calibration parameters. The workflow of our proposed network is shown in Fig. 1.

A. Input Data Preprocessing

We first project the LiDAR point cloud onto the image plane to obtain a sparse depth map. For each 3D point $p_i^L = [x_i \ y_i \ z_i]^T \in \mathbb{R}^3$ in the point cloud, given an initial extrinsic parameter $\mathbf{T}_{init}$ and camera intrinsic matrix $\mathbf{K}$, we can project it onto a 2D coordinate system on the image plane, denoted as pixel $p_i^I = [u_i \ v_i]^T \in \mathbb{R}^2$. The projection process can be expressed as:

$$d_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{K}\mathbf{T}_{init} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R}_{init} & \mathbf{t}_{init} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}, \tag{1}$$

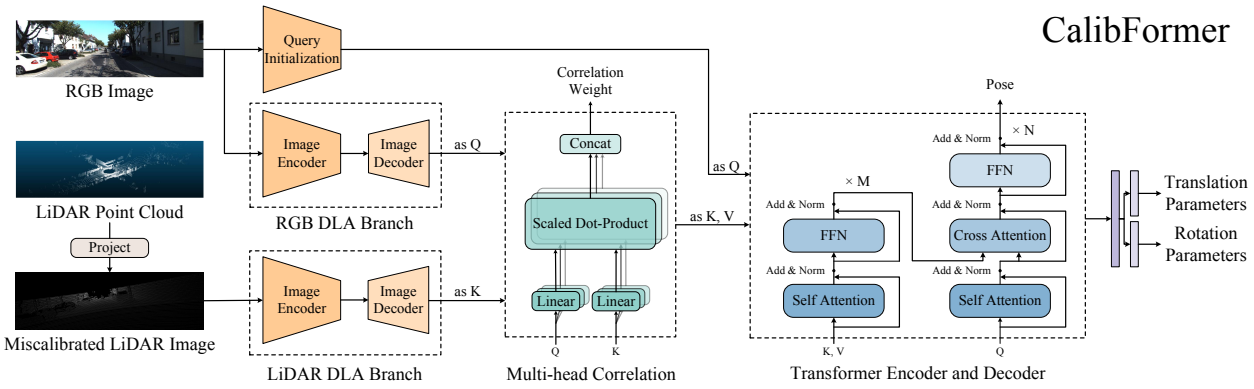

Fig. 1. The overview of our proposed method for camera and LiDAR calibration. Firstly, we project the LiDAR point cloud onto the image plane, generating a miscalibrated LiDAR image using the initial extrinsic parameter $\mathbf{T}_{init}$ and the camera matrix $\mathbf{K}$. Our network takes both camera images and LiDAR images as inputs. After extracting fine-grained features, we employ a multi-head correlation module and a transformer architecture to obtain a 6-DoF transformation $\mathbf{T}_{pred}$ representing the deviation between the initial extrinsic parameter $\mathbf{T}_{init}$ and the accurate extrinsic parameter $\mathbf{T}_{LC}$.

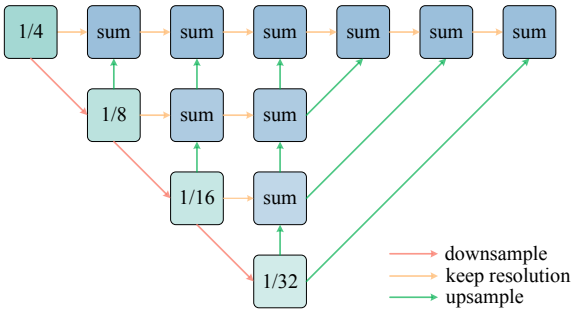

Fig. 2. Overview of deep layer aggregation. After obtaining the features generated by the backbone at different layers, these features are respectively upsampled and aggregated to obtain a high-resolution feature map.

where $\hat{p}_i^I = [u_i \ v_i \ 1]^T$ and $\hat{p}_i^L = [x_i \ y_i \ z_i \ 1]^T$ represent the homogeneous coordinates of p_i^I and p_i^L, respectively. $\mathbf{R}_{init}$ and $\mathbf{t}_{init}$ represent the initial rotation matrix and translation vector of extrinsic parameter $\mathbf{T}_{init}$. The depth of p_i^L after projection onto the image plane is represented by d_i, which is used to construct the depth map X_{dm} along with p_i^I.

To further utilize the LiDAR data, we also record the intensity I_i of each LiDAR point p_i^L during projection. Together with depth d_i, we can obtain a two-channel LiDAR image $X_{lidar} \in \mathbb{R}^{2\times H\times W}$, where H and W represent the height and width of the image.

B. Fine-grained Feature Extraction

The feature extraction network consists of two symmetric branches for extracting features from the camera image X_{cam} and LiDAR image X_{lidar}, respectively. In the camera branch, we use a pre-trained ResNet-18 [27] model to extract image features. Meanwhile, the LiDAR branch has a similar structure to the image branch, except that the number of input channels for the first convolutional layer is adjusted to two to accommodate the LiDAR image X_{lidar}.

We use an enhanced version of the Deep Layer Aggregation (DLA) [15] to fuse multi-scale features with different receptive fields, as illustrated in Fig. 2. Unlike the original DLA [28], it has more skip connections between feature maps of different scales. Additionally, to dynamically adjust the receptive field based on the image, the convolutional layers in the upsampling module of the DLA are replaced with deformable convolution layers [29]. We apply this enhanced DLA on both the camera and LiDAR branches separately, resulting in high-resolution feature maps F_{lidar} and F_{cam}, which are used for subsequent calibration.

We additionally employ a camera-guided query initialization module to extract features F_{query} from camera images. This module consists of a ResNet-18 and a global average pooling layer, employed for generating the initial pose query.

C. Multi-head Feature Matching

After separately extracting fine-grained features from the input data, a feature matching module is used to calculate the correlation between the misaligned features of the two modalities caused by the noisy initial parameter $\mathbf{T}_{init}$ during LiDAR point projection in Section III-A. To compute the correlation between features in a fine-grained manner, we apply a multi-head correlation module.

Firstly, the LiDAR feature map F_{lidar} and camera feature map F_{cam} are flattened into queries Q and keys K, both with a dimension of d_k. Then, we compute the dot product between the query Q and the key K. After that, we divide the result by $\sqrt{d_k}$ to obtain the correlation weights. The computation process can be expressed as:

$$\text{Correlation}(Q, K) = \frac{QK^T}{\sqrt{d_k}}. \tag{2}$$

The correlation weights reflect the correlation between the two modalities.

To capture different correlations between queries and keys in multiple dimensions, we use n sets of linear projections to transform the queries and keys independently. We compute the correlation weights separately for each set and then concatenate them to obtain the final output. The process is depicted below:

$$\begin{aligned} \text{MultiHead}(Q, K) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_n), \\ \text{head}_i &= \text{Correlation}(QW_i^Q, KW_i^K), \end{aligned} \tag{3}$$

where W_i^Q and W_i^K are linear projection matrices.

Given that the initial deviation of extrinsic parameters falls within a certain range, the offset of the LiDAR and camera image features is also expected to be within a

certain range. Specifically, for an image with a resolution of 1241×376 from the KITTI dataset, if the translation and rotation deviations of the calibration parameters are within 0.5m and 5° respectively, the projected point cloud deviation is approximately within 100 pixels. To leverage this characteristic, we apply a window-based approach, in which only the correlation weights within a particular window are utilized for calibration. Given a window size d, the correlation weights between the camera feature $F_{cam}(p_i)$ and the LiDAR feature $F_{lidar}(p_j)$ are only considered if $|p_i - p_j|_\infty \leq d$, where p_i and p_j are the 2D positions of camera feature map F_{cam} and LiDAR feature map F_{lidar}, respectively. Therefore, we can obtain the correlation feature map $F_{corr} \in \mathbb{R}^{((2d+1)^2 \times n) \times H \times W}$, where H and W denote the height and width of the feature map, respectively, and n denotes the number of heads.

D. Transformer-based Parameter Regression

We employ a transformer architecture to extract calibration parameters from the correlation feature map F_{corr}. For the correlation feature F_{corr}, we first increase its dimension to d_k by densely connected convolutional layers [30]. Subsequently, it is flattened and employed as input Q, K, and V for the transformer encoder. Given the high resolution of the correlation feature map, we utilize a Swin Transformer encoder [16] to mitigate computational complexity. We employ a Transformer decoder [17] to estimate calibration parameters. The query Q is derived from the initial pose query F_{query}, while the key K and value V are obtained from the encoder's output. The positions are embedded as position encoding using a multi-layer perceptron (MLP) and element-wise added to the key features. Finally, a feed-forward network (FFN) is utilized to regress the pose information, and then we obtain the translation parameters and rotation parameters separately. The translation parameter is represented by a 1×3 vector $\mathbf{t}_{pred}$, and the rotation parameter is represented by a 1×4 quaternion $\mathbf{q}_{pred}$.

E. Loss Function

To guide the network convergence, we employ the following loss function:

$$L = \lambda_t L_t + \lambda_r L_r + \lambda_p L_p, \tag{4}$$

where λ_t, λ_r and λ_p represent the weights for translation loss, rotation loss, and point cloud distance loss, respectively.

For the translation part, we used the smoothed L1 loss to represent the error:

$$L_t = \text{SmoothL1}(\mathbf{t}_{gt} - \mathbf{t}_{pred}). \tag{5}$$

Regarding the rotation component, to avoid the double-covering issue of quaternions, we use the angular distance to measure the difference between the quaternions. For a unit quaternion $\mathbf{q}$, using $\Re(\mathbf{q})$ and $\Im(\mathbf{q})$ to respectively represent its real part and imaginary part, the rotation loss is defined as follows:

$$L_r = D_a(\mathbf{q}_{gt}, \mathbf{q}_{pred}) = 2 \arctan(\frac{\|\Im(\mathbf{q}_{gt} \times \mathbf{q}_{pred}^{-1})\|}{\Re(\mathbf{q}_{gt} \times \mathbf{q}_{pred}^{-1})}). \tag{6}$$

The point cloud distance loss describes the distance between corresponding points in the point clouds before and after the extrinsic transformation:

$$L_p = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{T}_{gt}^{-1} \mathbf{T}_{pred} \mathbf{p}_i^L - \mathbf{p}_i^L\|, \tag{7}$$

where N donates the number of points in this point cloud.

F. Calibration Inference

We denote the rotation component as a rotation matrix $\mathbf{R}_{pred}$ converted from the network prediction quaternion value $\mathbf{q}_{pred}$. By concatenating the rotation matrix $\mathbf{R}_{pred}$ with the translation vector $\mathbf{t}_{pred}$, we obtain the predicted error:

$$\mathbf{T}_{pred} = \begin{bmatrix} \mathbf{R}_{pred} & \mathbf{t}_{pred} \\ \mathbf{0} & 1 \end{bmatrix}. \tag{8}$$

Given the initial extrinsic parameter $\mathbf{T}_{init}$ and the predicted error $\mathbf{T}_{pred}$, the calibration parameters can be obtained as follows:

$$\widehat{\mathbf{T}}_{LC} = \mathbf{T}_{pred}^{-1} \mathbf{T}_{init}. \tag{9}$$

IV. Experiments

A. Dataset Preparation

We evaluate our method on KITTI odometry dataset [31], which consists of 22 sequences from different scenes. We use sequences 01 to 21 as the training and validation set, and sequence 00 as the test set. To acquire a sufficient amount of training data, we introduce uniformly distributed random deviations $\Delta\mathbf{T}$ within a certain range to the extrinsic parameters of the data. We obtain the initial extrinsic parameter $\mathbf{T}_{init} = \Delta\mathbf{T} \times \mathbf{T}_{LC}$, and use it along with the point cloud and image as input. The ground truth is denoted as $\mathbf{T}_{gt} = \Delta\mathbf{T}$. A significant amount of training data can be obtained by randomly generating deviations.

B. Implementing Details

Training details: The original resolution of the image is padded to 1280×384 and subsequently resized to 512×256 as the input for the model. The upsampling rate for feature maps is set to 4. The window size in the multi-head correlation module is set to 4. Furthermore, the Swin Transformer encoder is configured with 2 layers, while the Transformer decoder comprises 6 layers. We train the network using the Adam optimizer [32] with a learning rate of $5e^{-4}$ for 500 epochs and a batch size of 256. Latency is measured on an NVIDIA RTX 3060 GPU.

Evaluation Metrics: The calibration results are evaluated by translation and rotation parameters. For the translation component, we separately record mean absolute errors in the X, Y, and Z directions. Regarding the rotation component, although it can be represented using quaternions, we opt to convert it into Euler angles to provide a more intuitive assessment of rotation error. Similarly, we separately record mean absolute errors of roll, pitch, and yaw.

TABLE I

COMPARISON OF DIFFERENT METHODS ON THE KITTI ODOMETRY DATASET. THE UPPER HALF IS THE RESULT OF A DEVIATION OF $(\pm 0.25\text{m}, \pm 10°)$ AND THE LOWER HALF IS THE RESULT OF A DEVIATION OF $(\pm 0.5\text{m}, \pm 5°)$.

Deviation	Method	Translation(cm) ↓				Rotation(°) ↓			
		Mean	X	Y	Z	Mean	Roll	Pitch	Yaw
$(\pm 0.25\text{m}, \pm 10°)$[1]	CalibRCNN [13]	5.3	6.2	4.3	5.4	0.428	0.199	0.64	0.446
	CALNet [33]	3.03	3.65	1.63	3.80	0.20	0.10	0.38	0.12
	PSNet [34]	3.1	3.8	2.8	2.6	0.15	**0.06**	0.26	0.12
	Ours	**1.1877**	**1.1006**	**0.9015**	**1.5611**	**0.1406**	0.0764	**0.2588**	**0.0865**
$(\pm 0.5\text{m}, \pm 5°)$[2]	LCCNet [14]	1.6737	1.6695	1.3192	2.0325	0.1576	0.0556	0.3080	0.1091
	CalibDepth [35]	0.9188	0.9971	**0.5655**	**1.1938**	0.1633	0.0411	**0.0605**	0.3883
	Ours	**0.8751**	**0.7594**	0.6027	1.2633	**0.0562**	**0.0249**	0.1041	**0.0395**

[1] Following CalibRCNN, we evaluate the method's performance under a deviation of $(\pm 0.25\text{m}, \pm 10°)$.
[2] Following LCCNet, we evaluate the method's performance under a deviation of $(\pm 0.5\text{m}, \pm 5°)$.

TABLE II

ABLATION EXPERIMENTS ON KITTI ODOMETRY DATASETS.

Network Architecture	Translation(cm) ↓				Rotation(°) ↓				Latency(ms) ↓
	Mean	X	Y	Z	Mean	Roll	Pitch	Yaw	
w/ all modules	**0.8751**	**0.7594**	**0.6027**	**1.2633**	**0.0562**	**0.0249**	**0.1041**	**0.0395**	27.79
w/o multi-head correlation	1.0970	0.9300	0.9408	1.4203	0.0691	0.0332	0.1245	0.0496	26.21
w/o swin transformer encoder	1.0809	0.7727	0.8980	1.5720	0.0742	0.0383	0.1375	0.0414	26.35
w/o transformer architecture	1.2366	1.0320	0.9645	1.7133	0.0750	0.0426	0.1258	0.0566	20.79
w/o upsampling	1.1617	0.8904	0.7955	1.7991	0.0985	0.0350	0.2054	0.0550	**13.15**

TABLE III

COMPARISON OF DIFFERENT UPSAMPLING RATES.

Upsampling Rate	Translation(cm) ↓				Rotation(°) ↓				Latency(ms) ↓
	Mean	X	Y	Z	Mean	Roll	Pitch	Yaw	
1×	1.1617	0.8904	0.7955	1.7991	0.0985	0.0350	0.2054	0.0550	**13.15**
2×	1.0925	0.8435	0.7813	1.6526	0.0858	0.0284	0.1781	0.0509	19.66
4×*	0.8751	0.7594	**0.6027**	1.2633	**0.0562**	0.0249	**0.1041**	**0.0395**	27.79
8×	**0.8409**	**0.6839**	0.6449	**1.1939**	0.0576	**0.0242**	0.1050	0.0437	39.32

* denotes the configuration used in our network.

C. Quantitative Results

To compare performance with different methods, we follow their experimental setups, introducing two sets of initial deviations, $(\pm 0.5\text{m}, \pm 5°)$ and $(\pm 0.25\text{m}, \pm 10°)$, and evaluate the network's performance for each. The results, as presented in Table I, indicate that for both initial deviation settings, our proposed method outperforms other approaches.

Initial deviation $(\pm 0.25\text{m}, \pm 10°)$**:** The mean translation error of our method is 1.1877cm, and the mean rotation error is $0.1406°$. Compared to PSNet [34], our network shows a 61.7% improvement in the translation performance and a 6.3% improvement in the rotation performance.

Initial deviation $(\pm 0.5\text{m}, \pm 5°)$**:** Our method exhibits a mean translation error of 0.8751cm and a mean rotation error of $0.0562°$. Our network outperforms CalibDepth [35] by 4.7% in terms of translation performance and by 65.6% in terms of rotation performance.

D. Ablation Studies

In this section, we compare the impact of several network architectures on calibration performance. We perform a set of ablation experiments to show the effect of each component. The performance is evaluated under a deviation of $(\pm 0.5\text{m}, \pm 5°)$. The results of the ablation experiment are shown in Table II.

Multi-head Correlation: To investigate the influence of correlation computation methods on calibration results, we compare the multi-head correlation module with direct inner product computation. As shown in Table II, the multi-head correlation module yields a 20.2% improvement in translation performance and a 18.7% improvement in rotation performance, indicating its ability to better correlate corresponding features. Our method achieved greater performance gains with only a minor increase in computational overhead.

Transformer Architecture: We investigate the impact of using a transformer architecture to process correlation features on calibration performance. As a comparison, we substitute a fully connected layer for the transformer architecture. As shown in Table II, our approach yields a moderate increase in latency, resulting in a 29.2% improvement in translation performance and a 25.1% improvement in rotation performance. Furthermore, if only the Transformer decoder is utilized without the Swin Transformer encoder, the calibration results deteriorate. This outcome underscores the contribution of both the transformer encoder and decoder in locating and leveraging more valuable correlation features.

Upsampling: We compare the impact of aggregating the features of different layers, i.e., different upsampling rates. As shown in Table III, we observe that using a higher upsam-

(a) (b) (c)

Fig. 3. Examples of calibration results for different scenes on the KITTI dataset. (a) represents the projection of miscalibrated point clouds onto the image plane. (b) shows the projection result of the point cloud using the network's predicted extrinsic parameters, and (c) represents the corresponding ground truth result.

TABLE IV

CALIBRATION RESULTS ON UNFAMILIAR DATASETS AND COMPARISON WITH OTHER METHODS.

Method	Translation(cm) ↓				Rotation(°) ↓			
	Mean	X	Y	Z	Mean	Roll	Pitch	Yaw
LCCNet [14]	2.4896	1.9233	2.4894	3.0561	0.1937	0.0868	0.3647	0.1297
CalibDepth [35]	2.5413	3.4261	1.7634	2.4345	0.2818	0.0689	**0.1546**	0.6220
Ours	**1.1037**	**1.0382**	**0.7043**	**1.5685**	**0.0805**	**0.0380**	0.1556	**0.0479**

pling rate leads to better results. This is because a higher-resolution output feature map allows for a more detailed representation of the input data, potentially enabling the network to capture more fine-grained features and patterns. However, it is important to consider that using a higher upsampling rate also increases the computational complexity and memory requirements of the network, involving there may be a trade-off between performance and efficiency. As the upsampling rate increases to 8, the performance improvement becomes marginal compared to that of 4, while the computational overhead is much higher. Hence, a $4\times$ upsampling rate is a more appropriate choice.

E. Qualitative Results

Our network is capable of achieving accurate calibration results under varying scenes and different initial miscalibrations. Fig. 3 illustrates some of the visualized calibration results. We observe that even if initial extrinsic parameters exhibit errors in the direction of all six axes, our network is still able to align images and point clouds and generate results that are close to the ground truth.

F. Generalization Validation

The KITTI odometry dataset primarily consists of the "2011_09_30" and "2011_10_03" sequences from the KITTI raw dataset, along with a small part of the "2011_09_26" sequence. To evaluate our model's generalization capability, we train the network using the KITTI odometry dataset and subsequently evaluate its performance on the "2011_09_26" sequence of KITTI raw dataset, which includes unfamiliar scenes. Similarly, the initial deviation is set to $(\pm 0.5\mathrm{m}, \pm 5°)$. The results shown in Table IV indicate that due to variations in the scene, the performance of our network on the KITTI raw dataset is worse compared to the results on the KITTI odometry dataset. Nevertheless, it still achieves a mean translation error of $1.1037\mathrm{cm}$ and a mean rotation error of $0.0805°$. In comparison to LCCNet [14], our method demonstrates a 55.7% improvement in translation performance and a 58.4% improvement in rotation performance. The test results on the KITTI raw dataset underscore the robustness of our network, demonstrating consistent and strong performance across diverse scenes.

V. CONCLUSION

In this paper, we propose an end-to-end calibration network for estimating the 6-DoF rigid body transformation between the LiDAR and the camera, which are important sensor combinations in autonomous driving systems for perception. Our network consists of three main parts: feature extraction module, feature matching module, and transformer regression module. The fine-grained feature extraction module employs DLA to aggregate multi-layer features to get a high-resolution feature representation. In the multi-head feature matching module, we use a multi-head correlation module to calculate the correlation between two modalities in a fine-grained manner. In the transformer-based parameter regression module, we employ both Swin Transformer and Transformer for encoding and decoding, resulting in accurate translation and rotation parameters. In order to solve the problem of insufficient training samples, we introduce random deviations to the extrinsic parameters to augment the training data. Our network can achieve an absolute error of $0.8751\mathrm{cm}$ in translation and $0.0562°$ in rotation, with initial miscalibrations up to $\pm 0.5\mathrm{m}$ in translation and $\pm 5°$ in rotation, outperforming other state-of-the-art methods. Despite the increase in computational cost, the latency remains within a reasonable range.

REFERENCES

[1] X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C.-L. Tai, "Transfusion: Robust lidar-camera fusion for 3d object detection with transformers," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 1090–1099.

[2] Z. Chen, Z. Li, S. Zhang, L. Fang, Q. Jiang, and F. Zhao, "Autoalignv2: Deformable feature aggregation for dynamic multi-modal 3d object detection," *arXiv preprint arXiv:2207.10316*, 2022.

[3] Y. Li, J. Deng, Y. Zhang, J. Ji, H. Li, and Y. Zhang, "Ezfusion: A close look at the integration of lidar, millimeter-wave radar, and camera for accurate 3d object detection and tracking," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 182–11 189, 2022.

[4] T. Shan, B. Englot, C. Ratti, and D. Rus, "Lvi-sam: Tightly-coupled lidar-visual-inertial odometry via smoothing and mapping," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 5692–5698.

[5] J. Lin and F. Zhang, "R^3live: A robust, real-time, rgb-colored, lidar-inertial-visual tightly-coupled state estimation and mapping package," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 10 672–10 678.

[6] A. Geiger, F. Moosmann, Ö. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 3936–3943.

[7] X. Gong, Y. Lin, and J. Liu, "3d lidar-camera extrinsic calibration using an arbitrary trihedron," *Sensors*, vol. 13, no. 2, pp. 1902–1918, 2013.

[8] J. Kümmerle, T. Kühner, and M. Lauer, "Automatic calibration of multiple cameras and depth sensors with a spherical target," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1–8.

[9] G. Pandey, J. McBride, S. Savarese, and R. Eustice, "Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 26, no. 1, 2012, pp. 2053–2059.

[10] Z. Taylor and J. Nieto, "Motion-based calibration of multimodal sensor arrays," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 4843–4850.

[11] J. Levinson and S. Thrun, "Automatic online calibration of cameras and lasers." in *Robotics: science and systems*, vol. 2, no. 7. Citeseer, 2013.

[12] N. Schneider, F. Piewak, C. Stiller, and U. Franke, "Regnet: Multimodal sensor registration using deep neural networks," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1803–1810.

[13] J. Shi, Z. Zhu, J. Zhang, R. Liu, Z. Wang, S. Chen, and H. Liu, "Calibrcnn: Calibrating camera and lidar by recurrent convolutional neural network and geometric constraints," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 197–10 202.

[14] X. Lv, B. Wang, Z. Dou, D. Ye, and S. Wang, "Lccnet: Lidar and camera self-calibration using cost volume network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2894–2901.

[15] X. Zhou, V. Koltun, and P. Krähenbühl, "Tracking objects as points," in *European Conference on Computer Vision*. Springer, 2020, pp. 474–490.

[16] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong *et al.*, "Swin transformer v2: Scaling up capacity and resolution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 12 009–12 019.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[18] Q. Zhang and R. Pless, "Extrinsic calibration of a camera and laser range finder (improves camera calibration)," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3. IEEE, 2004, pp. 2301–2306.

[19] T. Tóth, Z. Pusztai, and L. Hajder, "Automatic lidar-camera calibration of extrinsic parameters using a spherical target," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 8580–8586.

[20] J. Beltrán, C. Guindel, A. de la Escalera, and F. García, "Automatic extrinsic calibration method for lidar and camera sensor setups," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 10, pp. 17 677–17 689, 2022.

[21] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[22] X. Li, Y. Xiao, B. Wang, H. Ren, Y. Zhang, and J. Ji, "Automatic targetless lidar–camera calibration: a survey," *Artificial Intelligence Review*, vol. 56, no. 9, pp. 9949–9987, 2023.

[23] C. Yuan, X. Liu, X. Hong, and F. Zhang, "Pixel-level extrinsic self calibration of high resolution lidar and camera in targetless environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7517–7524, 2021.

[24] G. Iyer, R. K. Ram, J. K. Murthy, and K. M. Krishna, "Calibnet: Geometrically supervised extrinsic calibration using 3d spatial transformer networks," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1110–1117.

[25] Y. Li, A. W. Yu, T. Meng, B. Caine, J. Ngiam, D. Peng, J. Shen, Y. Lu, D. Zhou, Q. V. Le *et al.*, "Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 182–17 191.

[26] R. Wan, S. Xu, W. Wu, X. Zou, and T. Cao, "From one to many: Dynamic cross attention networks for lidar and camera fusion," *arXiv preprint arXiv:2209.12254*, 2022.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[28] F. Yu, D. Wang, E. Shelhamer, and T. Darrell, "Deep layer aggregation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2403–2412.

[29] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable convnets v2: More deformable, better results," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 9308–9316.

[30] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[31] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2017.

[33] H. Shang and B.-J. Hu, "Calnet: Lidar-camera online calibration with channel attention and liquid time-constant network," in *2022 26th International Conference on Pattern Recognition (ICPR)*. IEEE, 2022, pp. 5147–5154.

[34] Y. Wu, M. Zhu, and J. Liang, "Psnet: Lidar and camera registration using parallel subnetworks," *IEEE Access*, vol. 10, pp. 70 553–70 561, 2022.

[35] J. Zhu, J. Xue, and P. Zhang, "Calibdepth: Unifying depth map representation for iterative lidar-camera online calibration," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 726–733.