CalibBD: Extrinsic Calibration of the LiDAR and Camera Using a Bidirectional Neural Network

AN DUY NGUYEN1 AND MYUNGSIK YOO2

1Department of Information Communication Convergence Technology, Soongsil University, Seoul 06978, South Korea
2School of Electronic Engineering, Soongsil University, Seoul 06978, South Korea
Corresponding author: Myungsik Yoo (myoo@ssu.ac.kr)

This work was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology under Grant NRF-2021R1A2B5B01002559.

ABSTRACT With the rapid growth of self-driving vehicles, automobiles demand diverse data from multiple sensors to perceive the surrounding environment. Calibrating preprocessing between multiple sensors is necessary to utilize the data effectively. In particular, the LiDAR-camera pair, a suitable complement with 2D-3D information for each other, has been widely used in autonomous vehicles. Most traditional calibration methods require specific calibration targets set up under complicated environmental conditions, which require expensive human manual work. In this study, we propose a deep neural network that does not require any specific targets and offline setup to find the six degrees of freedom (6 DoF) transformation between LiDAR and the camera. Unlike previous deep learning CNN-based methods, which use raw 3D point clouds and 2D images frame by frame, CalibBD utilizes Bi-LSTM for sequence data to extract temporal features between consecutive frames. It not only predicts the calibration parameters by minimizing both transformation and depth losses but also calibrates the camera parameters by using temporal loss to refine the calibration parameters. The proposed model achieves a steady performance under various deviations of mis-calibration parameters and achieved higher results in terms of accuracy than the state-of-the-art CNN-based method on the KITTI datasets.

INDEX TERMS LiDAR, camera, calibration, deep learning, accuracy.

I. INTRODUCTION

Nowadays autonomous vehicles and robots have replaced humans in challenging tasks that require a significant amount of human effort. Most autonomous systems must operate in a complex environment affected by various types of noise. The perception of an autonomous system needs to collect different information from multiple sensors to ensure operational stability and effectiveness in all scenarios. Recently, light detection and ranging (LiDAR) and cameras, which are used in autonomous vehicles and robotics, have been widely used in the field of computer vision. LiDAR is one of the most accurate 3D sensors that provides spatial and depth information. However, the point cloud provided by LiDAR lacks color and texture information, making it difficult to distinguish or identify the surrounding object. Over the past few decades, cameras have made great progress by providing rich color and texture information with very high resolution. However, cameras are sensitive to the intense illumination of lights, and it is challenging to obtain depth information from a 2D image. Therefore, the advantages of LiDAR and cameras compensate for each other’s disadvantages. Hence, calibration between LiDAR and cameras is vital for fusing 2D-3D data and guaranteeing the accuracy and precision of the system performance.

Prior research on LiDAR-camera calibration has focused on the target-based method, which requires a specific target with defined shapes, such as a rectangular checkerboard [1], [2], [3], [5], an orthogonal trihedron, or a spherical target [30], [31], [32]. The extrinsic parameter can be calculated by determining the corresponding matching features in the 2D image and 3D point clouds. However, it requires considerable
manual effort. It is suitable for application in a laboratory environment or a static scenario where LiDAR and cameras have an apparent field of view, and the data collection is affected by natural noises as low as possible. Therefore, the target-based calibration method is not suitable for real-world applications when the LiDAR and camera are mounted in a dynamic autonomous vehicle or a robot working in an environment with many different types of natural noise.

In contrast, we propose a deep learning method to solve the LiDAR-camera online calibration problem, which does not require any specific target without particular environmental settings, saving human effort and resources. Moreover, it can be applied in real-world applications when the results of extrinsic parameters are predicted by training using the KITTI dataset, which is a collection of images and point clouds provided by the LiDAR-camera system in actual urban areas. However, a problem occurs when the system experiences vibrations due to intensive changes in the autonomous vehicle speed or is exposed to intense illumination. This causes errors in various factors between consecutive frames that require the LiDAR-camera system to be recalibrated.

We combined the Bi-LSTM and CNN modules in our model, which leverages the potency of the Bi-LSTM working on the sequence data to recover the aforementioned problem. By extracting the temporal features, the training network obtains information on the related poses and extrinsic and intrinsic parameters of the sensors in continuous frames. Therefore, the model can detect this problem error whenever it occurs, and recalibrates the sensors by compensating for the changed factor caused by noise as shown in Fig. 1. Our contributions can be summarized as follows:

- CalibBD is the end-to-end learning network merging Bi-LSTM and CNN modules for LiDAR-camera calibration problems. The network is trained on the sequence data with temporal loss to quantify the geometric error.
- Further improving the calibration performance in terms of the accuracy compared to state-of-the-art CNN based methods by optimizing the pose transformation of sensors between consecutive frames.
- The model uses the KITTI dataset [21] to evaluate the calibration accuracy. The model performed well under many different scenarios with large mis-calibration ranges.

II. RELATED WORKS

In the last decade, many different methods have been proposed to solve the calibration problem of multimodal sensor systems, especially LiDAR-camera systems. In summary, calibration techniques can be divided into two main approaches: target-based and target-less, as shown in Table 1.

A. TARGET-BASED APPROACH

Target-based approaches require researchers to prepare calibration targets that are usually planar and have a specific shape, such as rectangular checkerboards. The corresponding features, such as corner points, vertices, and boundary lines, are generally extracted from the 2D image and 3D point cloud to obtain the extrinsic parameters between sensors. Zhang and Pless [1] were the first to provide a solution to calibrate a camera and 2D laser finder using a checkerboard that provides a baseline for further studies on 3D LiDAR and cameras.

In contrast to Zhang and Pless et al. [1], by using five different poses on the checkerboard for calibration, Verma et al. [2] required only three different poses of LiDAR and camera for calibration using a genetic algorithm. Furthermore, Zhou et al. [3] proposed an approach for extracting the boundary lines and plane correspondence using the RANSAC algorithm [24]. They proved that parallel boundaries share the same constraints that obtain calibration parameters using a single pose of the checkerboard. In addition, the fundamental operation of LiDAR is time-of-flight (ToF), which is affected by the reflection bias of the scanning lasers. Kim, et al. [33] has proposed an SVD and “point to plane” ICP method to regress a rigid transformation of LiDAR and cameras by extracting and matching 3D-3D corresponding planes.

In general, this often occurs when LiDAR scans narrow areas of the checkerboard, such as boundary lines. Therefore, edge points and vertices are not always available and accurate measurements are not always provided within the point cloud. An et al. [5] designed a white auxiliary board behind the checkerboard to ensure the accuracy of edge-point measurements. Combining 3D-2D and 3D-3D correspondences [5]
TABLE 1. Related works on the LiDAR - camera calibration.

| Category         | Method     | Dataset | Description               | Deviation | Rotation error (degree) | Translation error (cm) |
|------------------|------------|---------|---------------------------|-----------|-------------------------|------------------------|
| Target-based     | Zhang and Pless et al. [1] | A checkerboard | PnP algorithm             | -         | 1.95                    | 2.37                   |
|                  | L. Zhou et al. [3]       | A checkerboard | RANSAC + SVD algorithm    | -         | 1.50                    | 2.04                   |
|                  | Kim, E.S et al. [33]     | A checkerboard | 3D-3D corresponding planes + ICP | -         | 0.87                    | 1.42                   |
|                  | Huang J.K et al. [6]     | A checkerboard | Using reference LiDAR frame + IoU | -         | 0.74                    | 1.02                   |
| Target-less      | RegNet [10]             | KITTI    | CNN + NiN                 | 0.28      |                         |                         |
|                  | CalibDNN [28]            | KITTI    | DNN                       | 0.42      |                         | 5.3                    |
|                  | CalibNet [12]            | KITTI    | CNN + 3D spatial layer    | 0.41      |                         | 4.34                   |
|                  | CFNet [29]              | KITTI    | CNN+PnP                   | 0.087     |                         | 0.995                  |

provided more constraints to minimize the reprojection errors and optimize the final calibration parameters.

Huang et al. [6] has improved on Zhou’s work on feature extraction to obtain the vertices of the checkerboard without using the RANSAC algorithm. By assembling a reference for the LiDAR frame, they minimized the cost function, which shows the relationship between the actual point cloud frame and the reference LiDAR frame to achieve the “best” transformation for estimating the vertices. However, target-based methods have the disadvantage that they do not perform well under adversary external conditions and therefore cannot be used in a highly dynamic environment or in real-time applications.

B. TARGET-LESS APPROACH

Unlike the target-based approach, the target-less approach can estimate the 6 DoF rigid transformation between the LiDAR camera without using any calibration targets. Scaramuzza et al. [7] were one of the first target-less approaches that manually established correspondence in the overlapping area between the field of view of the 3D laser range finder (LRF) and camera, and the extrinsic calibration parameters were obtained using the PnP algorithm. Pandey et al. [8] presented an algorithm based on mutual information (MI) that shows the correlation between point cloud reflection and image intensities. The calibration parameters were obtained by maximizing the MI with the steepest gradient descent. Furthermore, Ishikawa et al. [9] introduced a motion-based method for target-less calibration problems using a hand-eye robot. By providing the initial parameters of LiDAR and camera motions, extrinsic parameters were acquired by iterating the motion scale value between consecutive frames until convergence.

With the rapid development of deep learning in computer vision, its applications have brought innovation to segmentation, classification, object detection, and object tracking. It has shown outstanding performance in extrinsic calibration compared with other algorithms. Schneide et al. [10] were among the first researchers to work on a deep learning method for calibration by presenting RegNet. They leveraged the work of Network in Network [11] for extracting and matching the semantic features in LiDAR and cameras. These features are then fed forward to the fully connected layer (FC) to predict the 6 DoF transformation. Furthermore, Iyer et al. [12] proposed CalibNet, which utilizes a 3D spatial transformer by maximizing the photometric and geometric losses between the point cloud and image input, and has a strategy similar to that of RegNet. Zhu et al. [15] introduced a method using the semantic information obtained from SOIC [14], which uses the semantic centroid of the image and point cloud to transform the problem into a PnP problem. Yuan et al. [13] showed RGNNet, which considers the tolerance loss function, to achieve calibrated parameters by leveraging Riemannian geometry. Zhao et al. [28] proposed a DNN architecture that is lightweight with a single iteration to obtain a calibration transformation by maximizing the consistency of multimodal data. Its performance can be refined using a training model with multiple iterations of different miscalibration ranges. Recently, CFNet [29] was proposed by Xudong to define the calibration flow, which is the deviation in pixel positions between the initial reprojection and ground truth.

We observed that none of the above deep learning methods consider the pose transformation of the temporal information between continuous frames. The problem occurs when the calibration parameters are affected by various external factors, such as system vibrations. Because a single frame is affected by this error, the feeding the input frame by frame input affects the training process results. Inspired by the traditional method, which uses the relative pose transformation of LiDAR and camera motions in consecutive frames to obtain extrinsic parameters. In this study, we adopted the Bi-LSTM module to extract temporal information, which includes relative pose transformation and enhances the calibration accuracy by optimizing the calibrated constraint between successive frames.

III. METHODOLOGY

Our work aims to design an end-to-end deep learning model using sequences of data from LiDAR and a camera. First, the point cloud from LiDAR is projected onto a camera frame
by an initial transformation $T_{\text{init}}$ which is miscalibrated to obtain a LiDAR depth map. Using the sequence data of depth maps from LiDAR and images from the camera as input pairs, we can obtain the calibration parameters, including the 6-DoF extrinsic parameter with rotation and translation between LiDAR and camera coordinates. In this section, we describe the details of our proposed calibration method, including the data processing, network architecture, and loss function. The workflow of the proposed architecture is shown in Figure 2.

### A. DATA PROCESSING

The inputs were the sequence data of the pair between the depth maps and RGB images, as discussed above. We need to transform the 3D point cloud $P_i = [X_i, Y_i, Z_i]$ into a depth map with 2D pixel coordinates $p_i = [u_i, v_i]$ by providing an initial transformation $T_{\text{init}}$ and intrinsic camera parameters $K$. The projection process can be defined by the following formula:

$$z_i = K \cdot T_{\text{init}} \cdot \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix},$$

where $z_i$ is the depth value of the projected point at each pixel in the camera coordinates, which expresses the distance information from the camera to the corresponding points in the LiDAR point cloud. The pixel depth value is zero if there is no point from the LiDAR point cloud projected onto this pixel.

The $R_{\text{init}}$ and $T_{\text{init}}$ are the initial rotation matrix and the initial translation vector of the initial transformation $T_{\text{init}}$ between LiDAR and the camera, respectively. We use $\Delta T$ as a random deviation parameter from the ground truth to generate the initial transformation $T_{\text{init}} = T_{\text{gt}} \cdot \Delta T$ to provide different training data. It consists of three adjacent pairs of images and depth maps that use the same initial transformation $T_{\text{init}}$ as inputs.

### B. NETWORK ARCHITECTURE

The proposed architecture is illustrated in Fig. 2, which contains two main branches of 2D images and depth
maps between continuous frames as inputs. The model obtains 6 DoF extrinsic parameters based on the encoder-decoder architecture combined with the Bi-LSTM module. Fig. 4 shows the detail of the encoder-decoder architecture. First, the input of the RGB image and depth map is fed to the encoder network to extract multiscale features with semantic information regarding the surrounding environment. Then, a decoder with multiple scaling layers is used for upsampling the feature map between the LiDAR and camera to find the correspondence between the pixel and depth value. Next, the Encoder-Decoder network outputs images with pairs of intensity and depth in pixels, which are then fed to the SIFT module to extract the temporal features at the current frames. The SIFT output will be input to the Bi-LSTM module, as shown in Fig. 5, with forward and backward LSTM to estimate the camera’s pose during successive frames for the self-calibration. Finally, optimizing the LiDAR-camera transformation in consecutive frames is beneficial and enhances the calibration performance in terms of accuracy.

1) ENCODER NETWORK
Owing to the success of ResNet [16], we applied the ResNet-18 architecture to both RGB images and depth maps to extract multiple scale features. However, the weights of ResNet in the two branches are not shared because the number of RGB images and LiDAR depth map channels is different, whereas the LiDAR depth map has only one channel instead of three channels as RGB images. Leaky ReLU is used as the activation layer in the encoder network to ignore the “dying” ReLU problem with a slightly negative slope. Finally, feature maps were obtained by fusing the information from both the branches.

2) DECODER NETWORK
The feature maps of the encoder network are fed forward to the decoder network as the input. Owing to the five-level downsampling in the encoder network, the decoder network consists of five different levels to upsample the feature maps back to the original size. The structure of each level is inspired by PWCNet [17], which includes a warping layer, cost volume layer, and up-sampling layer. The warping layer has the function of scaling the image pixels to the correct size owing to camera distortions. The cost-volume layer is assumed to compute the matching cost between the depth map feature and the RGB features based on CMRNet [18]. The matching cost of the cost-volume layer is defined as follows:

\[ cv(p_{rgb}, p_{lidar}) = \frac{1}{N}c(x_{rgb}(p_{rgb}))^T c(x_{lidar}(p_{lidar})), \]

where \( p_{rgb} \) and \( p_{lidar} \) are the corresponding pixels in the RGB images and depth maps, respectively, and \( c(x) \) is a feature map vector. \( N \) is the total number of pixels that consider the vector \( c(x) \) column length, and \( T \) is the transpose operator applied to the RGB vector. The dimensions of the cost volume are different because of the different levels of the decoder, which is defined as \( d^2 \times H \times W \), where \( H \) and \( W \) are the height and width of the feature map \( x_{rgb}^D \) and \( x_{lidar}^D \), respectively, at each level of the decoder.

The upsampling layer is constructed using multiple CNN layers, with the output of the cost volume layer as inputs. The feature numbers of each convolution in the upsampling layer were 128, 128, 96, 64, and 32, respectively. At each decoder layer, the upsampling layers do not share weights and own their individual parameters. The output of Decoder 1 is fed toward Bi-LSTM as the input.

3) BI-LSTM NETWORK
Bi-directional long short-term memory (Bi-LSTM) combines two independent LSTM, one in the forward time series and the other in the backward time series. It can utilize temporal information from both sides of the sequence data. Due to the sequence data, the camera is susceptible to focusing loss when external noises are present such as excessive vibrations during aggressive driving. When the camera loses focus, the information of the principal points and focal length is no longer constant. This changes the camera’s intrinsic and affects the quality of the RGB images. Evidently, the focus loss causes the change in focal length in world units and creates a variant of the focal length in pixels. Consequently, some factors of the camera parameters are changed [33], such as the focal length, optical point, and camera poses that affect the intensity and precision of the captured image in pixel coordinates. It highly reduces the accuracy of the camera calibration and precision of the transformation between LiDAR and camera calibration. Therefore, the camera must be recalibrated over consecutive frames.

In this study, the proposed architecture adopts the CNN features as inputs for the BiLSTM module. The temporal features are extracted using the scale-invariant feature transform (SIFT) to determine the corresponding features between successive frames. The SIFT creates a “scale-space” to represent the original images that are scale-independent. After using SIFT, we obtain temporal features that contain position and intensity information. The main advantage of the SIFT feature extractor is that it is unaffected by the size and orientation of the images, which is suitable for camera self-calibration in continuous frames. Inspired by the sequential learning [34] used for visual odometry (VO), the Bi-LSTM was employed with forward and backward pipelines to extract more orientation features based on the temporal features. These features were then used to estimate the camera’s pose. By maintaining the memory in the hidden states, the Bi-LSTM aids the calibration of the camera transformation in consecutive frames. The number of hidden states was set to 256 for both forward and backward states. The updates of the forward \( H^f \) and backward \( H^b \) hidden states are as follows:

\[ H^f_t = \sigma(X_t W^f_{sh} + H_{t-1} W^f_{bh} + b^f_h), \]
\[ H^b_t = \sigma(X_t W^b_{sh} + H_{t+1} W^b_{bh} + b^b_h), \]

where \( \sigma \) denotes a sigmoid activation function. \( X_t \) is the input at time \( t \) with \( X_t \subset \mathbb{R}^{n \times d} \), where \( n \) is the number of data points.
and $d$ is the number of inputs ($d = 3$ in this study). Weight $W_{sh}$ is the weight from input $X$ to the hidden state $H$ of the LSTM, and weight $W_{hh}$ is the weight of the hidden state $H$ forward to the hidden state $H$ in the next LSTM module, where $W_{sh} \subset \mathbb{R}^{h \times d}$, $W_{hh} \subset \mathbb{R}^{h \times h}$, where $h$ is the number of hidden states, and $b_h \subset \mathbb{R}^{1 \times h}$ is the bias of the LSTM. Finally, the hidden state of Bi-LSTM, which feed-forward the output, is the sum of the forward and backward hidden states:

$$H_t = H_t^f \oplus H_t^b. \quad (6)$$

### IV. LOSS FUNCTION

In this section, we utilize three different types of loss functions to obtain the total loss: transformation loss, depth map loss, and temporal loss. The final loss applied during the training process is defined as

$$L_{total} = \lambda_{tr} L_{tr} + \lambda_d L_d + \lambda_t L_t. \quad (7)$$

where $\lambda_{tr}$, $\lambda_d$, and $\lambda_t$ denote the weight of each loss with a different critical effect to the performance of the model in terms of calibration accuracy. The weights are set to 1, 0.01, and 0.01, respectively, and were determined experimentally.

4) TRANSFORMATION LOSS

The goal is to estimate the 6 DoF rigid transformation between LiDAR and the camera, which is closest to the ground truth transformation. It includes the rotation and translation losses based on L1 loss, as follows:

$$L_{tr} = \lambda_{rot} L_{rot} + \lambda_{trans} L_{trans}, \quad (8)$$

where $\lambda_{rot}$ and $\lambda_{trans}$ represent the weights of rotation and translation losses, respectively. Owing to the rotation loss, rotation is present in quaternions, providing a more accurate comparison than using the Euclidean distance. The rotation loss of the prediction and ground truth can be expressed as follows:

$$L_{rot} = D_d(q_{pred}, q_{gt}), \quad (9)$$
where $q_{pred}$ and $q_{gt}$ are the prediction and ground truth of the rotation in the quaternions, respectively. $D_{q}$ denotes the angular distance between two quaternions based on [19].

The L1-loss for the translation vector was calculated as follows:

$$L_{trans} = |t_{gt} - t_{pred}|.$$

(10)

5) DEPTH MAP LOSS

We can obtain the predicted depth map by projecting the 3D LiDAR point cloud onto a 2D image with the prediction of the 6 DoF rigid transformation. The depth map was generated based on Eq. 11:

$$z_{pi} \cdot \begin{bmatrix} u_{pi} \\ v_{pi} \\ 1 \end{bmatrix} = K \cdot T_{pred} \cdot \begin{bmatrix} X_I \\ Y_I \\ Z_I \\ 1 \end{bmatrix}.$$

The depth map loss is the difference between each projected point $P_{pred} = (u_{pi}, v_{pi}, z_{pi})$ in the predicted depth map $D_{pred}$ and the same pixel position $P_{gt} = (u_{gt}, v_{gt}, z_{gt})$ in the ground truth depth map $D_{gt}$. The depth map loss can then be expressed as follows:

$$L_{d} = \frac{\sum_{p \in D_{gt}} ||P_{gt} - P_{pred}||^2}{N},$$

(12)

where N is the number of projected points on the ground-truth depth map.

6) TEMPORAL LOSS

The Bi-LSTM module extracts temporal features to recover the camera’s issues miscalibrated in different poses. It consists of the intrinsic parameter K and the related camera pose information between successive frames. Temporal loss can be formulated as follows:

$$L_{t} = q^I_{t} \cdot K^{-T} \cdot E \cdot K^{-1} \cdot q_{t}^{I+1},$$

(13)

where $q^I_{t}$ and $q_{t}^{I+1}$ are the matching pairs of SIFT feature points between two continuous frames. K is the intrinsic parameter of the camera and E denotes the essential matrix of the relative poses between two frames. In [27], a method was proposed to optimize camera transformation by providing two camera poses. During the training process, the updated weights in Bi-LSTM attempt to optimize the camera transformation $T_{cam}$ between sequence frames with the benefit of refining the parameters of the LiDAR-camera transformation according to Eq. 2.

V. EXPERIMENTS AND RESULTS

In this section, we provide a detailed description of the data preparation, training details, and evaluation metrics, and analyze the model performance in terms of the results and evaluations.

A. DATA PREPARATION

The proposed model was evaluated using the KITTI dataset [21], consisting of RGB images and LiDAR point clouds recorded from various scenes, including vehicles, roads, buildings, and residential areas during real-time driving. The RGB image and LiDAR point cloud were paired in order in each sequence based on synchronized timestamps between the camera and LiDAR.

First, we generate depth maps by projecting the 3D LiDAR point cloud into a 2D image coordinate with an initial transformation and the intrinsic parameters of the camera. We obtained an initial miscalibrated transformation to effectively generate a wide variety of datasets for training, where the initial transformation $T_{init}$ was calculated by $T_{init} = T_{gt} \cdot \Delta T$. The range of the deviation $\Delta T$ is set to a wide range ($\pm 1.5m$, $\pm 20^\circ$) of translation and rotation, respectively. We took pairs of RGB images and depth maps in three successive frames as input samples. Regarding the different dimensions of RGB images in the KITTI dataset (changing from $1224 \times 370$ to $1242 \times 376$), we resize them by adding all images to a fixed size of $1280 \times 384$, which satisfies the CNN architecture condition with the width and height of the input dimension multiplied by 32.

Sequences from 06 to 21 were used to generate 29,416 samples for the training dataset, and sequences 00 to 05 were used for testing with 4,854 samples. None of the testing data appeared during the training phase.

B. TRAINING DETAILS

According to the training process, the network used PyTorch (1.10.1) and Adam Optimizer [22] with default parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, and an initial learning rate of $1e^{-3}$. The network was trained on two RTX 360 GPUs with a batch size of 24 and 80 epochs. Two LSTM layers stack the Bi-LSTM module forward and backward, with 256 hidden states. To prevent the overfitting problem, we apply dropout [23] to the fully connected layer with a factor of 0.5.

C. EVALUATION METRICS

To provide a fair comparison to CFNet [29], which is a state-of-the-art method, we evaluate the network’s performance using Euclidian distance loss for the translation and Euler angle loss for the rotation. The absolute value of the translation error can be expressed as

$$e_t = |t_{pred} - t_{gt}|.$$

(14)

where $t_{pred}$ is the predicted translation vector and $t_{gt}$ is the ground-truth translation. The metric is applied to each direction of the translation vector in X, Y, and Z. Then, the mean value of translation errors can be computed as

$$\bar{t} = (e_X + e_Y + e_Z)/3.$$
FIGURE 6. Examples of the calibration results on different scenes of the KITTI raw dataset (a) The mis-calibrated 3D-2D image by projecting LiDAR point clouds into corresponding 2D RGB images with initial extrinsic parameters which have different ranges of the deviation, (b) the results of the 3D-2D projected image with model prediction which re-calibrate the initial transformation (c) the corresponding ground truth.

TABLE 2. Comparison of the rotation and translation error with initial off-range ($\pm 1.5m, \pm 20^\circ$).

| Method  | Deviation Range | Rotation error (degree) | Translation error (cm) |
|---------|-----------------|-------------------------|------------------------|
|         |                 | $e_{\text{roll}}$ | $e_{\text{pitch}}$ | $e_{\text{yaw}}$ | $\hat{r}$ | $e_X$ | $e_Y$ | $e_Z$ | $\hat{t}$ |
| RegNet  | ($\pm 1.5m, \pm 20^\circ$) | 0.24 | 0.25 | 0.36 | 0.28 | 7 | 7 | 4 | 6 |
| CFNet   | ($\pm 1.5m, \pm 20^\circ$) | 0.059 | 0.110 | 0.092 | 0.087 | 1.025 | 0.919 | 1.042 | 0.995 |
| CalibBD | ($\pm 1.5m, \pm 20^\circ$) | 0.054 | 0.116 | 0.075 | 0.081 | 0.963 | 0.846 | 0.828 | 0.889 |

TABLE 3. Comparison of the rotation and translation error with initial off-range ($\pm 0.2m, \pm 20^\circ$).

| Method  | Deviation Range | Rotation error (degree) | Translation error (cm) |
|---------|-----------------|-------------------------|------------------------|
|         |                 | $e_{\text{roll}}$ | $e_{\text{pitch}}$ | $e_{\text{yaw}}$ | $\hat{r}$ | $e_X$ | $e_Y$ | $e_Z$ | $\hat{t}$ |
| CalibDNN | ($\pm 0.2m, \pm 20^\circ$) | 0.11 | 0.35 | 0.18 | 0.42 | 6.2 | 4.3 | 5.4 | 5.3 |
| CalibNet | ($\pm 0.2m, \pm 20^\circ$) | 0.15 | 0.9 | 0.18 | 0.41 | 4.2 | 1.6 | 7.22 | 4.34 |
| CFNet   | ($\pm 0.2m, \pm 20^\circ$) | 0.028 | 0.125 | 0.105 | 0.086 | 0.463 | 1.230 | 0.802 | 0.831 |
| CalibBD | ($\pm 0.2m, \pm 20^\circ$) | 0.021 | 0.136 | 0.084 | 0.08 | 0.325 | 1.066 | 0.878 | 0.756 |

The rotation error is calculated as $\hat{r} = (e_{\text{roll}} + e_{\text{pitch}} + e_{\text{yaw}})/3$. The rotation error can also be represented in the quaternion as

$$e_r = D(q_{gt} \ast \text{inv}(q_{\text{pred}})).$$

D. RESULTS AND EVALUATIONS

The calibration results of the model were evaluated using the KITTI raw dataset, as shown in Tables 2 and 3. It shows a comparison of the performance in rotation and translation errors compared to RegNet [10], CalibNet [12], CalibDNN [28], and CFNet [29], where CFNet [29] denotes the ’state-of-the-art’ deep learning method on the target-less LiDAR-camera calibration problem. To estimate extrinsic
TABLE 4. Results of the rotation and translation error on various sequences of the KITTI odometry dataset with initial mis-calibrated range (±1.5m, ±20°).

| Sequence | Rotation (degree) | Translation (cm) |
|----------|------------------|------------------|
|          | $\varepsilon_{\text{roll}}$ | $\varepsilon_{\text{pitch}}$ | $\varepsilon_{\text{yaw}}$ | $\varepsilon_{r}$ | $\varepsilon_{x}$ | $\varepsilon_{y}$ | $\varepsilon_{z}$ | $\varepsilon_{t}$ |
| Mean     | 0.032            | 0.055            | 0.038            | 0.092            | 0.978           | 1.126           | 1.473           | 1.724           |
| Median   | 0.029            | 0.047            | 0.031            | 0.087            | 0.952           | 1.056           | 1.542           | 1.653           |
| Std      | 0.022            | 0.05             | 0.029            | 0.065            | 0.437           | 0.476           | 0.802           | 1.062           |
| Mean     | 0.044            | 0.072            | 0.043            | 0.122            | 0.983           | 1.242           | 1.478           | 1.926           |
| Median   | 0.039            | 0.067            | 0.032            | 0.118            | 1.041           | 1.062           | 1.589           | 1.792           |
| Std      | 0.024            | 0.078            | 0.025            | 0.062            | 0.432           | 0.457           | 0.843           | 1.021           |
| Mean     | 0.069            | 0.088            | 0.048            | 0.089            | 0.128           | 0.137           | 0.357           | 1.767           |
| Median   | 0.089            | 0.082            | 0.064            | 0.082            | 0.108           | 1.556           | 0.363           | 1.568           |
| Std      | 0.041            | 0.072            | 0.031            | 0.048            | 0.063           | 0.418           | 0.149           | 1.045           |
| Mean     | 0.078            | 0.121            | 0.072            | 0.112            | 0.522           | 0.952           | 0.769           | 1.929           |
| Median   | 0.078            | 0.104            | 0.072            | 0.102            | 0.508           | 0.827           | 0.694           | 1.802           |
| Std      | 0.056            | 0.057            | 0.042            | 0.054            | 0.490           | 0.214           | 0.179           | 1.408           |
| Mean     | 0.027            | 0.032            | 0.014            | 0.072            | 0.921           | 0.668           | 0.548           | 1.732           |
| Median   | 0.025            | 0.026            | 0.013            | 0.069            | 1.122           | 0.490           | 0.378           | 1.582           |
| Std      | 0.014            | 0.020            | 0.010            | 0.027            | 0.774           | 0.544           | 0.458           | 0.952           |

FIGURE 7. Examples of the calibration results with different initial transformation on various scences of the KITTI odometry dataset. For better visualization, the images of the prediction and ground truth are cropped, and the corresponding targets are displayed in the red rectangles (a) the mis-calibration case (b) the cropped result of the prediction (c) the ground truth.

calibration parameters, these methods utilize different networks that are trained with varying deviations of misalignment. Table 2 presents the results resulting from the same experimental settings for RegNet and CFNet with their range of deviations as a comparison. (±1.5m, ±20°) on the KITTI raw dataset of the sequence “2009 – 09 – 30”. 
We achieved mean rotation errors of 0.081° (roll: 0.054°, pitch: 0.116°, yaw: 0.075°) and a mean translation error value of 0.889 cm (X = 0.963 cm, Y = 0.846 cm, Z = 0.828 cm). Table 3 presents the results compared to the studies CalibNet, CalibDNN, and CFNet for the experimental setting with the initial off-range set as (±0.2 m, ±20°).

In this experiment, the mean angle error of the rotation predicted by our network was 0.08° (roll: 0.021°, pitch: 0.136°, yaw: 0.084°) and the mean error of the translation vector was 0.756 cm (X = 0.325 cm, Y = 1.066 cm, Z = 0.878 cm). Thus, the proposed network achieved outperformance compared to all the methods evaluated in different settings.

Fig. 6 displays the model prediction results in 3D-2D projected images on the different scenes. This shows that the projected images obtained by the predicted transformation and the ground truth are very similar under the varying ranges of miscalibration.

In Table 4, we evaluated the proposed model on the KITTI odometry dataset, which consists of different extrinsic parameters in various scenarios. We used sequences from 00 to 04 to consider the mean, median, and standard deviation of the rotation matrix and translation vector. It shows good performance for sequences 00, 02, 03, and 04, which were collected from the urban area, including many objects such as vehicles, pedestrians, and buildings. In addition, Table 4 provides a detailed calibration of sequence 01, which consists of highway scenarios, which were not used during the training period. The mean angles of rotation error \( e_r < 0.122° \) and absolute mean of the translation error \( e_t < 2\text{cm} \) for sequence 01.

Fig. 7 shows the visualization results of 3D-2D projected images in various scenarios on the KITTI odometry dataset. This shows that the network can recalibrate large objects such as cars and planes on the roadsides in the first and second rows of Fig. 7. The third and fourth rows of Fig. 7 also show good matching to re-calibrate the small targets like a human or a traffic signal sign which are very similar compared to the ground truth. Regarding the major or minor deviations of the initial transformation, the model still predicts the best transformation for matching all corresponding RGB pixels and LiDAR points.

E. CONCLUSION

In this study, we proposed a supervised learning network called CalibBD to solve the miscalibration problem by estimating the 6 DoF transformation of the LiDAR and cameras. By merging the CNN and Bi-LSTM networks, the proposed network improves calibration performance in terms of accuracy and enables its application in online real-time calibration. To prevent outlier errors affecting the calibration performance, such as weather conditions or system vibration, we used temporal features consisting of the pose transformation information of sensors between successive frames. The proposed method shows good performance with large to minor deviations of the initial transformation, as demonstrated by the experiment and results. It achieves a mean rotation error of 0.08° and a mean absolute translation error of 0.756 cm with miscalibrated ranges of (±0.2 m, ±20°), which shows an outperformance compared to state-of-the-art CNN-based methods.

REFERENCES

[1] Q. Zhang and R. Pless, “Extrinsic calibration of a camera and laser range finder (improves camera calibration),” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Sendai, Japan, Sep. 2004, pp. 2301–2306.
[2] S. Verma, J. S. Berrio, S. Worrall, and E. Nebot, “Automatic extrinsic calibration between a camera and a 3D Lidar using 3D point and plane correspondences,” 2019, arXiv:1904.12433.
[3] L. Zhou, Z. Li, and M. Kaess, “Automatic extrinsic calibration of a camera and a 3D LiDAR using line and plane correspondences,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Madrid, Spain, Oct. 2018, pp. 5562–5569.
[4] E. S. Kim and Y. B. Park, “Extrinsic calibration between camera and LiDAR sensors by matching multiple 3D planes,” Sensors, vol. 20, no. 1, p. 52, 2019.
[5] P. An, T. Ma, K. Yu, B. Fang, J. Zhang, W. Fu, and J. Ma, “Geometric calibration for LiDAR-camera system fusing 3D-2D and 3D-3D point correspondence,” Opt. Exp., vol. 28, no. 2, pp. 2122–2141, 2020.
[6] J.-K. Huang and J. W. Grizzle, “Improvements to target-based 3D LiDAR for camera calibration,” IEEE Access, vol. 8, pp. 134101–134110, 2020, doi: 10.1109/ACCESS.2020.3010734.
[7] D. Scaramuzza, A. Harati, and R. Siegwart, “Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., San Diego, CA, USA, Oct. 2007, pp. 4164–4167.
[8] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, “Automatic targetless extrinsic calibration of a 3D Lidar and camera by maximizing mutual information,” in Proc. 26th AAAI Conf. Artif. Intell., 2012, pp. 2053–2059.
[9] R. Ishikawa, T. Oishi, and K. Ikeuchi, “LiDAR and camera calibration using motions estimated by sensor fusion odometry,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Oct. 2018, pp. 7342–7349.
[10] N. Schneider, F. Piewak, C. Stiller, and U. Franke, “RegNet: Multimodal sensor registration using deep neural networks,” in Proc. IEEE Intell. Vehicles Symp. (IV), Jun. 2017, pp. 1803–1810.
[11] M. Lin, Q. Chen, and S. Yan, “Network in network,” 2013, arXiv:1312.4400.
[12] G. Iyer, R. K. Ram, J. K. Murthy, and K. M. Krishna, “CalibNet: Geometrically supervised extrinsic calibration using 3D spatial transformer networks,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Oct. 2018, pp. 1110–1117.
[13] K. Yuan, Z. Guo, and Z. J. Wang, “RGGNet: Tolerance aware LiDAR-camera online calibration with geometric deep learning and generative model,” IEEE Robot. Automat. Lett., vol. 5, no. 4, pp. 6956–6963, Oct. 2020.
[14] W. Wang, S. Nobuhara, R. Nakamura, and K. Sakurada, “SOIC: Semantic online initialization and calibration for LiDAR and camera,” 2020, arXiv:2003.04260.
[15] Y. Zhu, C. Li, and Y. Zhang, “Online camera-LiDAR calibration with sensor semantic information,” in Proc. IEEE Int. Conf. Robot. Automation (ICRA), Paris, France, Aug. 2020, pp. 4970–4976.
[16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2016, pp. 770–778.
[17] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, “PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume,” in Proc. IEEE Conf. Comput. Vis. Pattern Recog., Jun. 2018, pp. 8934–8943.
[18] D. Cattaneo, M. Vaghi, A. L. Ballardini, S. Fontana, D. G. Sorrenti, and W. Burgard, “CMRNet: Camera to LiDAR-map registration,” in Proc. IEEE Intell. Transp. Syst. Conf. (ITSC), Oct. 2019, pp. 1283–1289.
[19] A. Kendall, M. Grimes, and R. Cipolla, “PoseNet: Convolutional network for real-time 6-DOF camera relocalization,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Dec. 2015, pp. 2938–2946.
[20] A. Geiger, F. Moosmann, O. Car, and B. Schuster, “Automatic camera and range sensor calibration using a single shot,” in Proc. IEEE Int. Conf. Robot. Autom., May 2012, pp. 3936–3943.
A. D. Nguyen, M. Yoo: CalibBD: Extrinsic Calibration of the LiDAR and Camera Using a Bidirectional Neural Network

[21] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jun. 2012, pp. 3354–3361.

[22] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980.

[23] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” J. Mach. Learn. Res., vol. 15, pp. 1929–1958, Sep. 2014.

[24] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” Commun. ACM, vol. 24, no. 6, pp. 726–740, 1987.

[25] J. Castorena, G. V. Puskorius, and G. Pandey, “Motion guided LiDAR-camera self-calibration and accelerated depth upsampling for autonomous vehicles,” J. Intell. Robotic Syst., vol. 100, nos. 3–4, pp. 1129–1138, Dec. 2020.

[26] Z. Taylor and J. Nieto, “Motion-based calibration of multimodal sensor arrays,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2015, pp. 4843–4850.

[27] T. Shen, Z. Luo, L. Zhou, H. Deng, R. Zhang, T. Fang, and L. Quan, “Beyond photometric loss for self-supervised ego-motion estimation,” in Proc. IEEE Int. Conf. Robot. Autom., May 2019, pp. 6359–6365.

[28] G. Zhao, J. Hu, S. You, and C.-C. J. Kuo, “CalibDNN: Multimodal sensor calibration for perception using deep neural networks,” Proc. SPIE, vol. 11756, Apr. 2021, Art. no. 117561D.

[29] X. Lv, S. Wang, and D. Ye, “CFNet: LiDAR-camera registration using calibration flow network,” Sensors, vol. 21, no. 23, p. 8112, Dec. 2021.

[30] S. Debattisti, L. Mazzei, and M. Pancirolli, “Automated extrinsic laser and camera inter-calibration using triangular targets,” in Proc. IEEE Intel!. Vehicles Symp. (IV), Jun. 2013, pp. 696–701.

[31] R. Gomez-Ojeda, J. Briales, E. Fernandez-Moral, and J. Gonzalez-Jimenez, “Extrinsic calibration of 2D laser-rangefinder and camera based on scene corners,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2015, pp. 3611–3616.

[32] Z. Bu, C. Sun, P. Wang, and H. Dong, “Calibration of the camera and flash LiDAR system with a triangular pyramid target,” Appl. Sci., vol. 11, no. 2, p. 582, Jan. 2021.

[33] Z. Lu and L. Cai, “Camera calibration method with focus-related intrinsic parameters based on the thin-lens model,” Opt. Exp., vol. 28, no. 14, pp. 20858–20878, 2020.

[34] S. Wang, R. Clark, H. Wen, and N. Trigoni, “DeepVO: Towards end-to-end visual odometry with deep recurrent convolutional neural networks,” in Proc. IEEE Int. Conf. Robot. Automat. (ICRA), May 2017, pp. 2043–2050.

AN DUY NGUYEN received the B.Eng. degree in electrical engineering from the Thai Nguyen University of Technology, Thai Nguyen City, Vietnam, in 2018. He is currently pursuing the M.S. degree with Soongsil University, Seoul, South Korea. His research interests include robotic perception and sensor calibration.

MYUNGSIK YOO received the B.S. and M.S. degrees in electrical engineering from Korea University, Seoul, South Korea, in 1989 and 1991, respectively, and the Ph.D. degree in electrical engineering from the State University of New York, Buffalo, NY, USA, in 2000. He was a Senior Research Engineer at the Nokia Research Center, Burlington, MA, USA. He is currently a Full-Time Professor with the School of Electronic Engineering, Soongsil University, Seoul. His research interests include visible-light communications, cloud computing, and internet protocols.