CalibDNN: Multimodal Sensor Calibration for Perception Using Deep Neural Networks

Ganning Zhao\textsuperscript{a}, Jiesi Hu\textsuperscript{a}, Suya You\textsuperscript{b}, and C.-C. Jay Kuo\textsuperscript{a}

\textsuperscript{a}University of Southern California, 3551 Trousdale Pkwy, Los Angeles, CA, USA 90089
\textsuperscript{b}U.S. Army Research Laboratory, 12025 E Waterfront Dr. Los Angeles, CA 90094

ABSTRACT

Current perception systems often carry multimodal imagers and sensors such as 2D cameras and 3D LiDAR sensors. To fuse and utilize the data for downstream perception tasks, robust and accurate calibration of the multimodal sensor data is essential. We propose a novel deep learning-driven technique (CalibDNN) for accurate calibration among multimodal sensor, specifically LiDAR-Camera pairs. The key innovation of the proposed work is that it does not require any specific calibration targets or hardware assistants, and the entire processing is fully automatic with a single model and single iteration. Results comparison among different methods and extensive experiments on different datasets demonstrates the state-of-the-art performance.

Keywords: Multimodal sensor calibration, Sensor fusion, Deep neural network, Intelligent perception

1. INTRODUCTION

The ability to leverage simultaneously multimodal sensor information is critical for many intelligent perception tasks including autonomous driving, robot navigation, and sensor-driven situational awareness. Modern perception systems often carry multimodal imagers and sensors such as electro-optical/infrared (EO/IR) cameras and LiDAR sensors, with future expectation of additional modalities. Robust and accurate estimation of their extrinsic (and intrinsic) parameters is essential to fuse and utilize the multimodal data for downstream perception tasks. While a number of techniques have been proposed, specifically for the LiDAR-camera registration problem, most of the existing methods\textsuperscript{1–4} are target-based that require specific environment with targets, complex setups and significant amounts of manual efforts. In addition, dynamic, online calibration of the deviations caused by sensor vibrations and environmental changes are difficult with existing methods. Target-less methods have been proposed,\textsuperscript{5–9} but they still require accurate initialization, parameters fine-tuning or precise motion estimates and significant amounts of data.

In this work, we develop a new deep learning-driven technique for accurate calibration of LiDAR-Camera pair, which is completely data-driven, does not require any specific calibration targets or hardware assistants, and the entire processing is end-to-end and full automatic. Recent application of deep learning in sensor calibration has shown promising results.\textsuperscript{10–17} We leverage these recent achievements and propose a deep learning-based technique CalibDNN. We model the data calibration as parameter regression problem and utilize advanced deep neural network to align accurately the LiDAR point cloud to the image to regress 6DoF extrinsic calibration parameters. Geometric supervisions, including depth map loss and point cloud loss, and transformation supervision are employed to guide the learning process to maximize the consistency of input images and point clouds. Given LiDAR-Camera pairs, the system automatically learns meaningful features, infers modal cross-correlations, and estimates the 6DoF rigid body transformation between the 3D LiDAR and 2D camera in real-time.

Our main contributions are:

1) Proposing a novel network architecture for LiDAR-Camera calibration. The system is simple with a single model and single iteration, which can also be extended into multi-iterations, dealing with larger miscalibration ranges.

Further author information: (Send correspondence to Ganning Zhao)
Ganning Zhao: E-mail: ganningz@usc.edu, Telephone: 1 213 245 0169
(2) Defining powerful geometric and transformation supervisions to guide the learning process to maximize the consistency of multimodal data.

(3) Further pushing the deep learning-based calibration to the real-world application via applying it on a challenging dataset with complex and diverse scenes.

### 2. RELATED WORKS

Different methods have been proposed to solve the problem of multimodal sensor calibration. Traditional methods can be categorized into target-based and target-less techniques.

For target-based techniques, Zhang and Pless\(^1\) proposed an extrinsic calibration approach between a camera and a laser range finder. They made a checkerboard visible to both camera and laser range finder in a scene, to obtain many constraints of calibration parameters under different poses of the checkerboard, and then solved the parameters by minimizing algebraic error and a re-projection error. Using several checkerboards as targets, Geiger et al.\(^2\) presented an automatic extrinsic calibration method between camera-camera and range sensor-camera. They needed a single shot for each sensor and a particular calibration setup. Using a novel 3D marker, Velas et al.\(^3\) proposed a coarse to fine approach for pose and orientation estimation. Guindel and Beltrán et al.\(^4\) employed a target with four symmetrical circular holes and then utilized two stages of segmentation and registration.

One of the first target-less techniques was proposed by Scaramuzza et al.\(^5\) They manually selected corresponding points from the overlap view between 3D LRF and camera and then used the PnP algorithm followed by a nonlinear refinement. With no need for human labeling, Levinson and Thrun\(^6\) proposed another online target-less calibration method. They optimized the alignment between the depth discontinuities in Laser with image edges by a grid search, when a sudden miscalibration was detected under a probabilistic monitoring algorithm. Pandey et al.\(^7\) proposed a mutual information (MI) based target-less calibration algorithm. They estimated parameters by maximizing MI, using Barzilai-Borwein steepest gradient ascent. Based on motion-based techniques, Ishikawa and Oishi et al.\(^8\) extended the hand-eye calibration framework to camera-LiDAR calibration. They first initialized parameters from camera motion and LiDAR motion, and then iteratively alternated camera motion and extrinsic parameters by sensor-fusion odometry until convergence. Park et al.\(^9\) further improved the motion-based method by introducing a structureless stage where 3D features were computed from triangulation. Although parameters initialization and sensor overlap were not required in these methods, the performance still depended on motion estimates and a large number of data.

With the recent rapid development of deep learning, its application to computer vision have shown a tremendous success. Its application of extrinsic calibration is a new topic that also makes great progress. Kendall et al.\(^10\) proposed the PoseNet that used a convolutional neural network to regress on the camera location and orientation. Schneide et al.\(^11\) presented the RegNet, in which they used three parts, including feature extraction, feature matching, and global regression, to regress on extrinsic parameters, based on Network in Network. Iyer et al.\(^12\) proposed CalibNet using ResNet and the same idea of three parts with RegNet. Instead of directly regress on extrinsic parameters, they utilized a geometrically supervised method with photometric loss and point cloud loss. Based on the success of RegNet and CalibNet, Shi et al.\(^14\) and Lv et al.\(^15\) proposed calibration methods by adding cost volume and Recurrent Convolutional Neural Network. SOIC, proposed by Wang et al.\(^16\) employed semantic centroids to convert the initialization problem to the PnP problem. They also used the cost function based on the correspondence of semantic elements between image and point cloud. Yuan et al.\(^17\) proposed the RGGGNet to utilize a deep generative model and Riemannian geometry for online calibration.

### 3. METHODOLOGY

We aim to design an end-to-end model for multimodal sensor calibration, which paves the way for downstream scene understanding tasks, for example, semantic segmentation. With point clouds from a LiDAR and RGB images from a camera as input pairs, the calibration output is the 6-DoF extrinsic parameters, which define the orientation and translation between the LiDAR and camera sensors. In this section, we will explain the details of the method, including system overview, data preprocessing, network architecture, loss function, and extend to iterative refinement method.
3.1 System overview

Figure 1 shows the system overview of the proposed CalibDNN. The output is the 6-DoF parameters, including a 3-DoF rotation vector and a 3-DoF translation vector. We take the point cloud obtained from the LiDAR and the correspondent RGB image obtained from a camera as the input. We do not know the extrinsic parameters between the LiDAR and camera in the real world, but we do need pre-calibrated data samples to generate ground truth data for training network model. Therefore, when projecting the point cloud onto the image plane, the so-called depth map, we add a random transformation in the extrinsic parameters to get a miscalibrated depth map used for model training. After which, we feed the miscalibrated depth map and the correspondent RGB image into the calibration network to predict the rotation and translation vectors, which will be used to compute the transformation loss. We also apply the predicted transformation matrix, converted from predicted vectors, onto the input depth maps and ground truth transformation onto the input depth maps to compute the depth map loss. We can also obtain the ground truth and predicted point clouds through back-projection from the correspondent depth maps to calculate the point cloud loss.

3.2 Data preprocessing

As discussed in the last subsection, the inputs are pairs of the point clouds and RGB images. We need to convert the 3-D point cloud to the 2-D depth map so that each pixel in the depth map represents the distance information, from the camera to the correspondent point in the real world. Given intrinsic parameters \( P \) and extrinsic parameters \( T \) the projection formula is defined as:

\[
y = P \cdot T \cdot x
\]  

(1)

Where \( x \) represents 3-D points in a point cloud and \( y \) represents correspondent 2-D points in the converted depth map.

Our method requires calibrated data pairs of image-point clouds, or image-depth map, for training network. To obtain the uncalibrated depth map, we intendedly miscalibrate the calibrated depth map by applying a random transformation \( T_{\text{rand}} \) and use as the input depth map. Therefore, the extrinsic parameters after adding the random transform are \( T_{\text{rand}} \cdot T \), and the input depth map, projected from the point cloud, is defined as:

\[
y_{\text{rand}} = P \cdot T_{\text{rand}} \cdot T \cdot x
\]  

(2)

\[
T_{\text{rand}} = \begin{pmatrix}
R_{\text{rand}} & t_{\text{rand}} \\
0 & 0 & 1
\end{pmatrix}
\]  

(3)
Given the miscalibrated depth map as input, the ground truth depth map is defined as:

\[ y_{gt} = P \cdot T_{rand}^{-1} \cdot P^{-1} \cdot y_{rand} \]  


(4)

Therefore, the target is to regress on the ground truth transformation \( T_{gt} = T_{rand}^{-1} \).

Since the converted depth map is often too sparse to extract feature information, we apply data interpolation of max-pooling operation onto the sparse depth map to produce a semi-sparse depth map.

### 3.3 Network architecture

As shown in Figure 2, there are two parts in the calibration network, including feature extraction and feature aggregation. Each part will be introduced in detail in this section.

**Feature Extraction**: There are two network branches to extract features from the RGB image and depth map separately. Thanks to the recent success of ResNet, we use ResNet-18 as the architecture to extract features from the two branches. The two branches are symmetric and initialized by pre-trained weights since, as mentioned in the research works, pre-trained weights can also boost feature extraction on the depth map. Additionally, feature relevance can be preserved using the same initialization and architecture, which contributes to feature matching in the aggregation part. To apply the pre-trained weights on the one channel depth map, we initialize the filter weights of the first convolutional layer by the mean weights along three channels.

**Feature Aggregation**: Obtaining features from two branches, we concatenate the extracted features along the channel dimension and input them into the aggregation network. This part is as important as the former part, as accurate prediction is highly dependent on the feature matching power, so exquisite network design is critical. Inspired by the architecture of ResNet, we concatenate two layers with a similar structure as layers in ResNet. Unlike ResNet, to reduce dimension, we use half the number of channels in the second layer. Then we input the features into a convolutional layer for further feature matching and dimension reduction. Finally, we decouple the output into two identical branches to predict rotation and translation vectors separately to predict extrinsic parameters. In each branch, we use a convolutional layer, with \( 1 \times 1 \) filters, to keep structural feature information and a fully connected layer to predict the vectors. Compared with using one branch to predict the dual quaternion, the performance of separate vector prediction is better since the separate convolutional layer and fully connected layer can automatically learn different translation and rotation information. Without pre-trained weights, we initialize the weights by He-Normal, which gives a more efficient prediction.

**Model generalization**: The system can also be generalized into different input image sizes by image downsampling or change network architecture. For example, given a larger input image size, we can add some average pooling layers in the feature aggregation part. Instead of using max-pooling layers, correspondent information along the channel is kept by average pooling. Some different input image size experiments are conducted on the RELLIS-3D dataset. Given a smaller input image size, we can eliminate the gray convolutional layer in Figure 2 to fit the image size.

**Parameters conversion**: With the predicted rotation vector \( r = (r_x, r_y, r_z)^T \) and translation vector \( t = (t_x, t_y, t_z)^T \), we need to convert them to the transformation matrix for further loss function calculation. The rotation vector can be directly used as a translation term in the transformation matrix. However, the rotation vector needs to be converted to rotation matrix \( R \in SO(3) \) by the well-known Rodrigues’ rotation formula:

\[ R = e^\tilde{r} = I + \frac{\tilde{r}}{||\tilde{r}||} \sin \theta + \frac{\tilde{r}^2}{||\tilde{r}||^2} (1 - \cos \theta) \]  

\[ (5) \]

Where \( I \) is an identity matrix, \( \tilde{r} \) is the antisymmetric matrix of rotation vector \( r \), and \( \theta \) is a rotation angle. Combined with the translation vector \( t \), we get the predicted transformation matrix \( T_{pred} \in SE(3) \):

\[ T_{pred} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \]  

\[ (6) \]
3.4 Loss function

We use the weighted sum of three types of loss functions as the total loss, including transformation loss $L_t$, depth map loss $L_d$ and point cloud loss $L_p$. The total loss function is defined as follow:

$$L_{total} = \lambda_t L_t + \lambda_d L_d + \lambda_p L_p$$  \hspace{1cm} (7)

Where $\lambda_t$, $\lambda_d$, and $\lambda_p$ denote the respective loss weight.

**Transformation loss**: The target is to regress on the rotation and translation vectors, which are the output of the calibration network. Therefore, we compute the L-2 norm between the prediction and the ground truth separately on the rotation vector and translation vector.

$$L_t = \alpha ||r_{pred} - r_{gt}|| + ||t_{pred} - t_{gt}||$$  \hspace{1cm} (8)

Where $r_{pred}$ is the predicted rotation vector, $r_{gt}$ is the ground truth rotation vector, $t_{pred}$ is the predicted translation vector, and $t_{gt}$ is the ground truth translation vector. Since there is a deviation between the scale of rotation L-2 norm and translation L-2 norm, we add a scalar $\alpha$ to control the rotation term.

**Depth map loss**: Given predicted transformation matrix $T_{pred}$, we apply the transformation to the input depth map and obtain the predicted depth map. For the input depth map with a significant miscalibration deviation, if we directly apply the predicted transformation to them, there will be a large blank area because those points projected outside the image are missing. We apply the transformation to the conversion between the point cloud data and depth map to recover the missing points. Employing this approach, we can directly input the point cloud into the system instead of converting the point cloud to the depth map first. Below formula shows the relationship between the predicted and ground truth depth maps:

$$y_{pred} = P \cdot T_{pred} \cdot T_{rand} \cdot T \cdot x$$  \hspace{1cm} (9)

$$y_{gt} = P \cdot T_{rand}^{-1} \cdot T_{rand} \cdot T \cdot x$$  \hspace{1cm} (10)
Where \( x \) is the 3D points in point cloud, \( y_{\text{pred}} \) is the correspondent 2D points in predicted depth map, \( y_{\text{gt}} \) is the correspondent 2D points in ground truth depth map, and \( T_{\text{rand}} \) is the random transformation.

The depth map loss between predicted and ground truth depth maps is defined as:

\[
L_d = \frac{1}{N} \sum_{i=1}^{N} (y_{\text{pred}} - y_{\text{gt}})^2
\]  

(11)

Where \( N \) is the number of pixels in the depth map.

**Point cloud loss:** Given intrinsic camera parameters, we can back-project the depth map to the point cloud data. Thus, we can get predicted point cloud and ground truth point cloud by back-projection from predicted depth map and ground truth depth map. We use the Chamfer Distance\(^{23} \) (CD) between these two point clouds as the loss function. Compared with the testing accuracy on Earth Mover’s distance loss (EMD), the performance of using Chamfer Distance loss is better since it preserves unordered information. To keep point cloud loss the same order of magnitudes with transformation loss, we also add scalars to compute the mean minimum distance between two point clouds. The loss function is defined as:

\[
L_p = d_{\text{CD}}(S_{\text{pred}}, S_{\text{gt}}) = \frac{1}{N} \sum_{x \in S_{\text{pred}}} \min_{y \in S_{\text{gt}}} ||x - y||_2^2 + \frac{1}{M} \sum_{y \in S_{\text{gt}}} \min_{x \in S_{\text{pred}}} ||x - y||_2^2
\]  

(12)

Where \( S_{\text{pred}} \) is the predicted point cloud, \( S_{\text{gt}} \) is the ground truth point cloud, \( N \) is the number of points in \( S_{\text{pred}} \) and \( M \) is the number of points in \( S_{\text{gt}} \).

### 3.5 Iterative refinement

We find that there is a limitation on the optimization process during training. The mean rotation and translation errors can only be reduced by a specific range, which means, when the miscalibration value is large enough, the mean errors cannot be reduced to a value small enough. However, although our model is a single iteration, it still can be extended to multi-iteration to boost performance when miscalibration ranges are significant. We can apply iterative refinement to improve the prediction model. We train different networks by different miscalibration ranges, specifically from large to small ranges. After that, given the input pairs with large miscalibration, we test it by feeding them into the models pre-trained from large miscalibration to small miscalibration, step by step. The process of iterative refinement is shown in Figure 3.

In Fig. 3, \( (r_i, t_i) \) are the miscalibration ranges in the form of rotation and translation vectors that meet the conditions of \( r_0 > r_1 > r_2 > \cdots > r_i \) and \( t_0 > t_1 > t_2 > \cdots > t_i \). \( T_i \) is the predicted transformation of each network. Thus, the final predicted transformation is defined as:

\[
T_{\text{pred}}^{-1} = T_0^{-1} T_1^{-1} T_2^{-1} \cdots
\]  

(13)

Figure 3. Iterative refinement
4. EXPERIMENTS AND DISCUSSIONS

4.1 Dataset preparation
We use the KITTI dataset, the most popular dataset used in autonomous driving research. We utilize the raw data from the KITTI dataset, including RGB images obtained from the PointGray Flea2 color camera and point clouds obtained from the Velodyne HDL-64E rotating 3D laser scanner. We use the synchronized and rectified version of sequence ‘2011_09_26’, with image resolution 1242 × 375. There are different types of scenes in this sequence, including city, residential, and road, which give considerable information for network training. The dataset is already calibrated by a traditional method. As discussed in subsection 3.2, to get the uncalibrated image pairs, we apply a random transformation to the depth map. To be consistent with the CalibNet, we generate 24,000 pairs of training images and 6,000 pairs of testing images, and use miscalibration in the range of (−10°, +10°) on rotation and (−0.25m, +0.25m) on translation. To test the generalization power of our method, we also use the sequence ‘2011_09_30’, with image resolution 1226 × 367, on which we apply zero paddings to warp the image into the same size with sequence ‘2011_09_26’. Different sequences are captured from different dates and scenes, with individual extrinsic parameters.

Compared with the urban scene in the KITTI dataset, the RELLIS-3D dataset is collected in an off-road environment, which provides strong texture, complex terrain, and unstructured features, with grass, bush, forest, and soil. It is challenging for most current algorithms that are mainly trained for urban scenes. We use RGB images obtained from the Basler acA1920-50gc camera and correspondent point clouds obtained from the Ouster OS1 LiDAR. To keep consistent with the image size of the KITTI dataset, we downsample the original images and depth maps, with the size of 1200 × 1920, into the size of 1242 × 375. We generate a training set of 20,000 pairs and a testing set of 5,000 pairs by the same miscalibration range as in the KITTI dataset.

4.2 Training details
We use the Tensorflow library to build the network. To accelerate network training and prevent overfitting, we use Batch normalization after each convolutional block and Dropout at fully connected layers. We set the dropout parameter as 0.5. We also add an L-2 regularization term when training on the Rellis-3D dataset. We use Adam Optimizer and set the parameters as the suggested values $\beta_1=0.9$, $\beta_2=0.999$ and $\varepsilon=1e^{-8}$. We use an initial learning rate $1e^{-3}$ and reduce it every a few epochs. For the loss function, we set the initial $\lambda_t$ equal to 4, $\lambda_d$ equal to 1 and $\lambda_p$ equal to 40 to keep them on a similar scale. Then, we keep $\lambda_d$ unchanged and slowly reduce $\lambda_t$ and $\lambda_p$. We train 25 epochs in total. We train our model on a 2 GPUs sever machine.

4.3 Results and evaluations
Figure 4 shows examples of calibration results in different scenes, which shows that our model can accurately calibrate the point cloud and RGB image in different scenes, from small to large miscalibration ranges.

To fairly compare with CalibNet, we evaluate the model performance by the geodesic distance over rotation transformation and absolute error over translation separately, which are defined as:

$$
\varepsilon_r = \frac{1}{\sqrt{2}} \| \log(R_{\text{pred}}^TR_{\text{gt}}) \|_F, \varepsilon_t = \| t_{\text{pred}} - t_{\text{gt}} \|
$$

(14)

Where $\varepsilon_r$ is the geodesic distance of the rotation matrix, $R_{\text{pred}}$ is the predicted rotation matrix, and $R_{\text{gt}}$ is the ground truth rotation matrix; $\varepsilon_t$ is the absolute error of the translation vector, $t_{\text{pred}}$ is the predicted translation vector, and $t_{\text{gt}}$ is the ground truth translation vector.

We report a mean absolute error (MAE) for rotation prediction of (Roll: 0.11, Pitch: 0.35, Yaw: 0.18) and translation prediction of (X: 0.038, Y: 0.018, Z: 0.096) on the testing set. Figure 5 shows the evaluation curve, in which (a) shows the low rotation absolute errors over the large range of miscalibration, and (c) shows the geodesic distances of most instances concentrates near 0. (b) shows the low translation absolute errors against the large range of miscalibration, and (d) shows the absolute translation errors of most instances concentrates near 0.
Comparison with the state-of-the-art methods: Table 1 shows the performance comparison among our methods, RegNet, CalibNet, and CalibRCNN. In RegNet, authors use iterative refinement, for which they estimate the extrinsic parameters from different networks, trained with miscalibration ranges from large to small, iteratively. Also, the whole miscalibration range of RegNet is larger than ours, which is \((-1.5m, 1.5m)/(\pm 20^\circ, 20^\circ)\); therefore, it is hard to make a rigorous comparison between our CalibDNN and RegNet. We try to keep consistency with CalibNet on the experiment conditions, including the same miscalibration ranges and the same number of training samples, which lead to comparable results between our method and the CalibNet. Although it can be extended to multi-iteration, our current method uses only one model with one iteration, but CalibNet uses multiple models with two strategies to fine-tune the translation values after obtaining the rotation values in the first model. One strategy is given ground truth rotation parameters, they predict the translation parameters separately (the 2nd row in Table 1); the other is given rotation estimation, they predict the translation parameters separately by an iterative re-alignment methodology (the 3rd row in Table 1). It is fair to compare our results with the second strategy of CalibNet. We also list the result of CalibRCNN, which is also a single model, and they use the same miscalibration ranges as us.

Therefore, we focus on the comparison among the results in the last three rows in Table 1, in which the best results are indicated by boldface. Our method CalibDNN in the last row outperforms the other two methods in almost all axes, especially the pitch and X-axis. To be more specific, compared with CalibNet, our method is simpler with just a single model and single iteration but achieves better performance. Compared with CalibRCNN, our method is still far beyond its performance. The excellent performance comes from the exquisite design of network architecture and reasonable training strategy.

Testing on urban scene dataset: To evaluate the generalization ability of the proposed model, we use the KITTI dataset of sequence ‘2011_09_30’ and Rellis-3D dataset as the testing sets and evaluate the model performance trained on the KITTI sequence ‘2011_09_26’. Figure 6 shows the results of KITTI sequence ‘2011_09_30’ in different scenes, with the same results layout with Figure 5. We observe that the misalignment is rectified in prediction, which implies the accurate prediction of extrinsic parameters. For sequence ‘2011_09_30’, we report the MAE rotation prediction of (Roll: 0.15, Pitch: 0.99, Yaw: 0.20) and translation prediction of (X: 0.055, Y: 0.032, Z: 0.096). The MAE is slightly larger because of the different scenes. For Rellis-3D, the MAE rotation prediction is (Roll: 1.40, Pitch: 3.44, Yaw: 2.33) and translation prediction is (X: 0.101, Y: 0.121, Z: 0.186).
The performance is worse since the training scenes are significantly different with the testing scenes, from urban to the field. This scene is challenging for calibration because of the complex and unstructured feature.

*Training on terrain scene dataset:* Since the scene of KITTI and Rellis-3D is totally different, one is the urban environment, and the other is the off-road environment, we re-train the model on Rellis-3D and show the testing results in different scenes in Figure 7, with the same layout as Figure 5. Although there is a visually detected deviation between prediction and ground truth, the misalignment of the input depth map is still rectified significantly. We report the MAE rotation prediction of (Roll: 1.00, Pitch: 2.57, Yaw: 1.94) and translation prediction of (X: 0.092, Y: 0.074, Z: 0.082). Compared with the MAE of Rellis-3D in last subsection, the performance is much better. Although not as good as on the KITTI dataset, the performance is still acceptable since the original miscalibration is still reduced significantly. Also, we find the original calibration of Rellis-3D is not very accurate, which also leads to a large MAE. We believe our CalibDNN rectifies the inaccurate calibration to some extent.

| Mean abs. err. | Rotation | Translation |
|----------------|----------|-------------|
|                | Roll     | Pitch       | Yaw   | X     | Y     | Z     |
| RegNet         | 0.24     | 0.25        | 0.36  | 0.070 | 0.070 | 0.040 |
| CalibNet       | 0.15     | 0.90        | **0.18** | 0.120 | 0.016 | 0.072 |
| CalibRCNN      | 0.19     | 0.64        | 0.44  | 0.062 | 0.043 | **0.054** |
| CalibDNN       | **0.11** | **0.35**    | **0.18** | **0.038** | **0.018** | 0.096 |
5. CONCLUSION

This paper proposes a novel approach for multimodal sensor calibration to predict the 6 DoF extrinsic parameters between the RGB camera and 3D LiDAR sensor. The calibration process is critical for multimodal scene perception since an accurate calibration paves the way for later information fusion. Unlike traditional calibration techniques, we utilize advanced deep learning to solve the challenging sensor calibration problem. The developed approach is fully data-driven, end-to-end, and automatic. The network model is simpler comparing with existing state-of-art methods with a single model and single iteration, combining the merits of geometric supervision and regression on transformation; the performance of our model is state-of-the-art. Given miscalibration range (-10°,10°) in rotation and (-0.25m, 0.25m) in translation, the model achieves a mean absolute error of 0.21° in rotation and 5.07cm in translation. Training and testing on challenging datasets are conducted, to demonstrate the value and utility of the developed approach to the real world applications. In the future, we will further improve the performance and apply iterative refinement to deal with a larger miscalibration range. Additionally, we will explore the non-DNN, non-backpropagation learning network which is also an interesting topic.
ACKNOWLEDGMENTS

This work was supported by US Army Artificial Intelligence Innovation Institute (A2I2). Computation for the work was supported by the University of Southern California’s Center for High Performance Computing (https://carc.usc.edu/).

REFERENCES

[1] Zhang, Q. and Pless, R., “Extrinsic calibration of a camera and laser range finder (improves camera calibration),” in [2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)/(IEEE Cat. No. 04CH37566)], 3, 2301–2306, IEEE (2004).

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

[3] Velás, M., Španěl, M., Materna, Z., and Herout, A., “Calibration of rgb camera with velodyne lidar,” (2014).

[4] Guindel, C., Beltrán, J., Martín, D., and García, F., “Automatic extrinsic calibration for lidar-stereo vehicle sensor setups,” in [2017 IEEE 20th international conference on intelligent transportation systems (ITSC)], 1–6, IEEE (2017).

[5] Scaramuzza, D., Harati, A., and Siegwart, R., “Extrinsic self calibration of a camera and a 3d laser range finder from natural scenes,” in [2007 IEEE/RSJ International Conference on Intelligent Robots and Systems], 4164–4169, IEEE (2007).

[6] Levinson, J. and Thrun, S., “Automatic online calibration of cameras and lasers..,” in [Robotics: Science and Systems], 2, 7, Citeseer (2013).

[7] Pandey, G., McBride, J. R., Savarese, S., and Eustice, R. M., “Automatic extrinsic calibration of vision and lidar by maximizing mutual information,” Journal of Field Robotics 32(5), 696–722 (2015).

[8] Ishikawa, R., Oishi, T., and Ikeuchi, K., “Lidar and camera calibration using motions estimated by sensor fusion odometry,” in [2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)], 7342–7349, IEEE (2018).

[9] Park, C., Moghadam, P., Kim, S., Sridharan, S., and Fookes, C., “Spatiotemporal camera-lidar calibration: a targetless and structureless approach,” IEEE Robotics and Automation Letters 5(2), 1556–1563 (2020).

[10] Kendall, A., Grimes, M., and Cipolla, R., “Posenet: A convolutional network for real-time 6-dof camera relocalization,” in [Proceedings of the IEEE international conference on computer vision], 2938–2946 (2015).

[11] Schneider, N., Piewak, F., Stiller, C., and Franke, U., “Regnet: Multimodal sensor registration using deep neural networks,” in [2017 IEEE intelligent vehicles symposium (IV)], 1803–1810, IEEE (2017).

[12] Lin, M., Chen, Q., and Yan, S., “Network in network,” arXiv preprint arXiv:1312.4400 (2013).

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

[14] Shi, J., Zhu, Z., Zhang, J., Liu, R., Wang, Z., Chen, S., and Liu, H., “Calibrcnn: Calibrating camera and lidar by recurrent convolutional neural network and geometric constraints,” arXiv preprint arXiv:2012.13901 (2020).

[15] Wang, W., Nobuhara, S., Nakamura, R., and Sakurada, K., “Soic: Semantic online initialization and calibration for lidar and camera,” arXiv preprint arXiv:2003.04260 (2020).

[16] Yuan, K., Guo, Z., and Wang, Z. J., “Rregnet: Tolerance aware lidar-camera online calibration with geometric deep learning and generative model,” IEEE Robotics and Automation Letters 5(4), 6956–6963 (2020).

[17] He, K., Zhang, X., Ren, S., and Sun, J., “Deep residual learning for image recognition,” in [Proceedings of the IEEE conference on computer vision and pattern recognition], 770–778 (2016).

[18] Schneider, L., Jasch, M., Fröhlich, B., Weber, T., Franke, U., Pollefeys, M., and Rätsch, M., “Multimodal neural networks: Rgb-d for semantic segmentation and object detection,” in [Scandinavian conference on image analysis], 98–109, Springer (2017).
[20] Hu, X., Yang, K., Fei, L., and Wang, K., “Acnet: Attention based network to exploit complementary features for rgbd semantic segmentation," in [2019 IEEE International Conference on Image Processing (ICIP)], 1440–1444, IEEE (2019).

[21] He, K., Zhang, X., Ren, S., and Sun, J., “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in [Proceedings of the IEEE international conference on computer vision], 1026–1034 (2015).

[22] Jiang, P., Osteen, P., Wigness, M., and Saripalli, S., “Rellis-3d dataset: Data, benchmarks and analysis,” arXiv preprint arXiv:2011.12954 (2020).

[23] Fan, H., Su, H., and Guibas, L. J., “A point set generation network for 3d object reconstruction from a single image,” in [Proceedings of the IEEE conference on computer vision and pattern recognition], 605–613 (2017).

[24] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R., “Vision meets robotics: The kitti dataset,” The International Journal of Robotics Research 32(11), 1231–1237 (2013).

[25] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al., “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” arXiv preprint arXiv:1603.04467 (2016).

[26] Ioffe, S. and Szegedy, C., “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in [International conference on machine learning], 448–456, PMLR (2015).

[27] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R., “Dropout: a simple way to prevent neural networks from overfitting,” The journal of machine learning research 15(1), 1929–1958 (2014).

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