2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

CalibDepth: Unifying Depth Map Representation for Iterative LiDAR-Camera Online Calibration

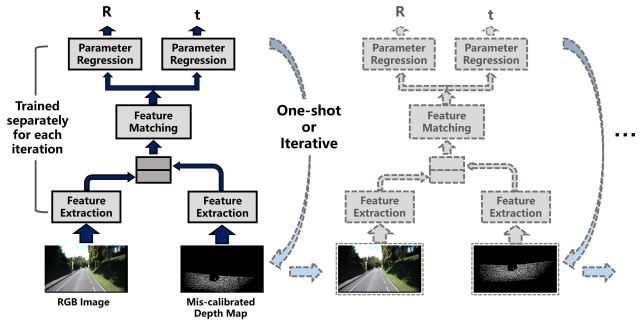
Jiangtong Zhu[1], Jianru Xue[1] and Pu Zhang[1]

***Abstract*— LiDAR-Camera online calibration is of great significance for building a stable autonomous driving perception system. For online calibration, a key challenge lies in constructing a unified and robust representation between multimodal sensor data. Most methods extract features manually or implicitly with an end-to-end deep learning method. The former suffers poor robustness, while the latter has poor interpretability. In this paper, we propose CalibDepth, which uses depth maps as the unified representation for image and LiDAR point cloud. CalibDepth introduces a sub-network for monocular depth estimation to assist online calibration tasks. To further improve the performance, we regard online calibration as a sequence prediction problem, and introduce global and local losses to optimize the calibration results. CalibDepth shows excellent performance in different experimental setups. Code is open-sourced at https://github.com/Brickzhuantou/CalibDepth.**

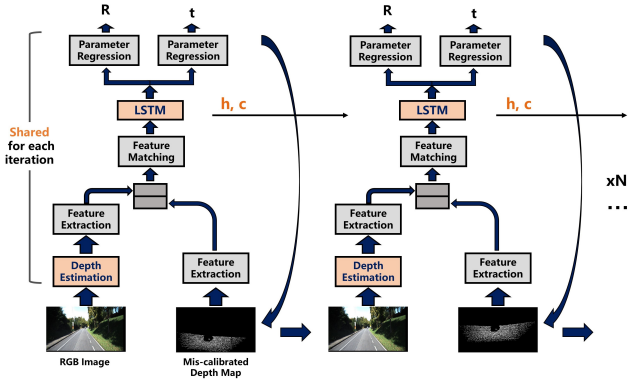
I. INTRODUCTION

LiDAR and camera are the two most widely used sensors in autonomous driving perception systems. LiDAR point clouds can obtain accurate positions, while image data includes rich visual cues. Fusion of the two modalities can largely facilitate downstream perception tasks such as detection [1]–[3] and segmentation [4], [5]. In the process of multimodal data fusion, the calibration matrix plays an important role. Accurate calibration results can facilitate the spatial alignment of point clouds and images at the data or feature levels, thereby obtaining accurate and robust fusion perception results.

The goal of the calibration task is to find the transformation matrix between the LiDAR and camera coordinate system. The methods can be divided into two categories: offline calibration [6]–[10] and online calibration [11]–[23]. Offline calibration relies on specific calibration markers and requires human interaction for feature selection, which is time-consuming and cannot automatically correct disturbance deviations in real time. For online calibration, traditional methods replace the feature extraction part of offline calibration with some learning-based methods to extract salient features from multimodal sensor data. However, the robustness of these features is generally not good enough to guarantee stability in different environments. In recent years, end-to-end methods [14]–[20] based on deep learning have demonstrated a powerful ability, which perform calibration tasks by adding perturbations to the calibration parameters of existing datasets and then training a neural network to predict the bias. Most of the deep-learning-based methods follow the basic architecture shown in Fig. 1a to design the network. Taking the original image and mis-calibrated depth map as input, the whole process is generally divided into feature extraction, feature matching and parameter regression. Most works use a one-shot method [17], [19], [22] for calibration, while some train multiple networks [15], [16], [20] for iterative refinement.



(a) Previous General Architecture of Online Calibration



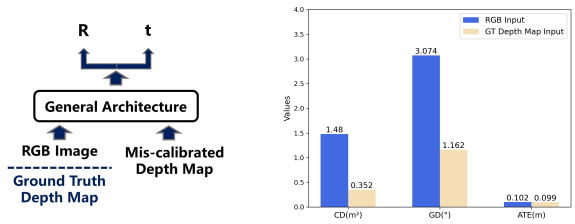
(b) Overview of CalibDepth

Fig. 1. Our CalibDepth introduces a depth estimation module to build a unified depth map representation for multimodal data, while previous architectures extract features directly from the input image. We do iterative refinement in an autoregeressive manner, and introduce LSTM to capture the context information, while previous methods are one-shot or iteratively calibrated with multiple pretrained networks.

One of the key points for online calibration is to build a common feature representation, so as to capture the relationships among multimodal features. It is natural to ask: what kind of features are suitable for the LiDAR-camera calibration task? Explicitly extracting features such as edges [12], [13] or semantics [11], [21] suffers poor transferability and robustness, while the features extracted implicitly in an

*This work was supported by STI 2030—Major Projects (No. 2021ZD0201300) and NSFC Projects (No. 62036008).

[1]The authors are with National Key Laboratory of Human-Machine Hybrid Augmented Intelligence, Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, China. jrxue@mails.xjtu.edu.cn



(a) A Simple Test for Depth Input (b) Test Results

Fig. 2. Expirical analysis on the impact of depth map representation for the calibration task. We replace the RGB image in the general pipeline with the ground truth depth map, and (b) shows the performance of these two approaches on the three calibration evaluation metrics: lower is better.

end-to-end manner have the problem of poor interpretability. Compared with edges or semantics, depth map could be more suitable as a unified representation for online LiDAR-camera calibration. On the one hand, it is natural to get accurate depth information from LiDAR point clouds, on the other hand, monocular depth estimation has been widely investigated by previous studies [24]–[34]. Thus, LiDAR and camera data can be easily aligned by the feat of the common depth cues. To preliminarily test the idea, we conduct a simple experiment based on part of KITTI [35] raw dataset. As shown in Fig. 2, we first obtain the ground truth depth map by point cloud projection, then use it to replace the input image of the general architecture in Fig. 1a. Under the same experimental setup, comparison results in Fig. 2b show that Chamfer distance(15) and Geodesic distance(13) are decreased by 76.2% and 62.2% using ground truth depth map as input. This inspires us to estimate accurate depth of the image to improve the calibration accuracy.

We propose CalibDepth, which introduces a sub-network for supervised monocular depth estimation to assist the online calibration task as shown in Fig. 1b. Without relying on additional binocular estimation methods or depth annotation ground truth, we complete the sparse depth map obtained by LiDAR projection as the supervision data for depth estimation and jointly optimize the depth estimation and calibration tasks. In addition, we regard calibration as a sequence prediction task, getting the calibrated action at each step in an autoregressive manner, and use LSTMs [36] to model contextual relationships in sequences.

In conclusion, our main contributions are as follows:

- We propose CalibDepth, an online deep learning calibration model with explicit depth supervision, using depth map as a unified representation between point cloud and image data, showing superior accuracy.
- We model the calibration process as a sequence prediction task, introducing LSTM to capture the contextual relationship and outputting the calibration action of each step in an autoregeressive manner. At the same time, we design local and global loss functions to optimize the sequence prediction results.
- Our model exhibits superior performance under different environmental setups designed for fair comparison. Ablation experiments also demonstrate the benefits of our individual module designs.

II. RELATED WORK

A. Offline Calibration

Offline calibration methods rely on special calibration objects, such as checkerboards [6], [7], spherical targets [9], [10], V-shaped targets [8], [37], and so on. The LiDAR and camera simultaneously capture the salient features of these targets for feature matching, and then use the nonlinear optimization method to regress the parameters. However, offline calibration is often time-consuming due to the need for specific calibrators and human selection for feature matching, and cannot automatically correct the bias in dynamic driving environments [12], [16].

B. Online Calibration

The online calibration method avoids manual feature selection from special calibration objects. It can be divided into two categories according to whether it is end-to-end. Multi-stage methods use learning-based methods to complete feature extraction, and then update calibration parameters based on traditional optimization algorithms [11]–[13], [21], [22], which often suffer the problem of cascading errors. The end-to-end approaches [14]–[20] directly output calibration results based on multimodal sensor data input. This kind of method often suffers from poor interpretability due to the lack of geometric constraints on intermediate features.

Iterative Refinement: Among learning-based methods, one-shot methods [17], [22] are usually difficult to achieve high accuracy. To deal with this problem, some works [18] adopt an iterative training manner, while others [15], [16], [20] refine the result with multiple networks trained under different mis-calibration degrees. Different from these approaches, we propose an autoregeressive iterative strategy with LSTM capturing contextual information, and perform local and global optimization to further improve the calibration accuracy.

Multimodal Data Representation: The representation of multimodal data also has a great influence on the calibration results. Edge [12], [13] and semantic mask [11], [21] features have been widely used for calibration tasks. However, edge features are easily affected by illumination and texture, while the semantic features require rich semantic annotations. In addition, some works try to optimize the calibration task by introducing additional inputs. Guindel et al. [23] propose a method to register both stereo cameras and LiDAR point clouds to optimize the calibration task. Shi et al. [19] introduce multi-frame images for motion estimation to assist in optimizing calibration. NetCalib [22] is the first to take depth map as the unified multimodal representation and achieves excellent results, but it also introduces another camera to calculate pixel depth through binocular disparity. In this paper, we introduce a monocular depth estimation sub-network to generate a unified depth map representation explicitly for end-to-end online calibration, which not only avoids the problem of cascading errors, but also enables the model to obtain better interpretability.

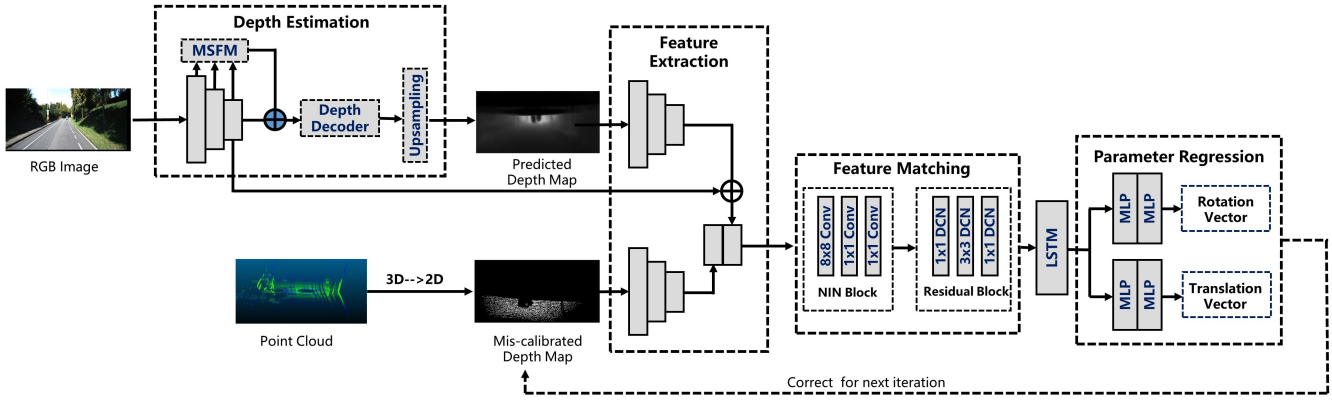


Fig. 3. Detailed network architecture of CalibDepth. The input image and point cloud firstly generate the predicted depth map and the mis-calibrated depth map through the depth estimation module and projective transformation, respectively. Then the unified multimodal depth maps are sent to the feature extraction, feature matching and parameter regression module to generate the rotation and translation vectors corresponding to the target transformation matrix. The predicted transformation matrix of the current iteration is applied to the mis-calibrated depth map for correction, and the corrected depth map will be the input of the next iteration.

C. Monocular Depth Estimation

The goal of the monocular depth estimation task is to predict the depth value for each pixel in an image. According to the training mode, it can be divided into supervised [24]–[27], semi-supervised [28]–[30] and unsupervised [31]–[34] methods. Supervised methods use the depth ground truth as the supervision signal, but the acquisition of dense depth label is generally difficult; unsupervised methods are usually trained with scene geometric constraints between stereo pair-wise images or monocular image sequences. We adopt a semi-supervised training method, that is, the depth estimation training is supervised by the depth completion result of the point cloud projected sparse depth map.

To our best knowledge, we are the first to introduce monocular semi-supervised depth estimation into the online calibration task, which greatly improves the final calibration performance.

III. Method

CalibDepth takes the RGB image and LiDAR point cloud as input, generating unified depth map representation respectively, and the following calibration process includes feature extraction, feature matching and parameter regression. All these processes are performed iteratively to generate highly accurate calibration results. An overview of CalibDepth's architecture is shown in Fig. 1b.

A. Problem Formulation

Taking multimodal data as input, the goal of the online calibration task is to calculate the extrinsic calibration parameters based on a rough initial value. Like [14]–[19], [22], in order to obtain the training data for online calibration, we apply a random transformation matrix T_{rand} to the real calibration matrix T to get the initial matrix T_{init}. Then the point cloud's mis-calibrated depth map can be generated by:

$$Z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K(T_{init}P_H)_{(1\sim3)} = K(R_{init}P + t_{init}) \quad (1)$$

$$T_{init} = \begin{pmatrix} & R_{init} & & t_{init} \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2)$$

where P_H represents the homogeneous coordinates of the 3-D LiDAR point $P = (X, Y, Z)$, and $x = (u, v)$ is the corresponding 2-D projected point. K is the intrinsic matrix of camera and Z_c is an arbitrary scale factor related to depth. The initial transformation matrix T_{init} consists of a rotation matrix R_{init} and a translation vector t_{init}. Then, we take the randomly mis-calibrated depth map as the input of the LiDAR branch, and the goal of online calibration is equivalent to solving T_{rand}^{-1} to correct the error.

B. Network Architecture

In this section, we detail the architecture of CalibDepth. The illustration is shown in Fig. 3.

Depth Map Generation: For the point cloud branch, we follow the method mentioned in section III-A to generate a mis-calibrated depth map. For the image branch, we refer to [34] to design the depth estimation module. ResNet18 [38] without the last two blocks is firstly used to extract multi-scale features of the image, and then MSFM module composed of GRUs is introduced to iteratively complete the depth estimation task with features of different scales. In each iteration, the depth map decoder is composed of multiple convolutional layers, and the decoded results are passed through an upsampling module with learnable parameters to obtain the final required resolution.

Different from [34], which uses geometric constraints between multiple frames for self-supervised depth estimation, we use the depth map results obtained by point cloud projection as the supervised label for image depth estimation. However, since the original depth projection results are relatively sparse, we refer to the method of [39] to complement

the original sparse depth map and provide dense supervision signals for monocular depth estimation tasks.

Feature Extraction: Both the image and point cloud branches have generated unified depth map representations, and this part employs symmetric modules to extract features from their respective depth maps. For the point cloud branch, we use the first three blocks of ResNet18 for feature extraction, and the obtained feature map is 1/8 of the input depth map size. For the image branch, we also use the first three blocks of ResNet18 to extract features from the predicted depth map. In addition, we take the feature map of the same size obtained from the original image in the depth estimation part as a residual and add it to the current features to get the final feature map of the image branch. All the above ResNet18 blocks are initialized with weights pretrained on ImageNet.

Feature Matching: This part aims to capture the deviation between feature maps obtained from multimodal data. Firstly, the multi-modal feature maps obtained in the previous part are concatenated along the channel dimension, and then we use a NIN [40] block and a Residual block for further feature extraction. The NIN block contains multiple 1x1 convolutions, which can be used to keep the structural features information. For the Residual block, we replace all the convolution layers with Deformable Conv [41] to increase the receptive field and deal with different mis-calibration degrees.

Sequence Generative Modeling: The final output of the network is the rotation and translation vectors corresponding to the predicted T_{rand}^{-1}. Since the prediction accuracy obtained by the one-shot method is not high enough, we model the output as an action sequence of length N in an autoregressive manner, as shown in Fig. 1b. In the current n-th iteration, we firstly flatten the feature map obtained by feature matching to get x_n, and then obtain h_n and c_n through a two-layer LSTM module with hidden size 256. h_n is then sent to the parameter regression modules corresponding to rotation and translation, each of which is composed of two full connection layers with size 256×128 and 128×3, respectively. In addition, h_n and c_n are sent to the next iteration as context information.

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} W \begin{pmatrix} h_{n-1} \\ x_n \end{pmatrix} \tag{3}$$

$$c_n = f \odot c_{n-1} + i \odot g \tag{4}$$

$$h_n = o \odot tanh(c_n) \tag{5}$$

The predicted rotation vector $r_n = (r_x, r_y, r_z)^T$ and translation vector $t_n = (t_x, t_y, t_z)^T$ will be converted into a transformation matrix to perform a correction on the mis-calibrated depth map. And the corrected depth map will be used as the input for the next iteration to generate a more refined calibration result. The Rodrigues formula is used to convert the rotation vector r_n to a rotation matrix R_n as follows:

$$R_n = e^{\hat{r}_n} = I + \frac{\hat{r}_n}{||\hat{r}_n||} sin\theta_n + \frac{\hat{r}_n^2}{||\hat{r}_n||^2}(1 - cos\theta_n) \tag{6}$$

where $\hat{r}_n$ is the skew-symmetric matrix form of r_n and θ_n is the rotation angle. Then we can combine R_n with t_n to get the current transformation matrix as the equation (2) has been defined. The final predicted transformation is the combination of each iteration:

$$T_{pred}^{-1} = T_0^{-1} T_1^{-1} ... T_N^{-1} \tag{7}$$

C. Loss Functions

As we introduce the monocular depth estimation to assist the online calibration task, our loss function can be divided into two parts: depth estimation loss and calibration loss.

Depth Estimation Loss: The depth estimation loss measures the deviation of the predicted depth map D from the available ground truth D_{gt} at the pixels. And the dense ground truth depth map is obtained by completing the sparse point cloud projection map with the method of [39]. Inspired by [30], we choose to use berHu loss $||\cdot||_\delta$ to assign more weights to larger depth biases.

$$L_{depth} = \sum_{x \in \Omega_D} ||D(x) - D_{gt}(x)||_\delta \tag{8}$$

$$||d||_\delta = \begin{cases} |d|, d \le \delta \\ \frac{d^2 + \delta^2}{2\delta}, d > \delta \end{cases} \tag{9}$$

$$\delta = \frac{1}{5} \underset{x \in \Omega_D}{max}(|D(x) - D_{gt}(x)|) \tag{10}$$

Calibration Loss: As the calibration task has been modeled as a sequence prediction problem, we introduce local loss and global loss to optimize the final result. On the one hand, the local loss can keep the calibration result of each step as close to the ground truth as possible. On the other hand, the global loss is introduced to ensure that the overall result is accurate enough.

1) Local Loss: In each iteration, we use the SmoothL1 function defined in PyTorch to calculate the difference between the prediction and the ground truth separately on the rotation vector and translation vector. Then we compute the sum of all iteration steps as the local loss.

$$L_i = SmoothL1(r_i, \hat{r}_i) + SmoothL1(t_i, \hat{t}_i) \tag{11}$$

$$L_{local} = \sum_{i=1}^{N} L_i \tag{12}$$

2) Global Loss: Global loss can help avoid divergence between different iteration steps. We define it as the weighted sum of multiple loss functions to optimize the results from different perspectives.

The geodesic distance over SO(3) is computed to represent the difference between the rotation matrices and the absolute translation error is used as the second part of the global loss:

$$L_{GD} = |arccos(\frac{Tr(R^T \hat{R}) - 1}{2})| \cdot \frac{180}{\pi} \tag{13}$$

$$L_t = |t - \hat{t}| \tag{14}$$

where R and t are the predicted rotation matrix and translation vector, while $\hat{R}$ and $\hat{t}$ are the ground truth.

Chamfer distance can be used to evaluate the distance between two point clouds S_1, S_2, which is defined as follows:

$$L_{CD} = \sum_{x \in S_1} \underset{y \in S_2}{min} ||x - y||_2^2 + \sum_{y \in S_2} \underset{x \in S_1}{min} ||x - y||_2^2 \tag{15}$$

The final loss consists of a weighted sum of all the losses mentioned above.

$$\begin{aligned} L_{all} &= \lambda L_{depth} + \omega L_{local} + L_{global} \\ &= \lambda L_{depth} + \omega L_{local} + \alpha L_{GD} + \beta L_t + \gamma L_{CD} \end{aligned} \tag{16}$$

IV. Experiments

We evaluate the proposed calibration approach on the KITTI raw and KITTI odometry datasets. In this section, we detail the dataset preparation, implementation process and analysis the results of different experiments.

A. Dataset Preparation

Previous works use different data processing methods, and we found it would have a great impact on the results. Therefore, in order to make a fair comparison, we use different environmental setups to compare with different works. As shown in Table I, we use the same data processing methods of CalibNet [18] and CalibDNN [17] to generate 24000 data pairs from the sequence "2011_9_26" in KITTI raw for the training of T1, and use the sequence "2011_9_30" for test. T2 is designed for comparison with NetCalib [22], using drive 0013, 0020 and 0079 in "2011_9_26" for validation, drive 0005, 0070 for testing, and the rest for training. T3 keeps the same setup as CalibRCNN [19], taking sequences 00-06 in KITTI odometry for training and "2011_9_26" in KITTI raw for testing. T4 adopts the same data processing method as LCCNet [15], with the sequences 01-20 in KITTI odometry as the training set and 00 as the test set. All the initial mis-calibrated ranges are also matched to the corresponding works, respectively.

B. Implementation Details

Metrics: To facilitate comparison with previous work, we convert the output rotation vector to Euler angles, and then calculate the absolute error between the predicted value and the ground truth in all dimensions of angles and translation vectors, respectively. In the ablation study section, we choose the chamfer distance(CD), geodesic distance(GD) and absolute translation error(ATE) as the evaluation metrics.

Training Details: The original size of the image and mis-calibrated depth map is 1242×375, we resize them to 512×256 as the input of our model. Keeping batch size to 6, we train the proposed network on a single NVIDIA TITAN Xp for a total of 50 epochs. We use the Adam Optimizer [42], with a learning rate initialized to $1e^{-4}$ and decreased by a factor of 0.5 every 8 epochs. The weights of all loss functions are set as: $\lambda = 0.05, \omega = 10, \alpha = 0.1, \beta = 3, \gamma = 0.2$. When compared with LCCNet, we keep the iterative step size at 5, and in other experiments, the step size is set to 3.

C. Comparison to state-of-the-art

Baselines: We chose five different methods as baselines and kept the same experimental configurations as them for comparative experiments.

1) CalibNet: Following the general pipeline, CalibNet [18] introduces photometric and point cloud distance loss to supervise the calibration task. Furthermore, it applies a simple iterative realignment method to refine the result.

2) CalibDNN: CalibDNN [17] is an one-shot method which achieves great performance thanks to the exquisite design of network architecture.

3) NetCalib: NetCalib [22] firstly obtains the depth map with a stereo camera pair and then follows the general pipeline to do calibration.

4) CalibRCNN: CalibRCNN [19] uses LSTM to extract the temporal features between the input consecutive frames and considers the geometric constraints to refine the calibration accuracy.

5) LCCNet: LCCNet [15] takes the correlation layer from PWC-Net [43] for feature matching. To do iterative refinement, it trains five different networks under five different mis-calibration ranges: $(20°, 1.5m)$, $(10°, 1.0m)$, $(5°, 0.5m)$, $(2°, 0.2m)$, $(1°, 0.1m)$. We chose the results of the three-step iteration and five-step iteration as the baselines for comparison.

Results: The comparison results with the baselines are shown in Table II. Our CalibDepth outperforms all baselines under the first three environmental setups, which suggests that the introduction of a depth estimation sub-network and reasonable iterative training design is indeed effective, without using additional sensor information or multi-frame input. LCCNet-5 achieves excellent calibration accuracy with five different pretrained models to do iterative refinement. But the performance comes at the cost of model memory. Just iterating over a single model trained under the largest mis-calibration range, our CalibDepth outperforms LCCNet-3, which takes the first three models of LCCNet-5 to do calibration. In addition, we calculated the number of parameters for both models. The parameters of the first three models officially open sourced by LCCNet are about 210 million, while the total number of CalibDepth's parameters is about 43 million, which means that CalibDepth can achieve higher accuracy than LCCNet-3 with roughly 1/5 the number of parameters.

We set batch size to 1 and test the inference speed on a single NVIDIA TITAN Xp using PyTorch protobuf serving. CalibDepth can achieve 32 FPS for a single iteration of calibration together with depth prediction, which is accepted to run in a real-time manner. The visualization performance of the predicted depth map and the iterative calibration results are shown in Fig. 4, which intuitively reflects the convincing ability of CalibDepth.

D. Ablation studies

Depth Estimation Module: Table III shows the impact of the depth estimation module in CalibDepth. The first line corresponds to CalibDepth without the depth estimation

TABLE I

DIFFERENT ENVIRONMENTAL SETUPS

Name	Dataset	Disturbance range	Training set	Validation set	Test set
T1	KITTI raw	10°,0.25m	24000 pairs in 9-26	/	9-30
T2	KITTI raw	10°,0.2m	9-26 without 0005,0070,0013,0020,0079	0013,0020,0079 in 9-26	0005,0070 in 9-26
T3	KITTI odometry	10°,0.25m	00-06	/	9-26
T4	KITTI odometry	20°,1.5m	01-20	/	00

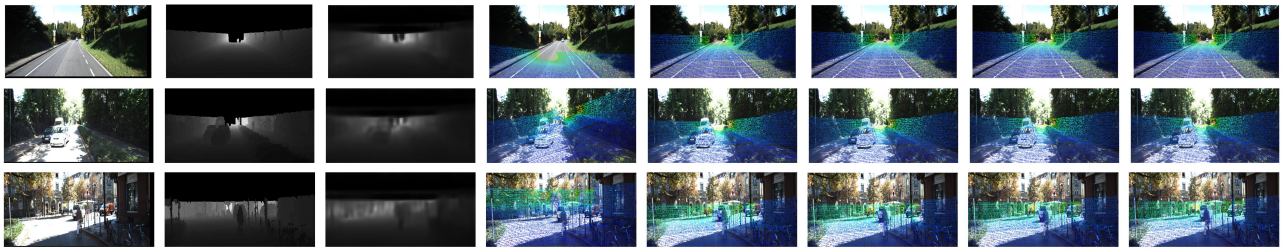


Fig. 4. Visualization of the predicted depth map and the calibration iterative process.

TABLE II

COMPARISON RESULTS WITH OTHER ALGORITHMS

Dataset	Methods	Rotation(°) E_R	Roll	Pitch	Yaw	Translation(cm) E_t	X	Y	Z
T1	CalibNet	0.410	0.180	0.900	**0.150**	7.82	12.10	3.49	7.87
	CalibDNN	0.447	**0.150**	0.99	0.20	6.10	**5.50**	3.20	9.60
	CalibDepth	**0.348**	0.180	**0.682**	0.181	**4.75**	6.66	**1.12**	**6.48**
T2	NetCalib	0.523	0.230	0.970	0.370	2.40	3.86	1.55	1.79
	CalibDepth	**0.401**	**0.114**	**0.955**	**0.133**	**0.79**	**1.07**	**0.58**	**0.73**
T3	CalibRCNN	0.675	0.216	1.330	0.478	5.50	6.60	4.40	5.50
	CalibDepth	**0.377**	**0.113**	**0.913**	**0.147**	**1.45**	**1.80**	**1.08**	**1.48**
T4	LCCNet-3	0.324	0.309	0.330	0.334	1.53	1.27	2.21	1.11
	LCCNet-5	0.030	0.030	0.019	0.040	0.36	0.24	0.38	0.46
	CalibDepth	0.123	0.064	0.226	0.080	1.17	1.31	1.02	1.17

TABLE III

DEPTH ESTIMATION ABLATION

Method	CD(m^2)	GD(°)	ATE(cm)
w/o Depth Estimation	0.395	1.116	0.842
w/ Depth Estimation	**0.297**	**0.992**	**0.792**
w/ Depth Ground Truth	0.015	0.203	0.716

module and extracting features directly from the image. After the introduction of depth estimation, the errors on CD, GD and ATE are reduced by 24.8%, 11.1% and 5.9%, respectively. We also do an experiment to extract features directly from the ground truth depth map, and it is surprised to find that the CD and GD could be reduced by 96.2% and 81.8%. The results reveal that our depth estimation is helpful indeed to increase the calibration accuracy, especially on rotation related metrics. However, there is still a large gap between the predicted depth map and the ground truth, which greatly limits the calibration performance of CalibDepth.

Iteration Step Size: Here we test the effect of the step size on the calibration results in the iterative process. As shown in Table IV, when increasing the step size from 1 to 3, there is a significant drop in all types of errors. However, when the step size is increased to 5, though there is still a slight decline for ATE, it is a struggle for CD and GD to achieve

TABLE IV

STEP SIZE ABLATION

Step Size	CD(m^2)	GD(°)	ATE(cm)
1	0.886	1.877	2.919
3	**0.297**	**0.992**	0.792
5	0.355	1.027	**0.604**

TABLE V

LOSS FUNCTIONS ABLATION

Exp.	L_{depth}	L_{local}	L_{global}	CD(m^2)	GD(°)	ATE(cm)
1		✓		0.441	1.229	**0.742**
2			✓	0.354	1.052	1.233
3		✓	✓	0.378	1.050	0.763
4	✓	✓	✓	**0.297**	**0.992**	0.792

better results. This suggests that longer iteration steps does not necessarily mean better, and compared to rotation, step size is more friendly to translation-related metrics.

Loss Functions: Table V shows the impact of different loss functions on the calibration task. From the results of the first three experiments, we can learn that the rotation related metric is more sensitive to global loss, while the local loss plays an important role in reducing the absolute translation error. Combining global and local losses can achieve better overall results. Experiment 4 adds the depth supervision loss, which further improves the calibration accuracy, especially for the rotation. This further illustrates that explicit deep supervision can lead to improved performance.

V. CONCLUSIONS

By introducing a depth estimation module with explicit supervision and an iterative training strategy, our proposed CalibDepth outperforms most online calibration methods, demonstrating strong accuracy and interpretability. Further experimental results also encourage us to explore better unified depth map representations to achieve more accurate calibration results.

REFERENCES

[1] Y. Li, A. W. Yu, T. Meng, B. Caine, J. Ngiam, D. Peng, J. Shen, B. Wu, Y. Lu, D. Zhou, Q. V. Le, A. Yuille, and M. Tan, "DeepFusion: Lidar-Camera Deep Fusion for Multi-Modal 3D Object Detection," Mar. 2022, number: arXiv:2203.08195 arXiv:2203.08195 [cs].

[2] X. Wu, L. Peng, H. Yang, L. Xie, C. Huang, C. Deng, H. Liu, and D. Cai, "Sparse Fuse Dense: Towards High Quality 3D Detection with Depth Completion," Jul. 2022, number: arXiv:2203.09780 arXiv:2203.09780 [cs].

[3] 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," p. 15.

[4] Z. Zhuang, R. Li, K. Jia, Q. Wang, Y. Li, and M. Tan, "Perception-Aware Multi-Sensor Fusion for 3D LiDAR Semantic Segmentation," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 16 260–16 270.

[5] G. Krispel, M. Opitz, G. Waltner, H. Possegger, and H. Bischof, "Fuseseg: Lidar point cloud segmentation fusing multi-modal data," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 1874–1883.

[6] Qilong 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) (IEEE Cat. No.04CH37566)*, vol. 3. Sendai, Japan: IEEE, 2004, pp. 2301–2306.

[7] 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*. IEEE, 2012, pp. 3936–3943.

[8] K. Kwak, D. F. Huber, H. Badino, and T. Kanade, "Extrinsic calibration of a single line scanning lidar and a camera," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 3283–3289.

[9] 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)*, 2020, pp. 8580–8586.

[10] 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.

[11] Y. Zhu, C. Li, and Y. Zhang, "Online Camera-LiDAR Calibration with Sensor Semantic Information," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. Paris, France: IEEE, May 2020, pp. 4970–4976.

[12] B. Nagy, L. Kovacs, and C. Benedek, "Online Targetless End-to-End Camera-LIDAR Self-calibration," in *2019 16th International Conference on Machine Vision Applications (MVA)*. Tokyo, Japan: IEEE, May 2019, pp. 1–6.

[13] 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, Oct. 2021.

[14] G. Wang, J. Qiu, Y. Guo, and H. Wang, "FusionNet: Coarse-to-Fine Extrinsic Calibration Network of LiDAR and Camera with Hierarchical Point-pixel Fusion," in *2022 International Conference on Robotics and Automation (ICRA)*. Philadelphia, PA, USA: IEEE, May 2022, pp. 8964–8970.

[15] X. Lv, B. Wang, Z. Dou, D. Ye, and S. Wang, "LCCNet: LiDAR and Camera Self-Calibration using Cost Volume Network," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Nashville, TN, USA: IEEE, Jun. 2021, pp. 2888–2895.

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

[17] G. Zhao, J. Hu, S. You, and C.-C. J. Kuo, "CalibDNN: Multimodal Sensor Calibration for Perception Using Deep Neural Networks," Mar. 2021, number: arXiv:2103.14793 arXiv:2103.14793 [cs].

[18] 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)*, Oct. 2018, pp. 1110–1117, arXiv:1803.08181 [cs].

[19] 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)*. Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 10 197–10 202.

[20] X. Lv, B. Wang, Z. Dou, D. Ye, and S. Wang, "CFNet: LiDAR-Camera Registration Using Calibration Flow Network," Apr. 2021, number: arXiv:2104.11907 arXiv:2104.11907 [cs].

[21] Z. Liu, H. Tang, S. Zhu, and S. Han, "SemAlign: Annotation-Free Camera-LiDAR Calibration with Semantic Alignment Loss," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Prague, Czech Republic: IEEE, Sep. 2021, pp. 8845–8851.

[22] S. Wu, A. Hadachi, D. Vivet, and Y. Prabhakar, "This Is the Way: Sensors Auto-Calibration Approach Based on Deep Learning for Self-Driving Cars," *IEEE Sensors Journal*, vol. 21, no. 24, pp. 27 779–27 788, Dec. 2021.

[23] C. Guindel, J. Beltrán, D. Martín, and F. García, "Automatic extrinsic calibration for lidar-stereo vehicle sensor setups," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–6.

[24] W. Chen, Z. Fu, D. Yang, and J. Deng, "Single-image depth perception in the wild," *Advances in neural information processing systems*, vol. 29, 2016.

[25] J.-H. Lee and C.-S. Kim, "Monocular depth estimation using relative depth maps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9729–9738.

[26] J. Li, R. Klein, and A. Yao, "A two-streamed network for estimating fine-scaled depth maps from single rgb images," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3372–3380.

[27] D. Xu, W. Wang, H. Tang, H. Liu, N. Sebe, and E. Ricci, "Structured attention guided convolutional neural fields for monocular depth estimation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3917–3925.

[28] N. dos Santos Rosa, V. Guizilini, and V. Grassi, "Sparse-to-continuous: Enhancing monocular depth estimation using occupancy maps," in *2019 19th International Conference on Advanced Robotics (ICAR)*. IEEE, 2019, pp. 793–800.

[29] L. He, C. Chen, T. Zhang, H. Zhu, and S. Wan, "Wearable depth camera: Monocular depth estimation via sparse optimization under weak supervision," *IEEE Access*, vol. 6, pp. 41 337–41 345, 2018.

[30] Y. Kuznietsov, J. Stuckler, and B. Leibe, "Semi-supervised deep learning for monocular depth map prediction," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 6647–6655.

[31] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5410–5418.

[32] R. Garg, V. K. Bg, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *European conference on computer vision*. Springer, 2016, pp. 740–756.

[33] A. Johnston and G. Carneiro, "Self-supervised monocular trained depth estimation using self-attention and discrete disparity volume," in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2020, pp. 4756–4765.

[34] Z. Zhou, X. Fan, P. Shi, and Y. Xin, "R-msfm: Recurrent multi-scale feature modulation for monocular depth estimating," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 777–12 786.

[35] 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.

[36] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] S. Chen, J. Liu, X. Liang, S. Zhang, J. Hyyppä, and R. Chen, "A novel calibration method between a camera and a 3d lidar with infrared images," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4963–4969.

[38] 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.

[39] J. Ku, A. Harakeh, and S. L. Waslander, "In defense of classical image processing: Fast depth completion on the cpu," in *2018 15th*

Conference on Computer and Robot Vision (CRV). IEEE, 2018, pp. 16–22.

[40] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[41] 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.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[43] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8934–8943.