

Article

DPCalib: Dual-Perspective View Network for LiDAR-Camera Joint Calibration

Jinghao Cao, Xiong Yang, Sheng Liu, Tiejian Tang , Yang Li  and Sidan Du *

School of Electronic Science and Engineering, Nanjing University, Nanjing 210023, China; 602022230006@smail.nju.edu.cn (J.C.); yx@smail.nju.edu.cn (X.Y.); anderliu@smail.nju.edu.cn (S.L.); ttj@smail.nju.edu.cn (T.T.); yogo@nju.edu.cn (Y.L.)

* Correspondence: coff128@nju.edu.cn

Abstract: The precise calibration of a LiDAR-camera system is a crucial prerequisite for multimodal 3D information fusion in perception systems. The accuracy and robustness of existing traditional offline calibration methods are inferior to methods based on deep learning. Meanwhile, most parameter regression-based online calibration methods directly project LiDAR data onto a specific plane, leading to information loss and perceptual limitations. A novel network, DPCalib, a dual perspective view network that mitigates the aforementioned issue, is proposed in this paper. This paper proposes a novel neural network architecture to achieve the fusion and reuse of input information. We design a feature encoder that effectively extracts features from two orthogonal views using attention mechanisms. Furthermore, we propose an effective decoder that aggregates features from two views, thereby obtaining accurate extrinsic parameter estimation outputs. The experimental results demonstrate that our approach outperforms existing SOTA methods, and the ablation experiments validate the rationality and effectiveness of our work.

Keywords: LiDAR-camera calibration; multimodal; sensor fusion; deep learning



Citation: Cao, J.; Yang, X.; Liu, S.; Tang, T.; Li, Y.; Du, S. DPCalib: Dual-Perspective View Network for LiDAR-Camera Joint Calibration. *Electronics* **2024**, *13*, 1914. <https://doi.org/10.3390/electronics13101914>

Academic Editor: Beiwen Li

Received: 1 April 2024

Revised: 28 April 2024

Accepted: 12 May 2024

Published: 13 May 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

LiDAR and cameras constitute the principal sensor modalities employed within autonomous driving perception systems. LiDAR sensors are instrumental in capturing highly precise spatial data in the form of point clouds, while cameras serve as repositories of rich visual information. Combining LiDAR and cameras ensures accurate and stable environment perception [1]. During the multimodal data fusion process, the calibration matrix assumes a critical role. Precise calibration outcomes play a pivotal part in enabling the spatial alignment of point clouds and images at both data and feature levels, ultimately leading to the attainment of accurate and resilient fusion perception results.

The goal of calibration tasks is to determine the transformation matrix between the coordinate systems of a LiDAR and a camera [2]. For example, as depicted in the top left corner of Figure 1, an autonomous driving vehicle with a visual system comprising a camera and LiDAR requires accurate calibration algorithms for the subsequent fusion and application of data from these two modalities. Accurate transformation relations between the camera coordinate system and the LiDAR coordinate system are necessary to align the data from the LiDAR and camera. However, manually calibrated initial parameters often contain errors, leading to inaccurate projection results. Therefore, we propose a deep learning-based calibration model named “DPCalib”, which is named for dual-perspective view network for LiDAR-camera joint calibration. The role of DPCalib is to utilize initially inaccurate extrinsic and point cloud-image data pairs, and through neural network computation, obtain calibrated extrinsic parameters at the pixel precision level. In other words, it achieves an extrinsic accuracy that matches the data from both modalities, essentially achieving pixel-level accuracy on a point-cloud-to-pixel basis.

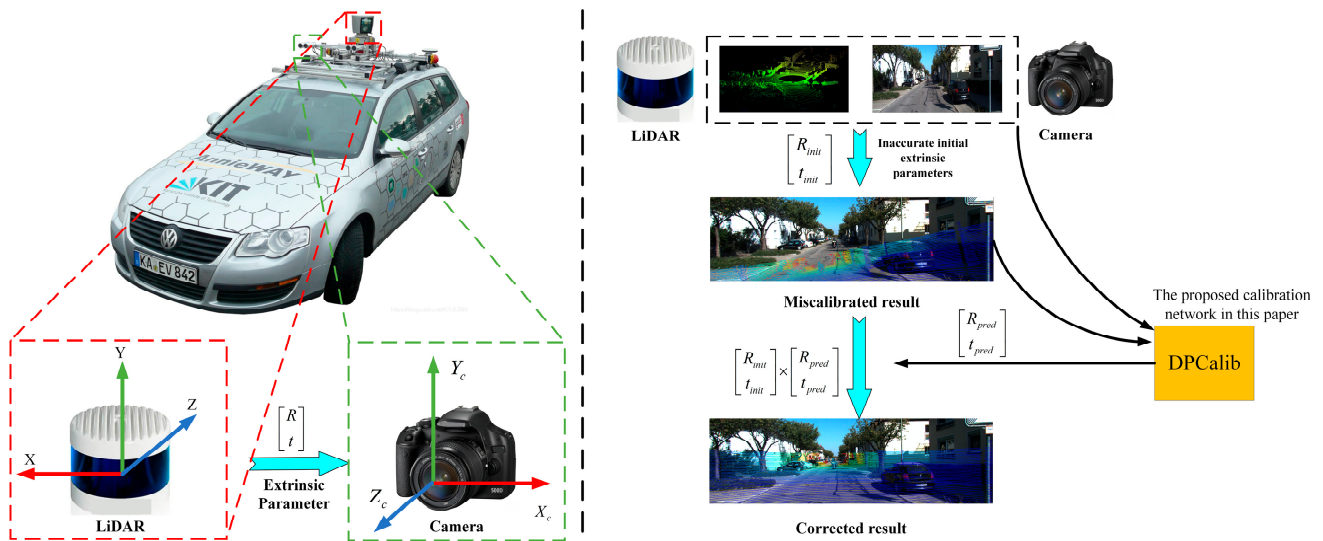


Figure 1. The task addressed by the LiDAR-camera joint calibration and the role of our proposed DPCalib in it. The joint calibration task primarily involves determining the rotation and translation relationship between the camera coordinate system and the LiDAR coordinate system in a multimodal fusion system to facilitate precise data fusion. Our proposed DPCalib takes a pair of a point cloud and image as input and calculates precise extrinsic parameters at the pixel level based on the initially inaccurate extrinsic parameters.

Traditional LiDAR-camera extrinsic calibration methods are divided into target-based and targetless calibration methods. Target-based calibration [3] methods rely on manually set specific targets, such as checkerboards [4], concentric circles [5], etc. Nontargeted [6] methods utilize common constraints between the two sensors, such as edges [7], normal planes [8], etc. However, experimental results [3,6] have shown that the accuracy and robustness of traditional calibration methods are inferior to methods based on deep learning [6]. It is worth noting that methods based on deep learning are fundamentally targetless, as they rely on feature extraction and fusion instead of specific targets. However, due to the significant differences in technical approach and characteristics between methods based on deep learning and traditional targetless methods, they are often categorized separately when discussing related work.

In recent years, thanks to the rapid advancement of computer vision and deep learning algorithms, end-to-end methods based on deep learning have demonstrated significant capabilities [9–11]. They can achieve online calibration in scenarios based on the initial extrinsic parameter input. Most deep learning-based methods typically adhere to a fundamental architecture as described below for network design. They take images and incorrectly calibrated depth maps as inputs. The entire process is generally divided into three stages: feature extraction, feature matching, and parameter regression. However, current methods tend to project the point cloud onto a limited viewpoint, resulting in the loss of geometric information in the point cloud. It can also impose limitations on the perception of certain angular and translational parameters, leading to significant errors in specific angles and, consequently, yielding suboptimal overall calibration results [12].

As shown in Figure 2, the point cloud from the LiDAR is transformed into the coordinate system of the camera using extrinsic and then projected onto the image plane using the intrinsic of the camera. When there are extrinsic errors between the camera and the LiDAR, different errors result in different pixel offsets. The pixel offsets caused by rotation errors, in particular, are far more pronounced than those caused by translation errors, and the patterns of point cloud displacement induced by different rotational errors are significantly distinct. According to the previous experimental reports [6,9–11], when a single viewpoint image is used as input, the neural network yields significant errors in the extrinsic pitch angle.

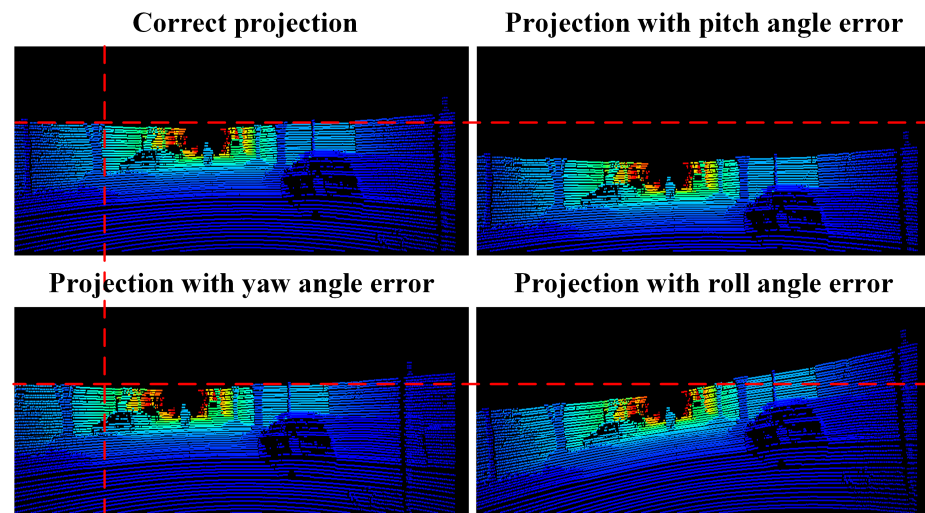


Figure 2. Projection of LiDAR point cloud onto the image plane based on extrinsic. The top-left corner depicts the correct projection, while the remaining images show the projection with errors in pitch, yaw, and roll parameters, respectively. It can be observed that different angular errors exhibit distinct manifestations in the image.

Therefore, we propose a question: Can we use a pair of mutually orthogonal maps as input to the neural network? In this way, when there are extrinsic errors between the LiDAR and the camera, the neural network can obtain more information instead of being restricted to a specific viewpoint. Based on the above reasoning, this paper introduces the concept of “dual perspective”.

Through the depth estimation method, we obtain the positions of pixels in the image, thus obtaining corresponding point cloud coordinates. Then, as shown in Figure 2, we can set up a plane orthogonal to the camera’s image plane, and project the corresponding points onto this orthogonal view to obtain a side view orthogonal to the main view. Similarly, the point cloud can also be projected onto these two planes simultaneously. Using these two viewpoints of the image as input and feeding them into the neural network provides the network with more information.

Figure 3 shows the projection relationship between the object and the camera, as well as the concept of “dual-perspective input”. The angle at which the real world is projected onto the camera coordinate system as the “principal view” is defined. The view orthogonal to the principal view is referred to as the “compensation view”. In this paper, the projections of the LiDAR point cloud and the depth map from these two perspectives will both serve as inputs to the neural network. Specific technical details and results are provided in Sections 3–5, respectively, where detailed explanations are given. The output of our “projection” operation is in pixel coordinate space. If the input is a point cloud, it is directly projected to the pixel coordinate space through specified extrinsic and intrinsic parameters. If the input is pixels (depth map), the pixel space input is first projected to 3D space through the current intrinsic and extrinsic pixel space. Then, the point cloud in 3D space is projected to the pixel space of the specified view through the intrinsic and extrinsic specified view.

Furthermore, the design of feature matching that directly leverages the semantic information from images and the geometric information from point clouds may encounter difficulties in feature extraction and convergence due to disparities stemming from multi-modal data inputs. Some previous works [3,6,13] preprocess images with depth estimation, then use the depth map and the projection of the point cloud as inputs to the neural network. While this approach can to some extent avoid the difficulty of data fusion and neural network [3,6,13] learning caused by different modalities, the results of depth estimation are dense at the pixel level, whereas the projection of the point cloud onto the image is sparse.

Therefore, further data fusion techniques are still required to eliminate the confusion caused by features of different modalities during neural network learning.

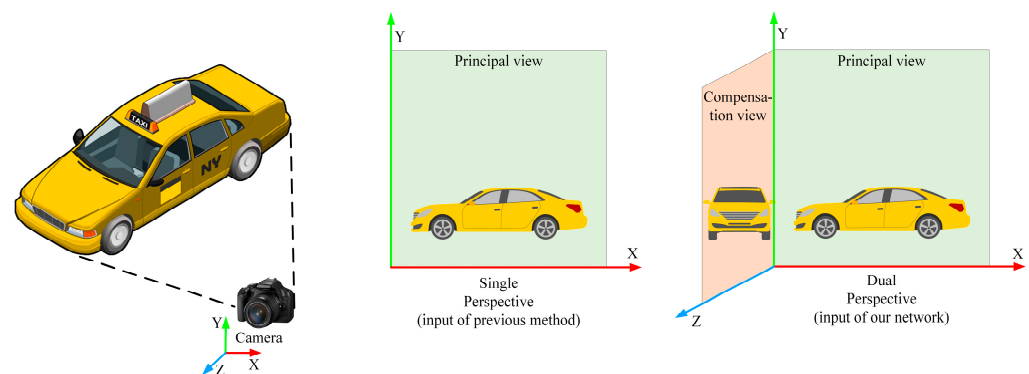


Figure 3. The projection relationship between the object and the camera, as well as the concept of “dual-perspective input”. Define the angle at which the real-world is projected onto the camera coordinate system as the “principal view”. The view orthogonal to the principal view is referred to as the “compensation view”. In this paper, the projections of the point cloud and the depth map from these two perspectives will both serve as inputs to the neural network.

Transformer [14,15] is an excellent neural network architecture that has shown outstanding performance in many visual tasks. However, it has a large number of learnable parameters and may not be suitable for tasks requiring high real-time performance. In the task of online calibration of the LiDAR-camera, due to the small number of parameters to be output and the need for a certain degree of real-time performance, directly using the Transformer structure may incur relatively high memory cost or excessively long inference times [16].

Therefore, we drew inspiration from the Transformer’s design principles and devised a neural network structure called Encoder-Decoder. In the encoder part, we incorporated the attention mechanism of the Transformer. By employing cross-attention and self-attention structures, we fused features from LiDAR and depth maps captured from different perspectives. In the decoder stage, to reduce the number of learnable parameters and mitigate the risk of overfitting, we utilized Convolutional Gated Recurrent Units (ConvGRU) [17] from recurrent neural networks to aggregate and decode the fused features. Since we believe that ConvGRU is the most efficient decoder based on the results of ablation study, that ConvGRU achieved the highest accuracy under approximately the same number of learnable parameters. The code will be open-sourced at <https://github.com/dogooooo/DPCalib> (accessed on 1 April 2024).

In summary, our contributions are as follows:

1. We propose DPCalib, a neural network architecture based on Encoder-Decoder with dual-perspective inputs. By leveraging a predefined compensatory view, we address the perceptual deficiencies caused by projecting LiDAR point clouds onto a specific viewpoint in the LiDAR-camera calibration task.
2. We design an Encoder based on attention mechanisms, utilizing self-attention and cross-attention mechanisms to effectively fuse features from dense depth maps and sparse point cloud projection images.
3. We propose an effective Decoder based on Convolutional Gated Recurrent Units (ConvGRU) to aggregate features between different viewpoints without requiring a large number of learnable parameters.
4. Our model demonstrates outstanding performance across various scenarios and error settings, surpassing existing methods of the same category. Furthermore, ablation experiments have confirmed the effectiveness of our innovative design.

2. Related Work

The joint calibration methods for LiDAR and camera can be categorized into traditional (a) offline calibration methods and (b) online calibration methods based on deep learning.

2.1. Offline Calibration Methods

Offline calibration methods mainly rely on manually designed features. These methods can be categorized into target-based [5,18–21] and targetless [22–24] methods. Target-based methods rely on specially designed calibration objects, where multiple sensors simultaneously focus on the same special calibration object. Based on the features of the special calibration object and manually designed constraints, the extrinsic parameters are eventually computed. Target-less methods, on the other hand, do not rely on specific markers. Instead, they utilize features within the scenes captured by two devices, such as edges of buildings or vehicles in a normal plane. These features serve as constraints to formulate equations and solve for the extrinsic parameters without the need for special markers.

2.1.1. Target-Based Methods

The core idea of these methods is to have two devices simultaneously observe particular objects such as checkboards [5], V-shaped boards [18,19], spheres [20,21], hollow circles [4], and other targets [25].

For example, Yoonsu Park [5] proposed a calibration method using the estimation of 3D corresponding points emerging from the laser points scanned across the polygonal planar board, where sides lay adjacent. Kiho Kwak et al. [18] introduced a method by a V-shaped board, they introduced two methods to enhance calibration accuracy. Initially, they assign varied weights to the distance between a point and a line feature based on the precision of feature correspondence. Secondly, they incorporate a penalization function to mitigate the impact of outliers within the calibration datasets. Zoltan Pusztai et al. [23] primarily centered on identifying box planes within LiDAR point clouds, their methodology enables the calibration of various LiDAR equipment. Additionally, it allows for LiDAR-camera calibration with minimal manual intervention.

2.1.2. Targetless Methods

Tamas et al. [22] introduced a nonlinear explicit calibration approach that omits correspondence, treating the calibration problem as a 2D–3D registration within a shared LiDAR-camera domain. This method utilizes minimal information, such as depth data and area shapes, to formulate a nonlinear registration system, directly yielding the calibration parameters for the LiDAR-camera setup. Moreover, ref. [23] proposed an innovative technique for determining the extrinsic calibration between 3D LiDAR and Omnidirectional Cameras. This method, devoid of 2D–3D corresponding points or intricate similarity measurements, relies on a series of corresponding regions and computes the pose by resolving a small nonlinear system of equations. Additionally, Pandey et al. [24] employed the effective correlation coefficient between surface reflectivity from LiDAR and intensity captured by a camera as a calibration function for extrinsic parameters, while keeping other parameters constant.

However, these calibration methods are relatively time-consuming. Matching corner points often require manual selection or the manual design of feature-matching rules, which results in a relatively small number of effectively matched points. Furthermore, they are susceptible to environmental factors, making them less robust.

2.2. Online Calibration Methods

Calibration methods based on deep learning replace the manual selection of corner points or the design of feature-matching rules with the use of learnable parameters, enabling online calibration without the need for calibration targets. The methods based on deep learning can be further divided into two categories: methods that directly regress parameters [3,10,26] and methods based on pixel flow estimation [17,27,28].

2.2.1. Pixel-Flow-Estimated-Based Methods

Methods based on pixel flow estimation use LiDAR projection images and images as input. They estimate the pixel flow between the projection images and the images using methods similar to optical flow estimation. Based on this pixel flow, corresponding points between the two inputs are found, and then the extrinsic parameters are solved based on the relationship between these corresponding points.

CFNet [29] draws inspiration from optical flow estimation, using the “flow” between projected points and RGB images for supervision and employs the RANSAC [30] algorithm to estimate LiDAR-camera extrinsic. DXQ-Net [31] extends this by adding an iterative structure based on GRU [17] network, continuously refining the estimation results of the network. SOIC [27] proposes a cost function for matching constraints based on image semantic elements and LiDAR point clouds. SemAlign [28] simultaneously performs semantic segmentation on both images and point clouds, transforming the calibration problem into a pattern-matching problem.

However, this category of methods involves optical flow estimation, where the larger the pixel search range, the greater the computational cost. Within limited computational resources, these methods can only handle relatively small initial errors. Published works [29,31] have also confirmed this viewpoint. Furthermore, since translation errors during the projection process result in only small pixel offsets, such methods are limited in their ability to estimate displacement accurately.

2.2.2. Regression-Based Methods

Methods based on parameter regression refer to techniques where input images and projected point clouds are processed through neural networks to directly output extrinsic parameters. Initially, these methods primarily used RGB images as input [10], but later iterations have incorporated improvements such as depth estimation [3] or semantic segmentation results [26] into the model.

RegNet [10] is the first model to propose the use of Convolutional Neural Networks (CNN) for calibration, and subsequent researchers have built upon this work to improve the model. CalibNet [11] introduces a geometrically supervised deep network capable of real-time automatic extrinsic parameter estimation. It conducts end-to-end training by maximizing the geometric and photometric consistency between input images and point clouds. RGGNet [9] utilizes Riemannian geometry and deep generative models to construct a tolerance-aware loss function for supervised training. CalibDNN [32] introduces a structure based on iterative optimization, achieving a multilevel correction calibration model. LCCNet [29] suggests building a cost volume using the dot product of image and point cloud projection features and trains the network using distance errors between back-projected point clouds and ground truth point clouds. There are also some works that preprocess images or point clouds before calibration. For example NetCalib [13] is the first work to use stereo depth estimation methods to process images and then use the depth map and point cloud projection as inputs. Subsequently, CalibDepth [3] also employs a depth estimation module for preprocessing the image component and designs an iterative structure for iteratively optimizing the calibration parameters.

However, this category of methods faces perceptual defects due to the projection of point clouds in specific viewpoints, leading to deformation and loss of information. Therefore, experimental evidence demonstrates that directly using images and projections from a specific viewpoint as input can result in unsatisfactory estimation in certain directions.

In summary, both of these technical approaches have areas that are worth improving. Our DPCalib is a parameter regression-based method, but we alleviate the shortcomings of this approach by utilizing a predefined compensation perspective. Additionally, in experiments with small inaccuracies, our method still maintains an advantage over pixel flow methods.

3. Approach

The overall workflow of DPCalib is illustrated in Figure 4. DPCalib takes the depth map from an image and a frame of LiDAR point cloud as inputs. The subsequent calibration process involves feature encoding, feature aggregation, and parameter regression, ultimately yielding high-precision calibration results.

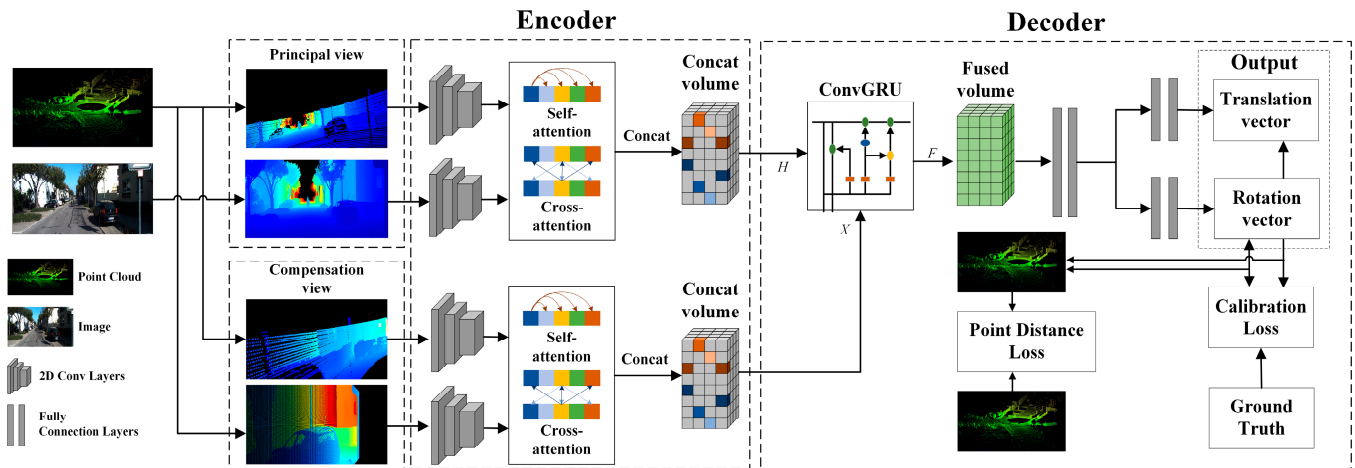


Figure 4. The network architecture of DPCalib. DPCalib is an Encoder-Decoder structured network. The principal view’s point cloud and depth image are projected to obtain compensated perspective maps. These images from two perspectives are then used as input to DPCalib. The encoder primarily extracts features from different perspectives. The features from different perspectives are aggregated and decoded through a combination of ConvGRU and fully connected layers in the decoder, resulting in the final output.

As shown in Figure 4, DPCalib utilizes an Encoder-Decoder structured network. Initially, the principal view’s point cloud and depth image undergo projection to generate compensated perspective maps. These images, originating from two perspectives, serve as input for DPCalib. The encoder is chiefly responsible for extracting features from various perspectives. Subsequently, the features extracted from diverse perspectives are amalgamated and decoded using a blend of ConvGRU and fully connected layers within the decoder, culminating in the final output.

3.1. Problem Formulation

For the sake of computational convenience and consistency in representation, the coordinate systems for the LiDAR, camera, and pixel in this paper are set to be consistent with the KITTI [33,34] dataset.

Using multimodal data as input, similar to previous works [3,29], the task of this model is to correct an erroneous initial extrinsic parameter T_{init} , obtaining the calibration matrix ΔT , and based on this, compute the correct extrinsic matrix $\hat{T}$. Therefore, it is necessary to project the multimodal data. For the point cloud $P_i(X_i, Y_i, Z_i) \in \mathbb{R}^3$, all of the points are transformed into 2D pixel points using T_{init} and the camera intrinsic parameters K_c . This process is represented as Equations (1) and (2):

$$Z_{init}^i \cdot \hat{p}_i = Z_{init}^i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = K_c T_{init} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \quad (1)$$

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

R_{init} and t_{init} are the rotation matrix and translation vector in T_{init} , respectively. A depth projection image of the LiDAR point cloud on the camera plane I_{LiDAR} is obtained in this way, where the value of each pixel point (u, v) on represents the depth value Z_{init} of that point in the camera coordinate system.

Furthermore, as shown in Figure 5 we define the x and y axes of the camera coordinate system as the horizontal and vertical directions of the image, respectively, and the z -axis as the coordinate axis perpendicular to the image plane. Therefore, we set up a virtual plane $I_{comp}(u', v')$ perpendicular to the camera's image plane (xy plane) and coinciding with the camera's image plane, the viewpoint of I_{comp} is named as "compensation view". The pixel points on the compensation view plane correspond to the LiDAR point cloud as shown in Equations (4) and (5).

$$K' = \begin{bmatrix} f'_u & 0 & c'_u \\ 0 & f'_v & c'_v \\ 0 & 0 & 1 \end{bmatrix} \quad (3)$$

$$u'_i = \frac{f'_u \times (-Z_i)}{X_i}, \quad v'_i = \frac{f'_v \times (-Z_i)}{Y_i} \quad (4)$$

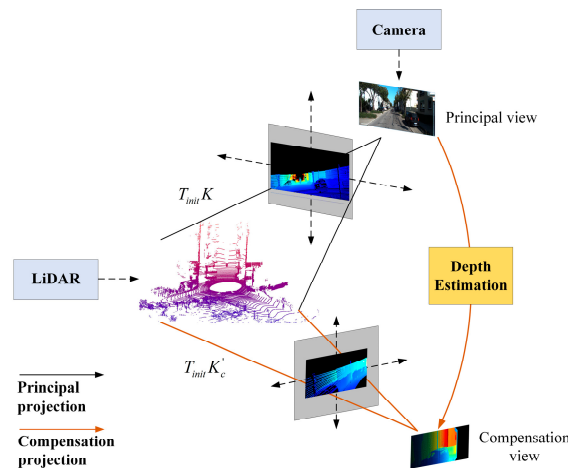


Figure 5. The LiDAR point cloud is initially transformed into the camera coordinate system using the initial extrinsic parameters. Subsequently, it is projected onto different perspectives using the camera's intrinsic parameters and the intrinsic parameters of the virtual plane. Similarly, the results of depth estimation are used to obtain the coordinates of points in space using the camera's intrinsic parameters, and then projected onto the compensation perspective using the intrinsic parameters of the virtual plane.

As shown in Equation (3), we define a virtual intrinsic matrix K' to project the point cloud onto a view that is orthogonal to the primary view. f'_u, f'_v, c'_u, c'_v are, respectively, the preset virtual intrinsic parameters representing the focal length and the pixel center. By adjusting the appropriate intrinsic parameters K' , we project the LiDAR point cloud, which has already been transformed into the camera coordinate system, onto the compensation plane. The setting of the virtual intrinsic parameters K' affects the receptive field and receptive position of the image. It should not be too large, as this would result in a significant portion of the image not being hit by the point cloud, leading to information wastage. Additionally, it should not be too small, as this would make the algorithm overly sensitive to errors, resulting in convergence difficulties. Furthermore, since depth estimation algorithms can also introduce errors, a too-small receptive field would decrease the algorithm's robustness.

Similarly, we first perform depth estimation on the image I_{image} by a stereo matching neural network Θ_d , and then project the depth map of the image to the same compensation perspective. As shown in Equations (5)–(7)

$$D_{image}(u, v) = \Theta_d(I_{image}(u, v)) \quad (5)$$

$$Z_{image}(u, v) = \frac{B \times f_u}{D_{image}(u, v)} \quad (6)$$

$$X_{image} = \frac{(u - c_u) \times Z_{image}}{f_u}, Y_{image} = \frac{(v - c_v) \times Z_{image}}{f_v} \quad (7)$$

Equation (5) reflects the process of computing disparities of input image $I_{image}(u, v)$ by the stereo matching neural network Θ_d , while Equation (6) describes the process of converting disparities $D_{image}(u, v)$ into corresponding depths $Z_{image}(u, v)$. Here, B represents the baseline of the stereo images, and f_u is the focal length of the image. Equation (7) represents the process of obtaining the corresponding point cloud for each pixel based on the depth map $Z_{image}(u, v)$ and the camera's intrinsic parameters. Finally, we obtain $P_{image}(X_{image}, Y_{image}, Z_{image}) \in \mathbb{R}^3$, which can be further transformed into the compensation view using Equations (3) and (4).

3.2. Network Architecture

DPCalib is an encoder-decoder architecture network that takes the projection of LiDAR data and the depth map from the principal and compensation perspectives as input. We set the LiDAR projection map and image depth map from the same perspective as from a group. Initially, ResNet18 [35] is used to extract features from each group, resulting in features from the principal perspective and features from the compensation perspective. Subsequently, attention mechanisms [14] are employed to fuse the features from each group separately, and the fused features are concatenated. Then, a GRU [17] operator is utilized to merge the features from both perspectives, yielding the merged features. Finally, a fully connected layer is employed to obtain the final extrinsic parameter output from the merged features. This section provides a detailed introduction to the architecture of DPCalib.

3.2.1. Encoder Architecture

As shown in Figure 4, ResNet18 [35] is used as the feature extraction network. Inspired by LCCNet [29], all the ReLU [36] activation functions in the ResNet18 are replaced with Leaky-ReLU [37]. We set the LiDAR projection map and image depth map from the same perspective as from a group. The depth map and LiDAR point cloud projection map on the principal perspective are denoted as $D_{image}(u, v) \in \mathbb{R}^2$ and $D_{LiDAR}(u, v) \in \mathbb{R}^2$, respectively. Similarly, the depth map and point cloud projection map from the compensation perspective are denoted as $D'_{image}(u, v) \in \mathbb{R}^2$ and $D'_{LiDAR}(u', v') \in \mathbb{R}^2$. Since the encoding process for the two sets of features from both perspectives is identical, we primarily focus on explaining the encoding process for the main perspective.

As shown in Figure 6, D_{image} and D_{LiDAR} are, respectively, input into two symmetric ResNet18 networks to obtain preliminary extracted features, denoted as $F_{image} \in \mathbb{R}^{(h/32, w/32, c1)}$ and $F_{LiDAR} \in \mathbb{R}^{(h/32, w/32, c1)}$. Their resolutions are 1/32 of the initial input map, and c represents the number of channels in the features (c is set to 512 in this paper). However, the data distribution of the dense depth map output by the depth estimation network and the sparse depth map obtained from the point cloud projection is notably different. The most significant difference is that the sparse depth map has many invalid values near the effective depth, whereas the depth values of a dense depth map are relatively smoother. Therefore, inspired by [38], we use an attention mechanism-based [14] method to fuse the features.

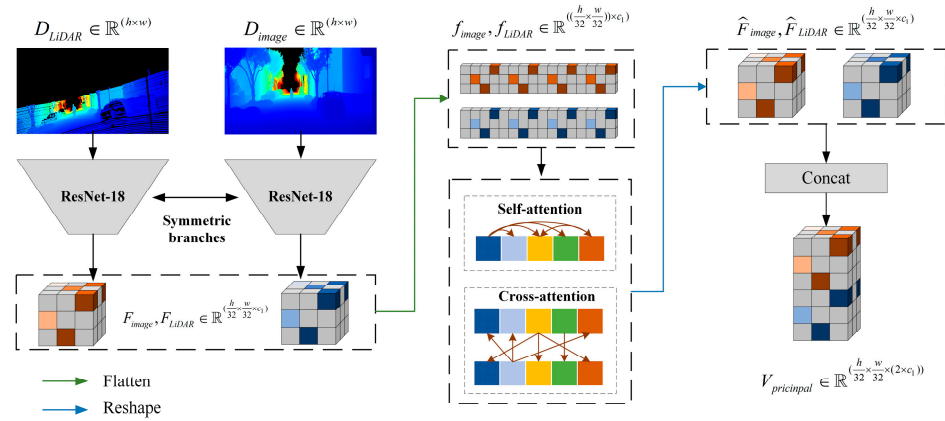


Figure 6. The workflow diagram of the feature matching structure in Encoder. ResNet18 is employed to generate feature maps. These feature maps then undergo the self-attention and cross-attention module to acquire fused features, which are subsequently aggregated into a cost volume by concatenation fuse method.

First, the flatten operation is applied to F_{image} and F_{LiDAR} along their resolution channels ($h \times w$), resulting in $f_{image} \in \mathbb{R}^{((h/32 \times w/32) \times c_1)}$ and $f_{LiDAR} \in \mathbb{R}^{((h/32 \times w/32) \times c_1)}$. Then, they are sequentially subjected to self-attention and cross-attention operations. It is worth noting that in this paper, self-attention operations are performed on the features first. This is based on [14], where it is considered more reasonable to extract their own patterns through self-attention operations before the fusion of features from two perspectives. We adopt a linear multihead attention mechanism structure, as shown in Figure 7.

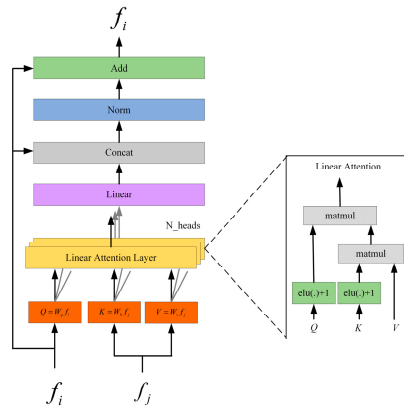


Figure 7. Attention Layer in DPCalib. In the self-attention mechanism, f_{image} or f_{LiDAR} simultaneously serve as inputs f_i and f_j into the structure shown in the diagram. In the cross-attention mechanism, f_{image} and f_{LiDAR} are separately used as inputs f_i and f_j .

Compared to the dot-product-based multihead attention mechanism, the linear multihead attention reduces the algorithm's complexity from $O(N^2)$ to $O(N)$, with no significant decrease in accuracy [14].

$$Attention(Q, K, V) = softmax(QK^T)V \quad (8)$$

In this structure, the input features f_i and f_j are divided into N groups along the feature channels and labeled as Q, K, V , respectively. Then, the features are merged through fully connected layers and matrix multiplication, as shown in Equation (8), to obtain the fused output. For DPCalib, in the self-attention mechanism f_{image} or f_{LiDAR} simultaneously serve as inputs f_i and f_j , in the cross-attention mechanism, f_{image} and f_{LiDAR} are separately used as inputs f_i and f_j .

After the attention mechanism fusion, the features are restored to their original shape through a reshape operation, denoted as $\hat{F}_{image} \in \mathbb{R}^{(h/32 \times w/32 \times c_1)}$ and $\hat{F}_{LiDAR} \in \mathbb{R}^{(h/32 \times w/32 \times c_1)}$, respectively. Then, the two features are concatenated through a concatenation operation to obtain the fused feature volume from the principal perspective, denoted as $V_{principal} \in \mathbb{R}^{(h/32 \times w/32 \times (2 \times c_1))}$. Similarly, the compensation perspective can also obtain $V_{compensation} \in \mathbb{R}^{(h/32 \times w/32 \times (2 \times c_1))}$ through similar operations.

3.2.2. Decoder Architecture

The classic Transformer architecture employs several stacked attention layers as a decoder [14]. However, this relies on a large number of learnable parameters. In the task of LiDAR-camera calibration, since only one 6-degree-of-freedom extrinsic parameter matrix is ultimately regressed, an excessive number of learnable parameters can lead to overfitting of the network and waste computational and time costs. Therefore, we adopt a GRU (Gated Recurrent Unit) [17] module as the decoder of the network.

GRU [17] is a module with selective storage, forgetting, and updating capabilities. It is originally designed to capture long-term dependencies in sequential inputs. However, in recent years, some works, such as RAFT [39], CREStereo [38], RMVSNet [40], use GRU [17] for updating or iterating over existing results. Inspired by the aforementioned works, we utilize ConvGRU [39] for feature correction, as shown in Equations (9)–(13).

$$V_{principal} \in \mathbb{R}^{(h/32 \times w/32 \times (2 \times c_1))} = H, V_{compensation} \in \mathbb{R}^{(h/32 \times w/32 \times (2 \times c_1))} = X \quad (9)$$

$$R = \sigma(XW_{xr} + HW_{hr} + b_r) \quad (10)$$

$$Z = \sigma(XW_{xz} + HW_{hz} + b_z) \quad (11)$$

$$\tilde{F} = \tanh(XW_{xh} + (R \odot H)W_{hh} + b_n) \quad (12)$$

$$F = Z \odot H + (1 - Z) \odot \tilde{F} \quad (13)$$

In the encoder stage, the feature $V_{principal}$ from the principal perspective are regarded as the hidden state H of the GRU, while the feature $V_{compensation}$ from the compensation perspective are regarded as the input X . Where R and Z regulate the forgetting and updating of the input H . F and $\tilde{F}$, respectively, denote the final fused output feature from the dual-perspective and the candidate output features. In addition, similar to the original GRU [39] network, σ represents the sigmoid activation function, and $\odot$ represents element-wise multiplication.

We construct two output head using fully connected layers to parse the fused volume $F \in \mathbb{R}^{(h/32 \times w/32 \times (2 \times c_1))}$. After passing through two layers of fully connected layers, it enters two separate decoding heads, each of which regresses the R_{pred} in quaternion form and the t_{pred} in the Cartesian coordinate system.

3.2.3. Process of Inference

Traditional “offline calibration” methods require collecting multiple sets of data in specified scenarios, followed by algorithm design to compute extrinsic, ensuring that the projections based on extrinsic correspond to most of the collected data. In contrast, methods based on deep learning only require one set of images/point cloud inputs to achieve real-time end-to-end extrinsic estimation. Therefore, we refer to these methods as “online methods”.

The inference process of our DPCalib can be described as follows: Given an input image I_{image} and misprojected point cloud map D_{LiDAR} , the input image I_{image} passes through a depth estimation network to obtain a depth map D_{image} . Then, D_{LiDAR} and D_{image} are projected to obtain D'_{LiDAR} and D'_{image} . These four images serve as inputs to the network. They undergo feature extraction to obtain $V_{principal}$ and $V_{compensation}$. $V_{principal}$ and $V_{compensation}$ are

then processed by a GRU-based decoder to obtain fused feature F . F passes through several fully connected layers to obtain R_{pred} and t_{pred} .

3.3. Loss Function

We employ two forms of loss functions for supervised training of the network, namely regression loss L_T and point cloud distance error loss L_P , as shown in Equation (14).

$$L_{total} = \lambda_T L_T + \lambda_P L_P \quad (14)$$

As shown in Equation (14), λ_T and λ_P are the hyperparameters to the L_T and L_P . The L_T provides direct supervision based on ground truth.

It can be divided into three parts, corresponding to the quaternion angle distance D_a of the predicted quaternion angles $Q_{pred} = (s_{pred}, a_{pred}, b_{pred}, c_{pred})$ and the ground truth quaternion angles $Q_{gt} = (s_{gt}, a_{gt}, b_{gt}, c_{gt})$; the conversion of predicted angles Q_{pred} to euler angles $R_{pred} = (pitch_{pred}, roll_{pred}, yaw_{pred})$, which is then compared to the ground truth euler angles R_{gt} with SmoothL1 Loss; and the SmoothL1 Loss between the predicted translation t_{pred} and the ground truth translation t_{gt} . They are shown in Equations (15) and (16).

$$D_a = \cos^{-1} \left(\frac{s_{gt}s_{pred} + a_{pred}a_{gt} + b_{pred}b_{gt} + c_{pred}c_{gt}}{\left\| \sqrt{s_{pred}^2 + a_{pred}^2 + b_{pred}^2 + c_{pred}^2} \right\| \left\| \sqrt{s_{gt}^2 + a_{gt}^2 + b_{gt}^2 + c_{gt}^2} \right\|} \right) \quad (15)$$

$$L_T = \lambda_r \left(D_a(Q_{gt}, Q_{pred}) + L1_{Smooth}(R_{gt}, R_{pred}) \right) + \lambda_t L1_{Smooth}(t_{gt}, t_{pred}) \quad (16)$$

where D_a is the quaternion angle distance between the Q_{pred} and Q_{gt} , λ_r , λ_t is the hyperparameters to balance the attention of the DPCalib between the rotational and translational extrinsic parameters.

In addition, we incorporate supervision based on point cloud error. The LiDAR point cloud input is denoted as $P = \{P_1, P_2, \dots, P_N\}$, $P_i \in \mathbb{R}^3$. The point cloud error loss L_P is presented in Equations (17) and (18). This type of loss allows for network supervision using the 3D information of the point cloud.

$$T_{pred} = \begin{bmatrix} R_{pred} & t_{pred} \\ 0 & 1 \end{bmatrix} \quad (17)$$

$$L_P = \frac{1}{N} \sum_{i=1}^N \left\| \left(T_{gt} T_{pred}^{-1} T_{init} P_i \right) - P_i \right\|_2 \quad (18)$$

where N represents the number of points in the point cloud, T_{init} is the initial transformation matrix between the camera and LiDAR, T_{gt} is a random perturbation added to T_{init} , and T_{pred} is the predicted value of T_{gt} by the DPCalib.

4. Comparing with Other State-of-the-Art

We evaluated the proposed DPCalib on the KITTI Raw [33] and KITTI Odometry [34] datasets, in this section, we provide a detailed overview of dataset preparation, implementation details, and an analysis of various experimental results.

4.1. Dataset Preparation

In order to make a fair comparison with prior research, our experimental setup aligns with previous works. As shown in Table 1, for Experiment 1 (Exp 1), we use the same experimental configuration as CalibNet [11], CalibDNN [32], and CalibDepth [3]. We train our model on the 2011_9_26 subset of the KITTI Raw dataset and test it on the 2011_09_30 subset. In Experiment 2 (Exp 2), we adopt the same experimental settings as NetCalib [13] and Calibdepth [3]. We use 0013, 0020, and 0079 from the 2011_09_26 subset of the KITTI Raw dataset as validation data and 0005 and 0070 as the test set, while the remaining data are used for training. Experiment 3 (Exp 3) is conducted for comparison with DXQ-Net [31]

and ablation experiments. The experimental setting involves using the 01–20 subset of the KITTI Odometry dataset as the training set and the 00 subset as the test set.

Table 1. Different experiments setups.

Index	Dataset	Disturb Range	Training Set	Validation Set	Test Set
Exp 1	KITTI Raw	10°, 0.25 m	9–26	/	9–30
Exp 2	KITTI Raw	10°, 0.2 m	9–26 without val set	13, 20, 79	5, 70
Exp 3	KITTI odom.	5°, 0.1 m	1–20	/	0

We add random rotation and translation errors within the specified error range based on T_{init} . The error settings followed a uniform distribution. Furthermore, during training, the errors for each batch were randomly generated in the code, while during validation and testing, the errors were pregenerated and saved. This approach ensures fairness in comparative and ablation experiments.

4.2. Implementation Details

In line with prior studies, we employ evaluation metrics for rotation R_{pred} and translation t_{pred} . Translation vectors are assessed using the average absolute error that can be denoted as E_r and E_t for the average absolute error across the translation and rotation components, respectively. For rotation matrices, we convert the output quaternions to Euler angles and compute the error.

We use CREStereo [38] to estimate the depth map of target images as the input of the image branch in DPCalib using the stereo input from the dataset. We trained the proposed network on a single RTX 3090, and the depth map from the principal viewpoint was resized to 512×256 as input to the network. The resolution of the compensated viewpoint is the same as the main viewpoint after resizing. In terms of the intrinsic K' of compensation view, we set the focal length $f'_u = 90$ and, the pixel center (c'_u, c'_v) is (512, 128). We make further elaborate on the rationale behind our design choices in the section discussing the section of ablation experiments.

It's worth noting that while the depth map obtained from stereo matching algorithms is generally similar to ground truth, there still exist significant errors, particularly at the edges. Through ablation experiments, we find that using these error-prone results directly as input during training does not facilitate loss function convergence.

Therefore, we first project a “sparse ground truth depth” using point clouds and T_{init} . This sparse ground truth depth temporarily replaces the depth estimation image for training the neural network, enabling the network to learn corner matching capabilities, this process is referred to as “guided training”. Subsequently, we fine tune the model using the “depth estimation image” as input.

The training settings for both the “guided training” and “fine-tuning” stages are identical. The batch size is set to 32, the total number of epochs is 40, the optimizer used is Adam, and the learning rate is 1×10^{-4} with a decay rate of 0.9. In terms of the weighting of the loss function, $\lambda_T = 0.95$, $\lambda_P = 0.05$, $\lambda_r = 0.7$, $\lambda_t = 0.3$.

4.3. Comparison to Existing Methods

According to the experimental conditions specified in Table 1, we conducted comparisons with different state-of-the-art (SOTA) works. The purpose of this approach is to maintain consistency with previous SOTA works in terms of experimental conditions. For example, CalibNet [11], CalibDNN [32], and CalidDepth [3] utilized the dataset and error range mentioned in Exp 1 for experimentation and comparison. Therefore, we also compared our method with these works under the same conditions. In the quantitative analysis, we directly referenced the error results reported by these works when they were published. The same approach was applied to Exp 2 and 3.

However, it is worth noting that most of these works did not release complete code or provide trained models and qualitative analysis results. We attempted to reproduce

their work but found it extremely difficult for most of the works, as we could not obtain the results mentioned in their papers. Among all the SOTA works, we only successfully reproduced NetCalib [13] and obtained results consistent with those mentioned in their paper. Therefore, in addition to publishing the results of NetCalib as publicly available in Exp 2, we also followed training strategy of original NetCalib to obtain results for Exp 1 and 3. These training results have been added to Tables 2 and 3. The bold formatting in all tables indicates that the corresponding metric achieved the best performance in horizontal or vertical comparisons. Furthermore, in this section, besides comparing Exp 2 with NetCalib, we directly compared the qualitative analysis of other experiments with ground truth.

Table 2. The comparison results between our method and other state-of-the-art (SOTA) approaches under the conditions of Exp 1.

Method	Rotation (°)			E_r	X	Translation (cm)		E_t
	Roll	Pitch	Yaw			Y	Z	
CalibNet [11]	0.180	0.900	0.150	0.410	12.10	3.49	7.87	7.82
CalibDNN [32]	0.150	0.990	0.200	0.447	5.50	3.20	9.60	6.10
CalibDepth [3]	0.180	0.682	0.181	0.348	6.66	1.12	6.48	4.75
NetCalib [13]	0.200	0.561	0.372	0.378	6.55	3.10	3.50	4.38
DPCalib (ours)	0.072	0.176	0.141	0.130	1.10	1.05	2.63	1.59

Table 3. The comparison results between our method and other state-of-the-art (SOTA) approaches under the conditions of Exp 3.

Method	Rotation (°)			E_r	X	Translation (cm)		E_t
	Roll	Pitch	Yaw			Y	Z	
DXQ-Net [31]	0.049	0.046	0.032	0.042	0.754	0.476	1.091	0.774
NetCalib [13]	0.083	0.189	0.103	0.125	1.618	0.917	1.337	1.291
DPCalib (ours)	0.030	0.134	0.051	0.072	0.482	0.460	0.958	0.633

The results of Exp 1 are shown in Table 2, where the maximum rotation error is set to $\pm 10^\circ$ and the maximum translation error is set to ± 0.25 m. This represents a relatively large error range. Consequently, previous algorithms tend to exhibit more noticeable errors in the pitch angle estimation. For example, CalibNet [11] achieved a pitch angle error of 0.900° , which is five times higher than its error in roll and yaw estimation. Moreover, the absolute value of the error is also relatively large. Similarly, CalibDNN [32] and CalibDepth [3] faced similar issues.

In addition to the aforementioned issues, we find that even when the model training converged, NetCalib [13] exhibited relatively larger errors in displacement compared to Calibdepth [3]. This is attributed to the output head of NetCalib. While most works [3,11,32] have two parallel output heads that separately output rotation matrices and translation vectors, NetCalib has only one output head that simultaneously outputs both values. This can result in the network being less sensitive to displacement errors. In contrast, our proposed DPCalib mitigates this problem by compensating for the viewpoint features. As a result, our DPCalib demonstrates superior performance in pitch angle estimation compared to other SOTA works. Overall, our DPCalib outperforms other SOTA works in both displacement and rotation error estimation. The qualitative analysis results of our DPCalib are depicted in Figure 8.

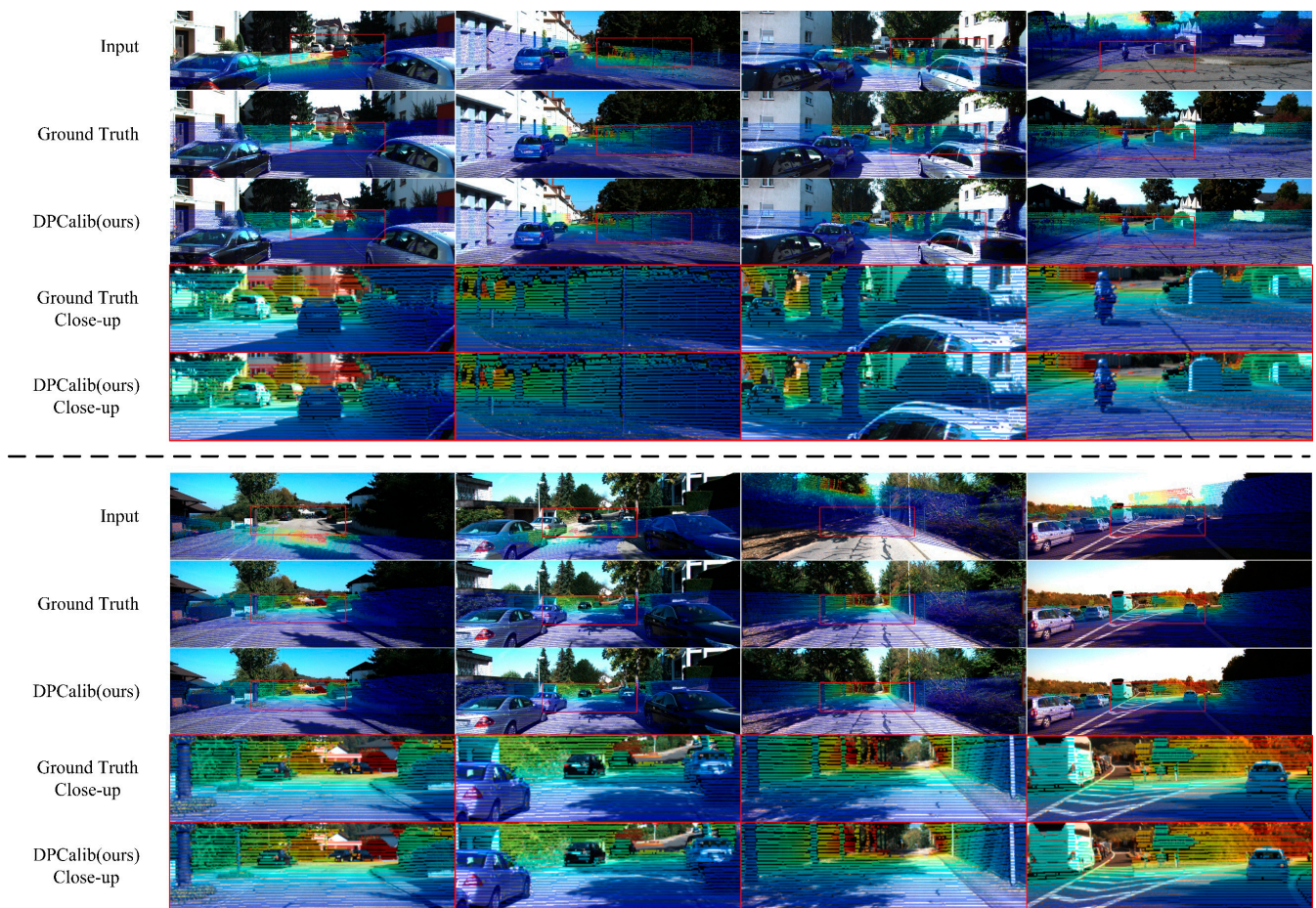


Figure 8. Presents the qualitative analysis results of DPCalib under the settings of Experiment 1. The red boxes indicate areas of that can more prominently reflect the differences between different methods in the local regions of the images.

As shown in Figure 8, it illustrates the precision of calibration with 8 images and LiDAR projections under different scenarios. The first row depicts the results obtained by projecting with initial extrinsic provided parameters that contain errors. It's evident that there is a misalignment between the geometric information of the point cloud and the semantic information of the image. The second row presents the situation when using correctly calibrated parameters, where the correspondence between the geometric information of the point cloud and the semantic information of the image is correct. The closer the algorithm's estimated results are to the ground truth, the better the algorithm's performance. The fourth and fifth rows, respectively, provide close-up views of the ground truth and the estimated values. We utilize clearly defined objects such as poles, pedestrians, and vehicles to further compare the differences between the algorithm's results and the ground truth. For images like Figure 8 that represent qualitative analysis results, our description follows a sequence from top to bottom and from left to right. For example, in the top row, the second image is referred to as "the 2nd input" or "the input 2", while the third image in the row below is referred to as "the 7th input" or "the input 7". Let's take the second and seventh inputs as an example. From both the overall image and the close-up view of the trees and pole-like objects, it can be observed that our algorithm has achieved results that are very close to the ground truth.

We reproduced the results of NetCalib [13] and conducted a qualitative analysis comparing it with our DPCalib, as shown in Figure 9. Overall, both our method and NetCalib [13] achieved the goal of updating the extrinsic parameters using initial extrinsic with errors. However, from Figure 9, especially in input 1–5 where pedestrians are present,

and in input 6–8 where pole-like objects are present, it can be observed that our method achieves more precise alignment of detailed information. This is reflected in the metrics, where the estimation errors of our DPCalib are slightly lower compared to NetCalib [13].

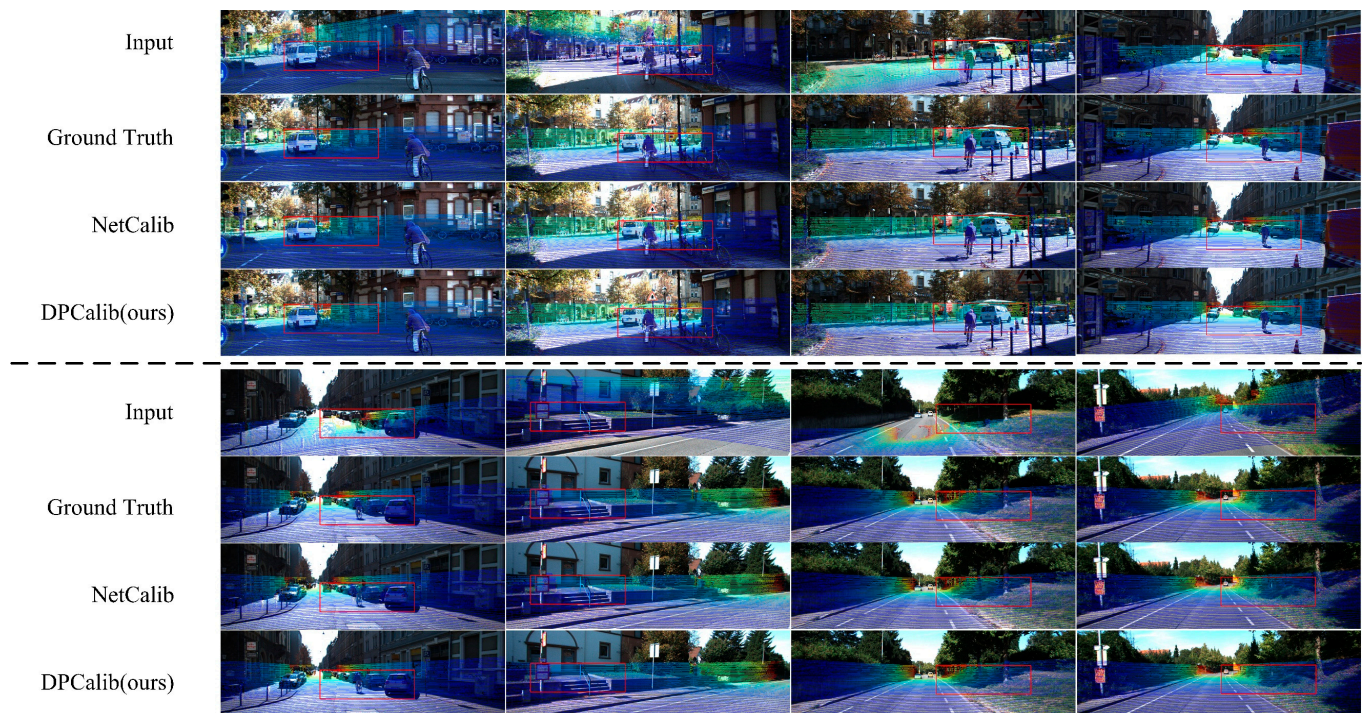


Figure 9. Presents the qualitative analysis results of DPCalib and other existing methods under the settings of Experiment 2. The red boxes indicate areas of that can more prominently reflect the differences between different methods in the local regions of the images.

The results of Exp 2 are shown in Table 4, where the maximum rotation error is set to $\pm 10^\circ$ and the maximum translation error is set to ± 0.2 m. The setup of Exp 2 is similar to Exp 1, but the difference lies in the data split proportions as shown in Table 1. In Experiment 1, the training set proportion is much higher compared to the test and validation sets, allowing the network to receive more comprehensive training. Additionally, the error settings in Exp 2 are slightly smaller than in Exp 1, resulting in better performance overall. The results of Exp 2 are consistent with those of Exp 1. We observed that state-of-the-art methods based on parameter regression still suffer from significant pitch angle errors. Once again, the results of Exp 2 reaffirm the effectiveness of our method. Regarding the qualitative analysis of Exp 2, it is shown in Figure 9.

Table 4. The comparison results between our method and other state-of-the-art (SOTA) approaches under the conditions of Exp 2.

Method	Rotation ($^\circ$)				Translation (cm)			E_t
	Roll	Pitch	Yaw	E_r	X	Y	Z	
NetCalib [13]	0.230	0.970	0.370	0.523	3.86	1.55	1.79	2.40
CalibDepth [3]	0.114	0.955	0.133	0.401	1.07	0.58	0.73	0.79
DPCalib (ours)	0.048	0.166	0.051	0.088	0.88	0.57	0.91	0.78

It is worth noting that both Exp 1 and Exp 2 are comparisons with parameter regression-based methods, while DXQ-Net [31] adopts a different technical approach, namely estimating pixel flow to find corresponding matching points and then deriving external parameters based on the correspondence between points. However, this method involves searching

for pixel points in the image, thus constrained by computational cost, and can only handle small errors. Therefore, their experimental setup limits the maximum angular error to $\pm 5^\circ$ and the maximum displacement error to ± 0.1 m, as specified in Table 1.

Additionally, we reproduced the results of NetCalib in Exp 3. NetCalib, being based on parameter regression, exhibited errors and characteristics consistent with those in Exp1 and 2. NetCalib has a perceptual deficiency in pitch angle perception and poor regression performance for translation errors. The experimental results of DPCalib outperformed NetCalib. We refer to this experimental setup as Exp 3, as shown in Table 3.

The results of Exp 3 are shown in Table 3. Although pixel flow-based and iterative methods do not encounter significant pitch angle errors, they are limited to handling small errors. This is because constructing the cost volume requires searching for matching points pixel by pixel between two images, which can incur significant computational costs if the search range is too large. Additionally, DXQ-Net [31], through iterative updates, mitigates the problem caused by missing perceptual information to some extent. However, even so, our method still performs slightly better than DXQ-Net [31] in estimating translation errors. It is worth noting that our method produces results in a single end-to-end output, whereas DXQ-Net [31] iterates multiple times to obtain results. In this scenario, although our method may have a slight disadvantage in angular error metrics, it still maintains a marginal advantage in translation error. The qualitative analysis results of Exp 3 are depicted in Figure 10. As shown in Figure 10, the parameters estimated by our method can adjust the initially erroneous extrinsic parameters to be nearly consistent with the ground truth.

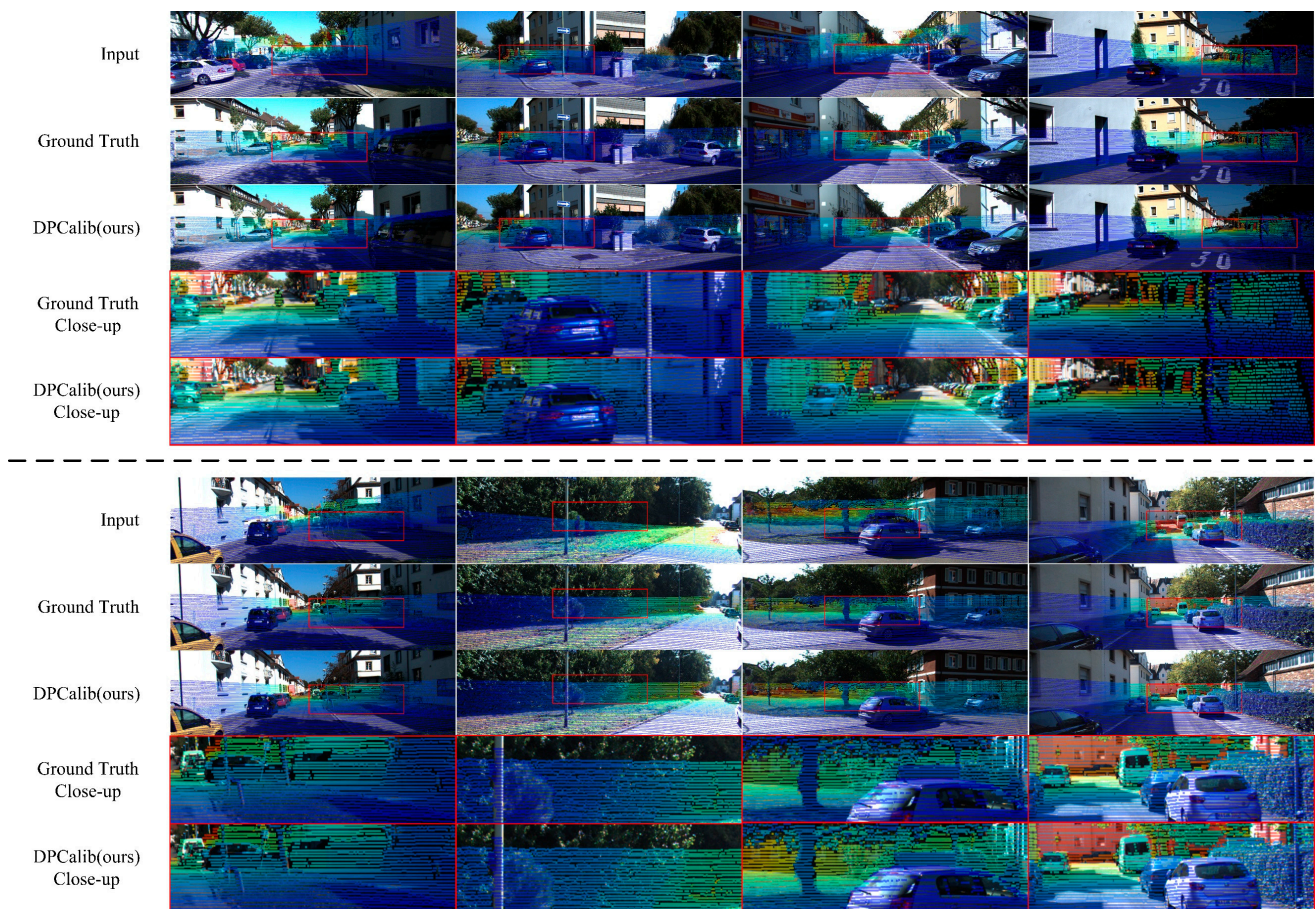


Figure 10. Presents the qualitative analysis results of DPCalib and other existing methods under the settings of Experiment 3. The red boxes indicate areas of that can more prominently reflect the differences between different methods in the local regions of the images.

Figure 11 depicts the histograms and error bars of the results corresponding to the three experiments. Each column represents a set of experimental settings, with the first row representing the histogram of translation errors, the second row representing the histogram of rotation errors, and the third row representing the error bars of both rotation and translation errors. The red segments represent the median of the experimental results, which are typically slightly lower than the average error. The “T” shaped bars and square boxes, respectively, represent the maximum and minimum errors, and the distance from the median to the maximum and minimum errors divided by two. It can be observed that although the input errors are randomly generated, the variance and standard deviation of our method are around the mean value, indicating that our method is stable and robust.

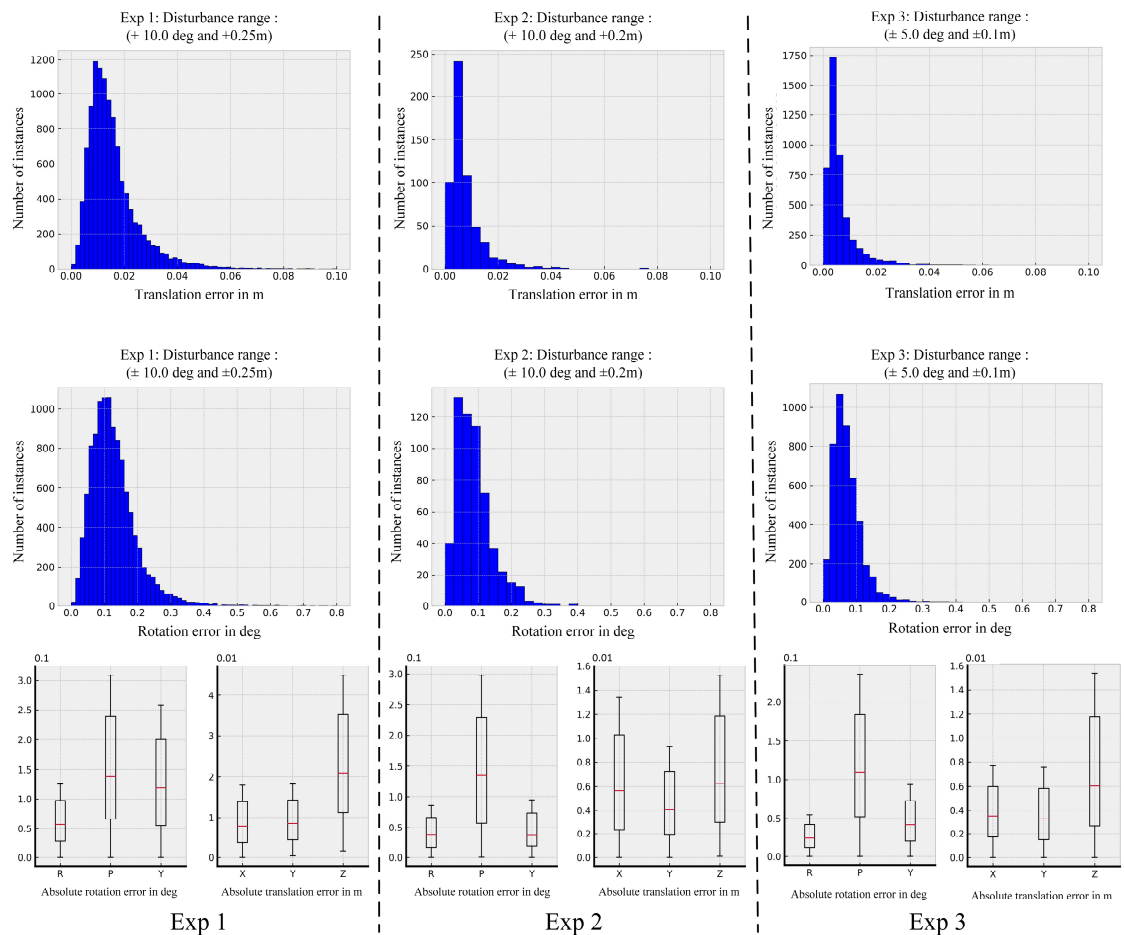


Figure 11. The histograms and box plots of the DPCalib output results, with different columns corresponding to different experimental settings as listed in Table 1.

Additionally, we conducted a comparison and analysis of the real-time performance of our work against the successfully reproduced work, NetCalib [13], serving as a reference. We selected three experimental environments—CPU (Intel i9-12900KF), GPU (NVIDIA RTX 3090), and onboard system (NVIDIA Jetson Orin)—to validate the real-time performance of our work. We conducted inference on 1000 samples from the dataset and calculated the average inference time. The results are shown in Table 5.

Table 5. The experimental results for real-time performance testing.

Methods	GPU	CPU	ORIN
NetCalib	62.75 FPS	2.10 FPS	13.08 FPS
DPCalib	185.18 FPS	16.47 FPS	33.36 FPS

As shown in Table 5, our DPCalib achieves a running speed of 185.18 FPS on an RTX 3090 device and 33.36 FPS on the NVIDIA Jetson Orin edge development board, which is significantly better than NetCalib and meets the real-time requirements. However, it's worth noting that the time shown in Table 2 only reflects the inference time of the algorithm. The data preprocessing steps such as image loading, depth estimation, and projection also consume approximately 1.77 FPS (tested on an RTX 3090 and i9-12900KF environment). In the future, we plan to develop multithreaded tools to further enhance the real-time performance of the model.

Combining the above results, we can conclude that our DPCalib outperforms other parameter regression-based methods, especially in estimating the specific pitch angle of the extrinsic parameters. Additionally, it shows significant improvements in other metrics compared to other state-of-the-art methods. As for methods based on pixel flow estimation, our DPCalib demonstrates the capability to handle larger errors and also exhibits a clear advantage in estimating translation extrinsic parameters. In the section on ablation experiments, we demonstrated the effectiveness and necessity of each module.

5. Ablation Study

This section validates DPCalib from three perspectives: the effectiveness and necessity of each module, the effectiveness and necessity of the training strategy, and the hyperparameters of the building blocks. We conducted ablation experiments on network architecture and hyperparameters following the settings of Exp 2 in Table 1. This choice is made because with large initial errors, it is easier to compare the influence of different network architectures on the final results and improvements. For our ablation experiments on training strategies, we followed the settings of Exp 3 in Table 1. This decision is based on the fact that with relatively small initial errors, it becomes easier to distinguish the subtle differences in experimental results caused by different training strategies.

5.1. Module Architecture

Table 6 summarizes our analysis of the network model's structure, focusing on three aspects: the inclusion of a compensating view, utilization of attention mechanism in feature extraction, and integration of a decoder. The "side" option indicates the presence of the compensating view, "GRU" denotes the incorporation of GRU, and "Atten" signifies the inclusion of an attention-based feature fusion module.

Table 6. The impact of each module of DPCalib on the final results.

Method	Side	GRU	Atten	Rotation (°)			E_r	X	Translation (cm)		
				$Roll$	$Pitch$	Yaw			Y	Z	E_t
NetCalib [13]	-	-	-	0.230	0.970	0.370	0.523	3.860	1.550	1.790	2.400
DPCalib (ours)	-	-	-	0.152	0.578	0.191	0.307	5.633	1.545	2.639	3.272
DPCalib (ours)	-	✓	✓	0.139	0.548	0.189	0.292	5.473	1.483	1.480	2.812
DPCalib (ours)	✓	-	-	0.077	0.220	0.113	0.137	1.122	0.663	1.052	0.946
DPCalib (ours)	✓	✓	-	0.048	0.171	0.053	0.090	0.871	0.579	1.065	0.838
DPCalib (ours)	✓	✓	✓	0.049	0.164	0.052	0.088	0.903	0.598	0.972	0.824

From the experimental data in Table 6, it can be observed that our DPCalib structure performs similarly to NetCalib when not using the auxiliary network, decoder, and attention-based feature extraction structure (referred to as the baseline). Comparing the experiments in the second and third rows of Table 6, we can infer that our feature extraction module and GRU based decoder are effective but not decisive. However, comparing the second to sixth rows, we can conclude that our compensated view features play a crucially positive role in the extrinsic parameter regression. The baseline achieves slightly better estimation accuracy for the rotation parameters compared to NetCalib, while slightly lower accuracy for the translation parameters. However, with the addition of the compensating view network, the performance of the DPCalib network in pitch angle estimation is sig-

nificantly improved, from 0.578 in the baseline to 0.164, which is the most notable finding of this study. Additionally, both the GRU decoder and attention mechanism contribute positively to the performance of the network.

In addition, we further validated the selection of the decoder. We compared our proposed GRU module with the 2D convolutional decoder and a decoder structure based on self-attention. All three decoders have the same number of learnable parameters. The results are shown in Table 7.

Table 7. The impact of fusion mechanism of decoder in the DPCalib on the final results.

Decoder	Rotation (°)				Translation (cm)			E_t	Learnable Params
	Roll	Pitch	Yaw	E_r	X	Y	Z		
2D Convs	0.077	0.220	0.113	0.137	1.122	0.663	1.052	0.946	16.13 M
GRUs	0.049	0.164	0.052	0.088	0.903	0.598	0.972	0.824	16.14 M
Self-Attens	0.057	0.199	0.067	0.108	0.888	0.578	1.103	0.856	17.51 M

As shown in Table 7, with the same order of magnitude of learnable parameters, judging by the mean of the final results, the GRU structure achieved the optimal accuracy. This is because the GRU can efficiently accomplish feature fusion by updating and gating features in the most effective manner. Conversely, relying solely on 2D convolutions cannot achieve this goal. Additionally, the decoder structure based on self-attention requires more learnable parameters to meet the task requirements, resulting in increased computational costs.

5.2. Training Strategy

We conducted ablation experiments on the training strategies mentioned in Section 4.2. We compared the performance of models trained on the test set using three strategies: directly using “sparse ground truth point cloud projection” instead of the estimated depth map of the image as input for the image branch, and first training the model using “sparse ground truth point cloud projection” as input and then fine-tuning the model using the results of image depth estimation as input, as well as directly initializing the model’s weights with random numbers and then training the model using the results of image depth estimation as input. The results are shown in Table 8.

Table 8. The impact of training strategies on DPCalib.

Method	Rotation Error (°)			Translation Error (cm)		
	Mean	Median	Std	Mean	Median	Std
Pretrain only	2.342	1.766	2.100	3.858	4.595	3.235
w/o Pretrain	0.112	0.081	0.114	1.021	0.704	1.015
w/Pretrain + Finetune	0.072	0.058	0.069	0.634	0.428	0.746

In Table 8, the first row labeled “Pretrain only” represents the results obtained from models trained on the training set using sparse projections of ground truth point clouds as input for the image branch. We maintained the dataset partitioning for our training strategy ablation experiments as outlined in Exp 3 of Table 1. This decision was made because different training strategies exhibit more significant results in experiments with small initial errors. We use the mean, median, and standard deviation to compare the accuracy and stability of different training strategies. As observed, the results are far from satisfactory. This is because the depth maps in the test set are dense and contain errors. Although using ground truth for projection can provide relatively accurate sparse depth maps, further fine-tuning is necessary due to the difference in input modalities.

The second row represents the results obtained from models trained directly on the depth estimation results of images as the input of the image branch. The third row

represents the results obtained by first pretraining the model using sparse ground truth as input for the image branch and then fine-tuning it using dense depth maps from images. It can be observed that pretraining with sparse ground truth as input followed by fine-tuning with image depth estimation results is effective. This is because using ground truth point cloud projections as input provides correct geometric information, allowing the neural network to learn the relationship between geometric information and extrinsic parameters. On the other hand, image depth estimation results contain errors, which can make learning difficult for the initially trained model.

5.3. Hyperparameters

We conducted ablation experiments to validate the settings of the network architecture and the intrinsic of the compensation perspective. Firstly, we experimented with the number of layers in the attention module of the neural network's encoder and the iteration count of the GRU in the decoder. The experimental settings followed those of Experiment 2 in Table 1. The results are presented in Table 9.

Table 9. The impact of the layers in the encoder and decoder on experimental results.

GRUs	Attens	Rotation (°)			E_r	X	Translation (cm)		E_t
		Roll	Pitch	Yaw			Y	Z	
0	1	0.050	0.183	0.060	0.098	0.812	0.658	1.105	0.858
1	1	0.049	0.164	0.052	0.088	0.903	0.598	0.972	0.824
2	1	0.052	0.162	0.054	0.089	0.880	0.505	1.078	0.821
1	0	0.041	0.171	0.052	0.088	0.811	0.579	1.065	0.818
1	2.	0.042	0.171	0.050	0.088	0.803	0.558	0.936	0.766

In Table 9, “GRUs” represent the number of iterations performed using a ConvGRU in the network, while “Attens” indicates how many times the attention structure is incorporated during the feature extraction stage. From Table 9, it can be observed that the estimation error for pitch angle is minimized when features are iterated twice through the GRU. This is because multiple iterations can effectively fuse features between the main view and the compensation view. The “Attens” structure in the feature extraction stage mainly affects the overall accuracy of the algorithm, primarily due to the sparse nature of the point cloud projection images.

However, the improvements brought by multiple iterations of these structures are marginal and come with computational and inference time costs. Therefore, we opt for a compromise and select the results from the second row as the experimental results for DPCalib in Experiment 2, which will be compared with other state-of-the-art methods in Table 4.

Additionally, we conducted a qualitative analysis of the predefined parameters for the compensation perspective. The experimental results are illustrated in Figure 12.

From Figure 12, it can be observed that if the focal length is set too large, many pixels in the image may not be hit by the point cloud, resulting in wasted information. Conversely, if the focal length is set too small, the image will be sensitive to errors in the depth estimation results, sometimes leading to ineffective information retrieval. Therefore, after iterative adjustments, we have selected the settings from the second column of Figure 11, where the radial and tangential focal lengths are chosen as 90 and 360, respectively, as the virtual intrinsic for the compensation perspective.

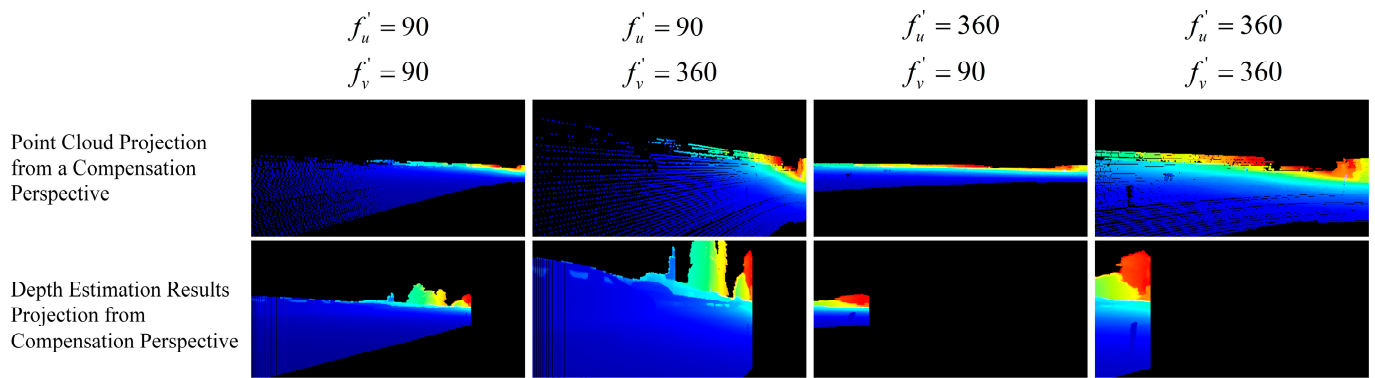


Figure 12. The results of projecting point clouds corresponding to different intrinsic and pseudo point clouds obtained from image depth estimation. The first row displays the projected point cloud images, while the second row illustrates the projection of pseudo point clouds derived from image depth estimation into space and then back onto the image.

5.4. Generalization

This section primarily discusses the generalization capability of DPCalib across different camera configurations and various scenes. Although we lack the resources to autonomously collect large-scale datasets, we can still perform generalization validation of our model using various publicly available datasets. Based on the references [33,34] in revised manuscript, it's evident that there are fundamental differences in sensor configurations, the intrinsic parameter of the target camera in the KITTI_Raw dataset can be represented as (19), while those in the KITTI_Odometry dataset can be represented as (20). Furthermore, there are notable differences in the relative extrinsic between the camera and LiDAR in different scenes.

Additionally, there are significant differences between the data collection locations, and scenarios between the KITTI_Raw dataset and the KITTI_Odometry dataset.

$$K_{raw} = \begin{bmatrix} 984.2439 & 0 & 690.0000 \\ 0 & 980.8141 & 233.1966 \\ 0 & 0 & 1 \end{bmatrix} \quad (19)$$

$$K_{odom} = \begin{bmatrix} 718.8856 & 0 & 607.1928 \\ 0 & 718.8856 & 185.2157 \\ 0 & 0 & 1 \end{bmatrix} \quad (20)$$

In light of these findings, we propose the following experiments to demonstrate the generalization ability of our DPCalib. The core idea of the generalization experiments is to utilize a model trained only on the KITTI_Raw dataset to infer results on the KITTI_Odometry dataset, thereby validating the model's adaptability to new scenes and device configurations. Based on this concept, we have designed the following four sets of experiments:

Exp A1 involves directly using the model from Exp 2 in Table 4 to infer results on the test set of Exp 3.

Exp A2 entails fine-tuning the results of Exp A1 by training for one additional epoch with a learning rate set to 1×10^{-5} .

Exp A3 serves as control groups. Exp A3 involve training models using the training set of Exp 3 for DPCalib to inference on the test set of Exp 3.

The experimental results are presented as shown in Table 10.

Table 10. Generalization ablation experiment results for DPCalib.

Index	Rotation (°)			Translation (cm)				E_t
	Roll	Pitch	Yaw	E_r	X	Y	Z	
Exp A1	0.044	0.148	0.153	0.115	1.174	2.161	2.977	2.104
Exp A2	0.045	0.134	0.055	0.078	0.505	0.532	1.108	0.715
Exp A3	0.030	0.134	0.051	0.072	0.482	0.460	0.958	0.633

Based on the results of Exp A1, it can be concluded that the model trained directly on the KITTI Raw dataset exhibits a certain level of adaptability to the scenes with different sensor configurations and collection scenarios. In terms of rotation angle estimation accuracy, the results from Exp A1 approach the accuracy achieved Exp A3 with KITTI odometry-trained models. However, there is a slight decrease in displacement accuracy compared to models trained directly on odometry data.

However, Exp A2 reveals that after one epoch of fine-tuning, the model's displacement accuracy improves, and its performance further approaches the result of Exp A3 that of models trained directly on the same dataset. From the experimental results in Table 1, it can be observed that directly replacing datasets with different sensor configurations or scenes does not result in a significant decrease in accuracy. This suggests that our model does not overfit a specific dataset, and our method exhibits certain robustness to different scenes and sensor configurations.

6. Discussion on Experimental Results

Section 4 provides detailed explanations of our dataset partitioning, algorithm training, hyperparameters, and implementation details. We also analyze the comparison results between our method and other SOTAs. In Exp 1 and Exp 2, we compared our method with other parameter regression-based methods under conditions of large initial errors. The results demonstrate that our method successfully optimizes the extrinsic performance of pitch angle using compensation perspective information. Additionally, our method exhibits significant superiority in other angles and translation metrics. In Exp 3, we compared our method with methods based on pixel flow estimation. We pointed out that this category of methods can only be used for estimating small errors. Compared with representative work in this category, such as DXQ-Net, the experimental results demonstrate that our method has advantages in the metrics of extrinsic displacement and overall accuracy.

In the section “Ablation study”, we validated the details of our designed network architecture, training strategies, and hyperparameters. Tables 6 and 7 demonstrate the effectiveness and necessity of our network design, while Table 8 illustrates the effectiveness and necessity of our proposed training strategies. Table 9 provides validation of the rationality of the network structure and compensation perspective parameters. Table 10 provides an analysis of the generalization ability of DPCalib. Based on qualitative or quantitative experimental results, we selected reasonable hyperparameters and determined the final version of our DPCalib for comparison with other state-of-the-art methods.

Futhermore, Video S1 presents a continuous video segment for qualitative analysis, comparing the results of our method with those of other methods.

7. Conclusions

This paper proposes a novel and effective neural network model, DPCalib, for joint calibration of LiDAR-camera systems. The main novelty of DPCalib lies in constructing a virtual compensation perspective and utilizing the geometric information from lidar and depth images to project the geometric information from the principal view to the compensation view. Consequently, the neural network's inputs are divided into two branches: the principal view branch, consisting of projected point clouds under erroneous extrinsic and image depth estimation results, and the compensation view branch, consisting of projected point clouds under erroneous extrinsic and depth images projected into point

clouds and then re-projected into images under the compensation view. The features extracted by ResNet18 from these four sets of images are decoded and fused through a GRU, followed by several fully connected layers to decode the estimated extrinsic.

Experimental results demonstrate that our method addresses the shortcomings of end-to-end parameter regression-based methods in estimating pitch angles and outperforms similar methods in terms of accuracy under conditions of large initial errors. While pixel flow regression-based methods can only handle small initial errors within limited computational costs, our method exhibits superior accuracy under their experimental conditions.

Supplementary Materials: The following supporting information can be downloaded at: <https://www.mdpi.com/article/10.3390/electronics13101914/s1>, Video S1: Inputs and Predicted Results in the Test Set.

Author Contributions: Conceptualization, J.C. and X.Y.; methodology, J.C.; software, S.L.; validation, J.C., X.Y. and S.L.; formal analysis, T.T.; investigation, T.T.; resources, J.C.; data curation, T.T.; writing—original draft preparation, Y.L.; writing—review and editing, Y.L.; visualization, J.C.; supervision, S.D.; project administration, S.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original data presented in the study are openly available in <https://www.cvlibs.net/datasets/kitti/> (accessed on 1 April 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Zendel, O.; Huemer, J.; Murschitz, M.; Dominguez, G.F.; Lobe, A. Joint Camera and LiDAR Risk Analysis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, BC, Canada, 17–24 June 2023; pp. 88–97.
2. Guislain, M.; Digne, J.; Chaine, R.; Monnier, G. Fine scale image registration in large-scale urban LIDAR point sets. *Comput. Vis. Image Underst.* **2017**, *157*, 90–102. [[CrossRef](#)]
3. Zhu, J.; Xue, J.; Zhang, P. CalibDepth: Unifying Depth Map Representation for Iterative LiDAR-Camera Online Calibration. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023; pp. 726–733.
4. Guindel, C.; Beltrán, J.; Martín, D.; García, F. Automatic extrinsic calibration for lidar-stereo vehicle sensor setups. In Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 16–19 October 2017; pp. 1–6.
5. Park, Y.; Yun, S.; Won, C.S.; Cho, K.; Um, K.; Sim, S. Calibration between color camera and 3D LIDAR instruments with a polygonal planar board. *Sensors* **2014**, *14*, 5333–5353. [[CrossRef](#)] [[PubMed](#)]
6. Förstner, W.; Gülch, E. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In Proceedings of the ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data, Interlaken, Switzerland, 2–4 June 1987; Volume 6, pp. 281–305.
7. Kim, E.-S.; Park, S.-Y. Extrinsic calibration between camera and LiDAR sensors by matching multiple 3D planes. *Sensors* **2019**, *20*, 52. [[CrossRef](#)] [[PubMed](#)]
8. Yuan, C.; Liu, X.; Hong, X.; Zhang, F. Pixel-level extrinsic self calibration of high resolution lidar and camera in targetless environments. *IEEE Robot. Autom. Lett.* **2021**, *6*, 7517–7524. [[CrossRef](#)]
9. Yuan, K.; Guo, Z.; Wang, Z.J. RGGNet: Tolerance aware LiDAR-camera online calibration with geometric deep learning and generative model. *IEEE Robot. Autom. Lett.* **2020**, *5*, 6956–6963. [[CrossRef](#)]
10. Schneider, N.; Piewak, F.; Stiller, C.; Franke, U. RegNet: Multimodal sensor registration using deep neural networks. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; pp. 1803–1810.
11. Iyer, G.; Ram, R.K.; Murthy, J.K.; Krishna, K.M. CalibNet: Geometrically supervised extrinsic calibration using 3D spatial transformer networks. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1110–1117.
12. Van Brummelen, J.; O'Brien, M.; Gruyer, D.; Najjaran, H. Autonomous vehicle perception: The technology of today and tomorrow. *Transp. Res. Part C Emerg. Technol.* **2018**, *89*, 384–406. [[CrossRef](#)]
13. Wu, S.; Hadachi, A.; Vivet, D.; Prabhakar, Y. NetCalib: A Novel Approach for LiDAR-Camera Auto-Calibration Based on Deep Learning. In Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 10–15 January 2021; pp. 6648–6655.

14. Sun, J.; Shen, Z.; Wang, Y.; Bao, H.; Zhou, X. LoFTR: Detector-free local feature matching with transformers. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 8922–8931.
15. Waswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.; Kaiser, L.; Polosukhin, I. Attention is all you need. In *Proceedings of the NIPS*; Curran Associates: Long Beach, CA, USA, 2017.
16. Huang, Z.; Shi, X.; Zhang, C.; Wang, Q.; Cheung, K.C.; Qin, H.; Dai, J.; Li, H. Flowformer: A transformer architecture for optical flow. In Proceedings of the European Conference on Computer Vision, Tel Aviv, Israel, 23–27 October 2022; pp. 668–685.
17. Dey, R.; Salem, F.M. Gate-variants of gated recurrent unit (GRU) neural networks. In Proceedings of the 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, USA, 6–9 August 2017; pp. 1597–1600.
18. Kwak, K.; Huber, D.F.; Badino, H.; Kanade, T. Extrinsic calibration of a single line scanning lidar and a camera. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA, USA, 25–30 September 2011; pp. 3283–3289.
19. Chen, S.; Liu, J.; Liang, X.; Zhang, S.; Hyypä, J.; Chen, R. A novel calibration method between a camera and a 3D LiDAR with infrared images. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 4963–4969.
20. Tóth, T.; Pusztai, Z.; Hajder, L. Automatic LiDAR-camera calibration of extrinsic parameters using a spherical target. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 8580–8586.
21. Kümmerle, J.; Kühner, T.; Lauer, M. Automatic calibration of multiple cameras and depth sensors with a spherical target. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 1–8.
22. Gong, X.; Lin, Y.; Liu, J. 3D LIDAR-camera extrinsic calibration using an arbitrary trihedron. *Sensors* **2013**, *13*, 1902–1918. [\[CrossRef\]](#)
23. Chen, C.; Lan, J.; Liu, H.; Chen, S.; Wang, X. Automatic calibration between multi-lines LiDAR and visible light camera based on edge refinement and virtual mask matching. *Remote Sens.* **2022**, *14*, 6385. [\[CrossRef\]](#)
24. An, P.; Ma, T.; Yu, K.; Fang, B.; Zhang, J.; Fu, W.; Ma, J. Geometric calibration for LiDAR-camera system fusing 3D-2D and 3D-3D point correspondences. *Opt. Express* **2020**, *28*, 2122–2141. [\[CrossRef\]](#) [\[PubMed\]](#)
25. Pusztai, Z.; Hajder, L. Accurate calibration of LiDAR-camera systems using ordinary boxes. In Proceedings of the IEEE International Conference on Computer Vision Workshops, Venice, Italy, 22–29 October 2017; pp. 394–402.
26. Luo, Z.; Yan, G.; Li, Y. Calib-anything: Zero-training lidar-camera extrinsic calibration method using segment anything. *arXiv* **2023**, arXiv:2306.02656.
27. Wang, W.; Nobuhara, S.; Nakamura, R.; Sakurada, K. Soic: Semantic online initialization and calibration for lidar and camera. *arXiv* **2020**, arXiv:2003.04260.
28. Liu, Z.; Tang, H.; Zhu, S.; Han, S. Semalign: Annotation-free camera-lidar calibration with semantic alignment loss. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 8845–8851.
29. Lv, X.; Wang, B.; Dou, Z.; Ye, D.; Wang, S. LCCNet: LiDAR and camera self-calibration using cost volume network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 20–25 June 2021; pp. 2894–2901.
30. Derpanis, K.G. Overview of the RANSAC Algorithm. *Image Rochester NY* **2010**, *4*, 2–3.
31. Jing, X.; Ding, X.; Xiong, R.; Deng, H.; Wang, Y. DXQ-Net: Differentiable lidar-camera extrinsic calibration using quality-aware flow. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 6235–6241.
32. Zhao, G.; Hu, J.; You, S.; Kuo, C.-C.J. CalibDNN: Multimodal sensor calibration for perception using deep neural networks. In Proceedings of the Signal Processing, Sensor/Information Fusion, and Target Recognition XXX, Online, 12–16 April 2021; pp. 324–335.
33. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets robotics: The kitti dataset. *Int. J. Robot. Res.* **2013**, *32*, 1231–1237. [\[CrossRef\]](#)
34. Geiger, A.; Lenz, P.; Urtasun, R. Are we ready for autonomous driving? The kitti vision benchmark suite. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3354–3361.
35. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26 June–1 July 2016; pp. 770–778.
36. Daubechies, I.; DeVore, R.; Foucart, S.; Hanin, B.; Petrova, G. Nonlinear approximation and (deep) ReLU networks. *Constr. Approx.* **2022**, *55*, 127–172. [\[CrossRef\]](#)
37. Xu, J.; Li, Z.; Du, B.; Zhang, M.; Liu, J. Reluplex made more practical: Leaky ReLU. In Proceedings of the 2020 IEEE Symposium on Computers and Communications (ISCC), Rennes, France, 7–10 July 2020; pp. 1–7.
38. Li, J.; Wang, P.; Xiong, P.; Cai, T.; Yan, Z.; Yang, L.; Liu, J.; Fan, H.; Liu, S. Practical stereo matching via cascaded recurrent network with adaptive correlation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, New Orleans, LA, USA, 18–24 June 2022; pp. 16263–16272.

39. Teed, Z.; Deng, J. Raft: Recurrent all-pairs field transforms for optical flow. In Proceedings of the Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Proceedings, Part II 16, 2020; pp. 402–419.
40. Yao, Y.; Luo, Z.; Li, S.; Shen, T.; Fang, T.; Quan, L. Recurrent mvsnet for high-resolution multi-view stereo depth inference. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 5525–5534.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.