mathematics

MDPI

Article

A Deep Joint Network for Monocular Depth Estimation Based on Pseudo-Depth Supervision

Jiahai Tan [1,2,*], Ming Gao [1], Tao Duan [2] and Xiaomei Gao [3]

[1] School of Optoelectronic Engineering, Xi'an Technological University, Xi'an 710021, China; minggao1964@163.com
[2] State Key Laboratory of Transient Optics and Photonics, Xi'an Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xi'an 710119, China; duantao@opt.ac.cn
[3] Xi'an Mapping and Printing of China National Administration of Coal Geology, Xi'an 710199, China; xiaomei7172@sina.com
* Correspondence: tanjiahai@xab.ac.cn

Abstract: Depth estimation from a single image is a significant task. Although deep learning methods hold great promise in this area, they still face a number of challenges, including the limited modeling of nonlocal dependencies, lack of effective loss function joint optimization models, and difficulty in accurately estimating object edges. In order to further increase the network's prediction accuracy, a new structure and training method are proposed for single-image depth estimation in this research. A pseudo-depth network is first deployed for generating a single-image depth prior, and by constructing connecting paths between multi-scale local features using the proposed up-mapping and jumping modules, the network can integrate representations and recover fine details. A deep network is also designed to capture and convey global context by utilizing the Transformer Conv module and Unet Depth net to extract and refine global features. The two networks jointly provide meaningful coarse and fine features to predict high-quality depth images from single RGB images. In addition, multiple joint losses are utilized to enhance the training model. A series of experiments are carried out to confirm and demonstrate the efficacy of our method. The proposed method exceeds the advanced method DPT by 10% and 3.3% in terms of root mean square error (RMSE(log)) and 1.7% and 1.6% in terms of squared relative difference (SRD), respectively, according to experimental results on the NYU Depth V2 and KITTI depth estimation benchmarks.

Keywords: monocular depth estimation; pseudo-depth net; transformer; encoder–decoder

MSC: 68T07

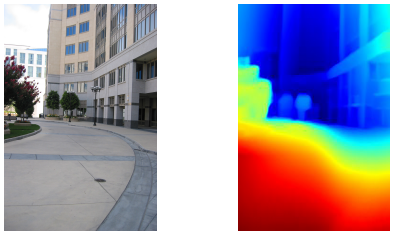
Citation: Tan, J.; Gao, M.; Duan, T.; Gao, X. A Deep Joint Network for Monocular Depth Estimation Based on Pseudo-Depth Supervision. *Mathematics* **2023**, *11*, 4645. https://doi.org/10.3390/math11224645

Academic Editor: Jonathan Blackledge

Received: 6 October 2023
Revised: 30 October 2023
Accepted: 9 November 2023
Published: 14 November 2023

1. Introduction

Monocular depth estimation (MDE) is a challenging task in the field of autonomous driving that aims at recovering the depth map from a given single image (see Figure 1). The estimated depth map is valuable in various intelligent transportation applications, including scene understanding [1], 3D mapping [2], object recognition [3], and obstacle avoidance [4]. Traditionally, the method of obtaining high-precision target depth information is usually to use LiDAR or structured light reflection on the surface of the object to obtain the depth point cloud, but because of its high price and the difficulty of synchronization, there is still a certain distance to be applied and deployed on a large scale in the field of autonomous driving. With the success of Tesla's pure vision program [5] and the stunning effect of Tesla AI Day [6], the camera has become one of the more popular sensor technologies in the field of autonomous driving because of its low price, rich content of acquired information, and compact size. Accordingly, monocular vision depth estimation has also been hotly anticipated by research and has received more attention.



(**a**) (**b**)

Figure 1. Depth estimation from a single image. (**a**) Input a single image. (**b**) Output the corresponding depth map.

With the advancements in convolutional neural networks (CNNs) [7,8], researchers have recently adopted various CNN models to enhance the precision of monocular depth estimation (MDE). Compared to traditional methods, like the approach presented in Saxena et al. [9], CNNs generally achieve higher accuracy. These CNN studies can be categorized into three main groups: (1) unsupervised MDE, (2) supervised MDE, and (3) self-supervised MDE. For instance, supervised MDE methods [10–14] have shown promising results by utilizing ground-truth labels for training. On the other hand, common unsupervised methods [15,16] and self-supervised strategies [17–19] aim to reduce reliance on ground-truth annotations, thereby enabling depth estimation without direct supervision.

CNNs have been the primary tool for depth estimation, using encoder–decoder architectures [20–22]. While most of the work has focused on the design of decoders [20,22], recent studies have shown that encoders are more important for accurate depth estimation [21–23]. A competent encoder must be able to efficiently leverage both local information, which refers to consistency within an item, and long-range dependencies, which refer to distance relationships between objects, because of the lack of depth information [24]. The encoder, where convolutional operators are almost unable to simulate long-range correlations in a constrained receptive field, may, therefore, be the bottleneck of existing depth estimation approaches. As an alternative to CNNs, the Vision Transformer (ViT) [25] has achieved great success in image recognition tasks, where it demonstrates the advantages of being a depth estimation encoder. By utilizing attention mechanisms, the transformer excels in establishing long-range dependency models with global receptive fields. However, ViT encoders lack spatial inductive bias when modeling local information [26], resulting in unsatisfactory performance in near-range depth estimation. In contrast, models that employ convolutional encoders yield better predictions for these regions. Thus, a significant research problem lies in successfully integrating Transformers and convolutional encoders for improved performance. In addition, since the depth estimation problem is a standard regression problem, the loss function is typically the mean square error (MSE) in log space or one of its variants. Although the optimized regression network produced reasonable solutions, the convergence rate was found to be rather slow, and the final solution was far from satisfactory. Therefore, this paper focuses on addressing two problems in existing monocular depth estimation methods. One is how to effectively integrate the transformer and convolutional encoder to improve performance. The other is how to design a matching loss function for the constructed model for better optimization.

To address these challenges, a depth estimation method is proposed based on an encoder–decoder architecture that combines pseudo-depth estimation and a depth estimation network. The proposed method enables end-to-end learning of the mapping from a single image to a depth map. The proposed method starts by encoding the input image into a feature representation. Then, the encoded feature is decoded to generate the corresponding depth map. In order to maintain dense pixel-wise output, the spatial information from the encoder is directly preserved and transferred to the corresponding decoder, without the need for additional parameters or operations. This helps preserve the spatial information and improve the accuracy of the depth estimation process.

(1) Our method introduces a combined pseudo-depth and depth module network that aims to provide both coarse and fine features to accurately predict high-quality depth images from a single RGB image.

(2) The pseudo-depth network utilizes upsampling mapping, residual modules, and an improved codec to directly obtain a depth map prior. The depth network employs an effective global Transformer strategy and Unet depth network to enhance the training of the model, which greatly improves estimation accuracy.

(3) Our method makes collaborative use of several loss functions to improve network training. By using multiple losses, the model can capture different aspects of depth estimation and improve overall performance.

The structure of the paper is as follows. In Section 2, the pertinent literature and earlier studies in this area are covered. The proposed method is fully explained in Section 3. In Section 4, the experimental findings are discussed. Section 6 concludes with a summary of the research and conclusions made in this work.

2. Related Work

2.1. Traditional Methods

MDE is a mature and challenging research area in computer vision and autonomous driving. Initially, researchers heavily relied on manual feature engineering and probabilistic graphical models to tackle this problem. For instance, Saxena et al. [24] successfully employed a combination of absolute and relative depth features, along with the utilization of Markov random fields (MRFs), to accurately predict the depth of monocular images. Building upon this work, Saxena et al. [9] extended their approach to 3D scene reconstruction. Liu et al. [27] used semantic labels as contextual information. Their approach involved employing a learned MRF to infer the semantic category of each pixel in the image, followed by the application of the L-BFGS technique to create a pixel-depth image.

Unlike the previously mentioned parameter-dependent approaches, Karsch et al. [28] treated the MDE as a non-parametric problem. In the pixel shift–based approach proposed by Karsch et al. [28], given an input image, similar images are first searched for within the dataset by comparing GIST features. Then, by transferring the labels from the input image to the matched image, a range of potential depth values for the scene is created. To address the challenges of over-smoothing and preserving occlusion boundaries in the predicted depth map, Liu et al. [29] devised a discrete-continuous condition matrix (CRF) approach. By utilizing this technique, they were able to generate a depth map that avoided excessive smoothing while maintaining accurate boundaries between occluded regions.

Admittedly, the above methods rely heavily on manually created features to predict depth values. These features are carefully designed to capture specific characteristics and patterns in the input data. However, one drawback of such pre-designed features is that they may not generalize well when applied to new and unfamiliar environments.

2.2. Deep Learning–Based Methods

The success of CNN in various computer vision tasks has also prompted the exploration of CNN-based depth prediction methods. Several notable studies [10,30–33] have addressed the CNN-based depth prediction problem. A monocular depth estimation method based on deep learning was pioneered by Eigen et al. [10]. Two networks were employed in their approach: the first one used the complete input image to predict a global depth map, and the second network improved the global prediction locally. Building upon their initial work, Eigen and Fergus [31] extended the method to incorporate multitask learning, which involved jointly learning depth prediction with other related tasks. Liu et al. developed DCNF that combines the strengths of the deep CNN and continuous conditional random fields (CRFs) within a unified framework to achieve accurate depth prediction. It is worth noting that the above methods [10,32] use fully connected (FC) layers, which involve numerous parameters and lead to high computational costs. This motivates further research into exploring more efficient and lightweight network architectures for MDE tasks.

Laina et al. [11] suggested a fully convolutional residual network (FCRN) for MDE in order to circumvent the delay brought on by the FC layer. Based on the fully convolutional portion of ResNet-50, the FCRN encoder creates feature maps at a 1/32 scale relative to the input image. The final depth map is produced by the FCRN decoder by combining these feature maps. Further research built upon the FCRN framework by exploring the impact of the depth of the encoder network on depth estimation accuracy. Studies by [34–38] increased the depth of the encoder network to over 100 layers. It was found that deeper networks with a larger reception range resulted in improved performance compared to using a variant of ResNet-50. Additional enhancements were made by Hu et al. [36] and Chen et al. [37], who incorporated multiscale features and reconstruction modules into the network. In order to further enhance performance, Cao et al. [34] developed depth estimation as a pixel classification job and used a fully connected conditional random field (CRF) as a post-processing technique. Li et al. [35] utilized multi-scale characteristics to estimate depth based on the outputs of several layers. Godard et al. [39] utilized the loss of left–right disparity coherence for the stereo dataset. Bian et al. [16,17,40,41] explored self-supervised approaches with extensions for scale inconsistency, rotation, dynamic objects, and object boundary blurring. Klingner et al. [42] improved depth estimation for moving objects using a novel semantic mask. Lee et al. [21] designed a multiscale local plane guidance layer. Yang et al. [26] combined transformers and CNNs, while Bhat et al. [22] proposed a transformer-based architecture for adaptive depth estimation. Ranftl et al. [23] introduced the dense prediction transformer that utilizes markers from different stages to improve predictions. These approaches aim to enhance depth estimation results and incorporate global information. Overall, these research contributions demonstrate the ongoing efforts to advance monocular depth estimation and address its limitations through innovative algorithmic strategies and network architectures.

The proposed method aims to reduce complexity and the number of parameters while improving the depth estimation results. The backbone network plays a crucial role by constructing a hierarchical representation of linear complexity for the input image. This hierarchical representation allows for multiscale high-level feature extraction, which is then fed into a back-end network. The back-end network leverages a multilevel localized planar bootstrap layer to predict depth. This layer effectively combines the benefits of convolutional neural networks (CNNs) and transformers. It utilizes a transformer representation learning approach with linear complexity, which helps reduce computational requirements while maintaining strong inference capabilities. By combining the strengths of CNNs and transformers, our method aims to achieve accurate depth estimation with reduced complexity and improved efficiency. This approach demonstrates the ongoing exploration of hybrid techniques to optimize network architectures for depth estimation tasks.

3. The Proposed Method

3.1. Framework Overview

Figure 2 provides an overview of the proposed method, which involves training two networks: the Pseudo-depth Net and the Depth Net. These networks are trained jointly using a large dataset of monocular videos. Given an RGB input I, the pseudo-depth and depth CNNs first estimate their pseudo-depth maps PD and depth maps D, respectively. Eachnetwork is then supervised based on the loss between the true depth image D^{prime} and the generated image D. In addition, the normal matching loss L_N and the edge-aware loss L_{edge} between the pseudo-depth image PD and the depth image D provide additional supervised information to the network. Next, the Pseudo-depth Net (see Figure 3) and Depth Net networks (see Figure 4) are described separately.

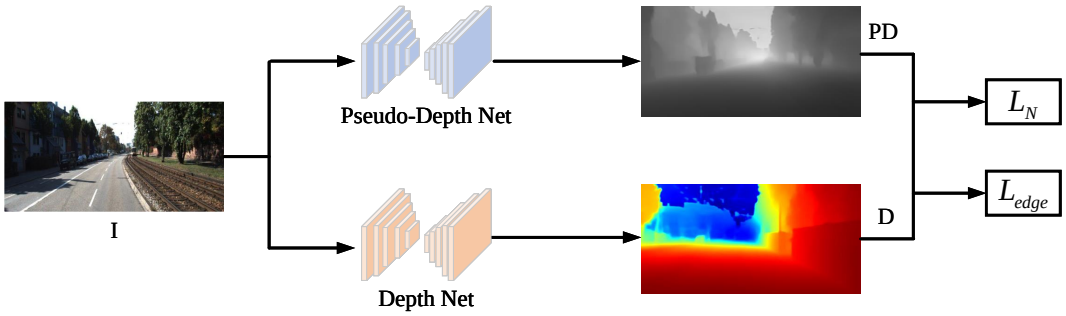


Figure 2. Network architecture for our depth estimate method.

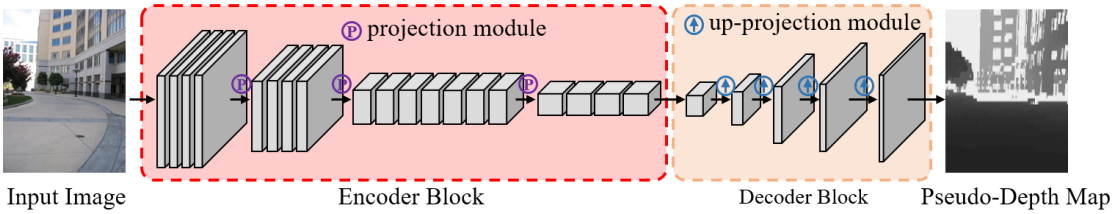


Figure 3. Network architecture for our Pseudo-depth Net.

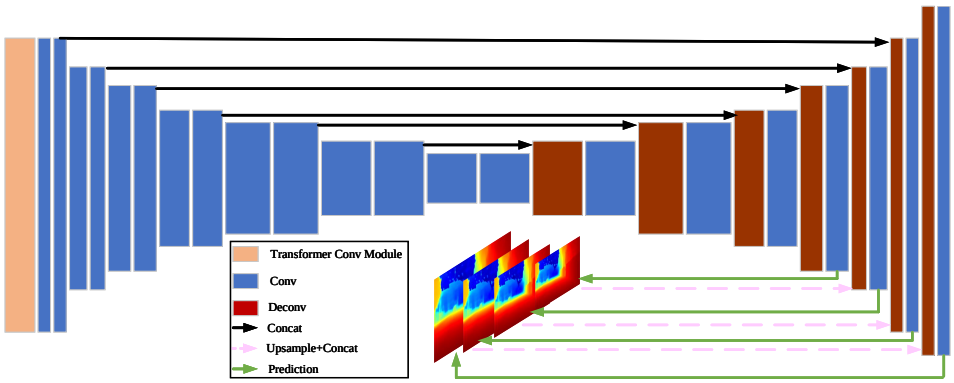


Figure 4. Network structure of Depth Net networks.The width and height of each rectangular block denote the spatial dimensions of the output channels and feature maps of the corresponding layer, respectively. Each decrease/increase in size signifies a change by a factor of two, with the exception of the first to the fourth convolutional layers, which have kernel sizes of 7, 7, 5, and 5. The first convolutional layer has an output of 32 channels. The Transformer Conv module and an Unet structure with multiscale side preconditioning are used to process the input color image.

3.2. Pseudo-Depth Net

Pseudo-Depth-Encoder Block. Figure 3 illustrates an overview of the pseudo-depth network. Pseudo-depth Net uses pre-trained ResNet-50 as an encoder block, which inputs 304×228 pixels. To overcome the gradient vanishing problem, ResNet-50 constructs a multilayer structure using jump connections. In addition, ResNet-50 has a large receptive field to capture a wide range of spatial information from input images, which is lightweight but has superior performance. The details are listed in Table 1. The encoder block first performs 7×7 convolution and 3×3 max-pooling on the input image with a stride of 2. Then, the encoder block contains a repeated application of residual learning, including skip modules and projection modules. The skip module exploits a shortcut connection to skip three layers and performs residual mapping. As shown in Figure 5a, the skip module performs identity mapping without extra parameters, which is formulated as

$$z_{out} = W_3\sigma(W_2\sigma(W_1 z_{in})) + z_{in}, \quad (1)$$

where z_{in} and z_{out} are the input and output features, respectively. $\{W_1, W_2, W_3\}$ denote three convolution operations, and σ is the rectified linear unit (ReLU). The details are listed in Table 2. As depicted in Figure 5b, the projection module performs a shortcut connection with linear projection W_4, which is formulated as

$$z_{out} = W_3\sigma(W_2\sigma(W_1 z_{in})) + W_4 z_{in}, \tag{2}$$

where z_{in} and z_{out} are the input and output features, respectively. $\{W_1, W_2, W_3, W_4\}$ denote the convolution operations, and σ is the ReLU. The specific details can be found in Table 3. The original ResNet-50 contains a series of convolutions and pooling operations, which decrease the resolution of the feature map. To retain the spatial information, the original ResNet-50 architecture eliminates both the FC layer and the last pooling layer. The encoder block can be formulated as

$$u = f_{en}(x) \tag{3}$$

where $u \in \mathbb{R}^{10\times8\times2048}$ is the output feature map and $f_{en}()$ encodes the input image x. The encoder block produces 2048 feature maps with a spatial resolution of 10×8 pixels.

Pseudo-Depth-Decoder Block. The encoder block reduces the spatial resolution of the input image, which is 304×228 pixels, to 10×8 pixels. Reversing the pooling operation with unpooling layers improves the spatial resolution of the feature map u. The unpooling layer upscales the feature map by mapping the element into the top-left corner and fills the holes with zeros. After the 2×2 unpooling layer, 5×5 convolution is performed to avoid zero elements, followed by a ReLU activation function. Inspired by residual learning, skip connections are considered to propagate context information, which is named the up-projection module. According to Table 4, a 3×3 convolution is followed by a 5×5 convolution.

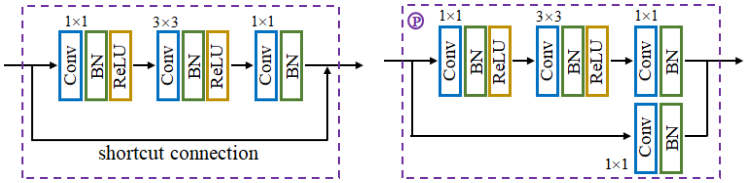


Figure 5. The encoder block contains two different residual learning. (**a**) The skip module; (**b**) the projection module.

Additionally, a projection connection is established from the unpooling layer to the 3×3 convolution. The up-projection module is formulated as

$$z_{out} = \sigma(W_2\sigma(W_1 up(z_{in})) + W_3 up(z_{in})), \tag{4}$$

where up represents the 2×2 unpooling operation, $\{W_1, W_2, W_3\}$ denote three convolution operations, and σ is the ReLU. z_{in} and z_{out} are the input and output features, respectively. To estimate the depth maps, four up-projection modules are stacked to upscale the feature map u ($2\times$ resolution per block). As shown in Figure 6, the up-projection module allows large feature channels, which can propagate context information to the depth map. Our model uses four up-projection modules to predict an output map with 160×128 pixels, which is approximately half the input resolution 304×228. The prediction maps are up-sampled using bilinear interpolation to their original size before being compared to the provided ground truth. The decoder block can be formulated as

$$\hat{y} = f_{de}(u), \tag{5}$$

where $f_{de}()$ decodes the feature map $u \in \mathbb{R}^{10\times8\times2048}$ to reconstruct the depth map $\hat{y}$.

Table 1. The architecture of pseudo-depth-encoder block.

Layer	Input	Output	Details
Input_image	-	In	Input size: 304 × 228 × 3
Convolution_1	In	Conv_1	Kernel number: 64, Kernel size: 7 × 7, stride: 2
Batch Norm_1	Conv_1	BN_1	
ReLU_1	BN_1	ReLU_1	
Projection_1	ReLU_1	Pro_1	Kernel number: 256
Skip_1	Pro_1	S_1	Kernel number: 256
Skip_2	S_1	S_2	Kernel number: 256
Projection_2	S_2	Pro_2	Kernel number: 512
Skip_3	Pro_2	S_3	Kernel number: 512
Skip_4	S_3	S_4	Kernel number: 512
Skip_5	S_4	S_5	Kernel number: 512
Projection_3	S_5	Pro_3	Kernel number: 1024
Skip_6	Pro_3	S_6	Kernel number: 1024
Skip_7	S_6	S_7	Kernel number: 1024
Skip_8	S_7	S_8	Kernel number: 1024
Skip_9	S_8	S_9	Kernel number: 1024
Skip_10	S_9	S_10	Kernel number: 1024
Projection_4	S_10	Pro_4	Kernel number: 2048
Skip_11	Pro_4	S_11	Kernel number: 2048
Skip_12	S_11	S_12	Kernel number: 2048
Convolution_2	S_12	Conv_2	Kernel number: 1024, Kernel size: 7 × 7, stride: 1
Batch Norm_2	Conv_2	BN_2	
Up-projection_1	BN_2	U_1	Kernel number: 512
Up-projection_2	U_1	U_2	Kernel number: 256
Up-projection_3	U_2	U_3	Kernel number: 128
Up-projection_4	U_3	U_4	Kernel number: 64
Convolution_3	U_4	Conv_3	Kernel number: 1, Kernel size: 3 × 3, stride: 1
ReLU_3	Conv_3	Out	

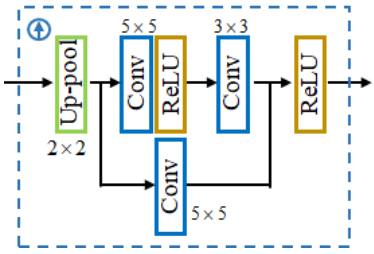


Figure 6. The up-projection module.

Table 2. The skip module.

Layer	Input	Output	Details
Input_feature	-	In	Input size: M × N × C
Convolution_1	In	Conv_1	Kernel size: 1 × 1, stride: 1
Batch Norm_1	Conv_1	BN_1	
ReLU_1	BN_1	ReLU_1	
Convolution_2	ReLU_1	Conv_2	Kernel size: 3 × 3, stride: 1
Batch Norm_2	Conv_2	BN_2	
ReLU_2	BN_2	ReLU_2	
Convolution_3	ReLU_2	Conv_3	Kernel size: 1 × 1, stride: 1
Batch Norm_3	Conv_3	BN_3	
Skip Connection	In, BN_3	SC	
ReLU_3	SC	Out	

Table 3. The projection module.

Layer	Input	Output	Details
Input_feature	-	In	Input size: M × N × C
Convolution_1	In	Conv_1	Kernel size: 1 × 1, stride: 1
Batch Norm_1	Conv_1	BN_1	
ReLU_1	BN_1	ReLU_1	
Convolution_2	ReLU_1	Conv_2	Kernel size: 3 × 3, stride: 1
Batch Norm_2	Conv_2	BN_2	
ReLU_2	BN_2	ReLU_2	
Convolution_3	ReLU_2	Conv_3	Kernel size: 1 × 1, stride: 1
Batch Norma_3	Conv_3	BN_3	
Convolution_4	In	Conv_4	Kernel size: 1 × 1, stride: 1
Batch Norma_4	Conv_4	BN_4	
Skip Connection	BN_3, BN_4	SC	
ReLU_3	SC	Out	

Table 4. The up-projection module.

Layer	Input	Output	Details
Input_feature	-	In	Input size: M × N × C
Up-pooling	In	Up	2 × 2 upsampling
Convolution_1	Up	Conv_1	Kernel size: 5 × 5, stride: 1
ReLU_1	Conv_1	ReLU_1	
Convolution_2	ReLU_1	Conv_2	Kernel size: 3 × 3, stride: 1
Convolution_3	Up	Conv_3	Kernel size: 5 × 5, stride: 1
Skip Connection	Conv_2, Conv_3	SC	
ReLU_2	SC	Out	

3.3. Depth Net

For depth prediction, the Transformer Conv block and Depth-Unet architecture are adopted. From a tensor $X \in \mathbb{R}^{\hat{H} \times \hat{W} \times \hat{C}}$, the initial step in our Transformer Conv module (as shown in Figure 7) is to create query (Q), key (K), and value (V) projections, which are improved with local context. The cross-channel context in terms of pixels is collected by a 1×1 convolutional layer. Then, to represent the spatial context of the channel, a 3×3 deep convolutional layer is used. This process ultimately produces the desired output.

$$Q = F_d^Q F_p^Q X, \tag{6}$$

$$K = F_d^K F_p^K Y, \tag{7}$$

$$V = F_d^V F_p^V X, \tag{8}$$

where 1×1 point-wise convolution is represented by $F_p^{\cdot}$. The 3×3 depth-wise convolution is denoted by the symbol $F_d^{\cdot}$. The network uses bias-free convolutional layers. The query and key projections are then rearranged to form a transposed attention map A of size $\mathbb{R}^{\hat{C} \times \hat{C}}$. The transformer conv module is generally described as

$$\hat{X} = W_p \cdot \hat{V} \cdot Softmax(\hat{K} \cdot \hat{Q} / \alpha), \tag{9}$$

where the output feature map is represented by $\hat{X}$. The learnable parameters are denoted by the expression $W \in \mathbb{R}^{C \times C}$. After reshaping tensors from the original size, the following matrices are produced: $\hat{Q} \in \mathbb{R}^{\hat{H}\hat{W} \times \hat{C}}$, $\hat{K} \in \mathbb{R}^{\hat{H}\hat{W} \times \hat{C}}$, and $\hat{V} \in \mathbb{R}^{\hat{H}\hat{W} \times \hat{C}}$. Before using the softmax function, the magnitude of the dot product between $\hat{K}$ and $\hat{Q}$ is modified using the learnable scaling parameter α.

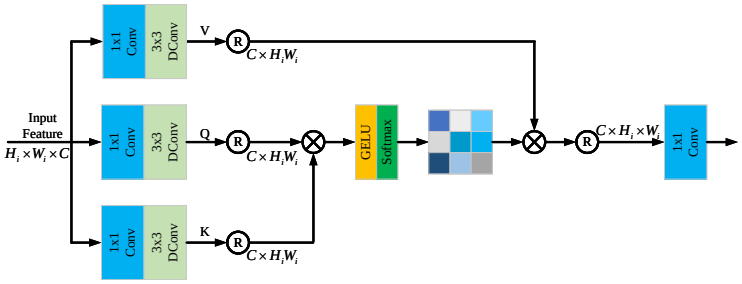


Figure 7. Network structure of Transformer Conv module.

As illustrated in Figure 4, the Depth-Unet architecture uses a skip connection encoder–decoder design with multi-scale side predictions. ReLU activation is applied to all convolutional layers, aside from the prediction layers. The function $1/(\alpha * \text{sigmoid}(x) + \beta)$, where $alpha = 10$ and $beta = 0.1$, is used to constrain the depth values in the prediction layers to be positive within a tolerable range. The Depth-Unet consists of multiple contraction parts and extension parts that are linked together over a long range. Convolutional layers with occasional strides of 2 make up the contraction portion, which results in a 64-downsampling factor overall. This makes it possible for the network to estimate significant depth map displacements. By hopping connections, the network's expansion portion gradually and nonlinearly up-samples the feature map and includes data from the contraction. A number of up-convolutional and convolutional layers are used to accomplish this. Since information can also move through long-distance links between the contraction and expansion layers, there are no data bottlenecks in the network. See Table 5 for more information.

Table 5. Specification of the Depth-UNet structure. Convolutions conv1 through conv6b make up the contracting portion. Up convolutions (upconvN), convolutions (iconvN, prN), and loss layers alternate throughout the expanding portion. Higher-layer features are concatenated with features from lower levels. pr1 produces the predicted disparity image.

Layer	Input	Output	Details
conv1	-	conv1	Kernel size: 7 × 7, stride: 2
conv2	conv1	conv2	Kernel size: 5 × 5, stride: 2
conv3a	conv2	conv3a	Kernel size: 5 × 5, stride: 2
conv3b	conv3a	conv3b	Kernel size: 3 × 3, stride: 1
conv4a	conv3b	conv4a	Kernel size: 3 × 3, stride: 2
conv4b	conv4a	conv4b	Kernel size: 3 × 3, stride: 1
conv5a	conv4b	conv5a	Kernel size: 3 × 3, stride: 2
conv5b	conv5a	conv5b	Kernel size: 3 × 3, stride: 1
conv6a	conv5b	conv6a	Kernel size: 3 × 3, stride: 2
conv6b	conv6a	conv6b	Kernel size: 3 × 3, stride: 1
pr6+loss6	conv6b	conv6b	Kernel size: 3 × 3, stride: 1
upconv5	conv6b	upconv5+pr6+conv5b	Kernel size: 4 × 4, stride: 2
iconv5	upconv5+pr6+conv5b	iconv5	Kernel size: 3 × 3, stride: 1
pr5+loss5	iconv5	iconv5	Kernel size: 3 × 3, stride: 1
upconv4	iconv5	upconv4+pr5+conv4b	Kernel size: 4 × 4, stride: 2
iconv4	upconv4+pr5+conv4b	iconv4	Kernel size: 3 × 3, stride: 1
pr4+loss4	iconv4	iconv4	Kernel size: 3 × 3, stride: 1
upconv3	iconv4	upconv3+pr4+conv3b	Kernel size: 4 × 4, stride: 2
iconv3	upconv3+pr4+conv3b	iconv3	Kernel size: 3 × 3, stride: 1
pr3+loss3	iconv3	iconv3	Kernel size: 3 × 3, stride: 1
upconv2	iconv3	upconv2+pr3+conv2	Kernel size: 4 × 4, stride: 2
iconv2	upconv2+pr3+conv2	iconv2	Kernel size: 3 × 3, stride: 1
pr2+loss2	iconv2	iconv2	Kernel size: 3 × 3, stride: 1
upconv1	iconv2	upconv1+pr2+conv1	Kernel size: 4 × 4, stride: 2
iconv1	upconv1+pr2+conv1	iconv1	Kernel size: 3 × 3, stride: 1
pr1+loss1	iconv1	output	Kernel size: 3 × 3, stride: 1

3.4. Loss Function

There are also several other well-established loss functions [39,43] that have been found to perform accurate depth estimation and reconstruction tasks using CNN. It has been demonstrated that structural similarity index measure [44] (SSIM) loss performs well in these tasks. The anticipated and actual depth maps' structural and luminance similarities are both taken into account by the SSIM loss. The loss function offers further in-depth details on the application and efficiency in complex estimate and reconstruction jobs.

$$L_{SSIM(y,\hat{y})} = \frac{(2 \cdot Avg_y \cdot Avg_{\hat{y}} + a)(Converge_{y\hat{y}} + b)}{(Avg_y^2 + Avg_{\hat{y}}^2 + a)(Var_y^2 + Var_{\hat{y}}^2 + b)}, \tag{10}$$

where the values for the anticipated depth map and the ground depth map, respectively, are y and $haty$. The average and variance are denoted as Avg and Var^2, respectively. The network's convergence constants are a and b.

Another common loss function is the mean squared error:

$$L_2 = \|y - \hat{y}\|_2^2, \tag{11}$$

where the predicted map $\hat{y}$ should be at approximately the ground truth map y. The network is trained using stochastic gradient descent using the input images x and the accompanying depth maps y.

The normal matching loss is used to compute the loss between the surface normals of the predicted depth map and the pseudo-depth map. Formally,

$$L_N = \frac{1}{K}\sum_{k=1} \|n_i - n_i^*\|_1 + \frac{1}{K}\sum_{k=1} \|n_i - \hat{n}_i\|_1, \tag{12}$$

where the surface normals based on the predicted depth, pseudo-depth, and ground-truth map are n_i, n_i^*, and $\hat{n}_i$, respectively. The entire number of pixels in the image is represented by N. The overall depth structure is strongly supervised by the pixel loss function.

Normals, essential geometric features, constitute a complimentary modality to depth. Edge-aware loss is utilized for structure-oriented ordering to improve the clarity of edges and thus ensure accurate depth estimation. We use surface normal maps to find planar regions where the normals are nearly identical and regions where the normals change significantly. We then track and sample paired points [45] on either side of these significantly varying edges. Within a small distance of the edge points, two points are randomly selected on each side, ensuring that these four points lie on a line orthogonal to the sampled edge points. This results in three pairs of points for sorting loss in Figure 8: (a,b), (b,c), and (c,d). After converting an image to grayscale, the gradient mappings M_y and $M_{\hat{y}}$ are obtained, along with the gradient size map M. The gradient size map is then thresholded in order to compute the edge E.

$$E = [M \geq \alpha \cdot max(M)], \tag{13}$$

where α is used to control the threshold of E density. For each edge point $e = (x, y)$ sampled from E, four edge points $e = (x, y)$ $[(x_k, y_k), k = a, b, c, d]$ are sampled by

$$\begin{cases} x_k = & x + \delta_k M_x(e)/M(e) \\ y_k = & y + \delta_k M_y(e)/M(e), \end{cases} \tag{14}$$

We sample them within a moderate distance β from the edge point e to obtain $\delta_a < \delta_b < 0 < \delta_c < \delta_d$. The values of α and β are set to 0.1 and 30, respectively. A margin of two pixels on each side of the edge is also provided in order to avoid sampling points too close to the edge point e, which would make it difficult to determine the ground-truth depth value. The whole sampling process is summarized in Algorithm 1.

In planar regions, paired points are also sampled in the same plane. In addition, to improve the global geometric quality, we globally randomize the sampling of paired points. Thus, the overall structure is enhanced, and the boundary regions of the object are also emphasized. More specifically, the approach involves sampling point pairs around the edges of the image and ensuring that the relative angle of the normal vectors of these sampled points is consistent with the pseudo-depth. To achieve this, the technique known as edge-guided sampling [45] is employed to create the point pairs <A, B>. Edge-aware loss is defined as follows:

$$L_{edge} = \frac{1}{N}\sum_{i=1}^{N} \|n_{A_i} \cdot n_{B_i} - n_{A_i}^* \cdot n_{B_i}^*\|_1 + \frac{1}{N}\sum_{i=1}^{N} \|n_{A_i} \cdot n_{B_i} - \hat{n}_{A_i} \cdot \hat{n}_{B_i}\|_1, \tag{15}$$

where the normals of the sampled points from the predicted depth map are represented by n_A and n_B. n_A^* and n_B^* stand for the normals of sampled points from the pseudo-depth map. $\hat{n}_{A_i}$ and $\hat{n}_{B_i}$ are the normal of sampled points from ground-truth map. Edge-

guided sampling and the relative normal loss can be used together to constrain the depth estimation in the object border region effectively. This approach ensures that the predicted depth values accurately represent the depth variations at the edges of the object. By utilizing the information from the edge-guided sampling, the network can better understand the object boundaries and refine the depth estimation accordingly. The relative normal loss further strengthens the consistency of the surface normal between the predicted depth map and the pseudo-depth map, helping to maintain accurate depth estimation, specifically in the boundary region of the object. Overall, this combined approach provides a more precise and reliable depth estimation in the object boundary region.

In summary, our objective function is defined as follows:

$$L = L_{SSIM} + L_2 + L_N + L_{edge}. \tag{16}$$

Algorithm 1: The procedure for edge-guided sampling

Input: Edge masks E, gradient maps M_x, M_y and gradient magnitude M, number of edge pixels L to be sampled

Output: point pair set S

Initial: Sampled points $S = \emptyset$;
for $i = 1, 2, \ldots, L$ **do**
    Sample an edge point e;
    Sample 4 points $[(x_k, y_k), k = a, b, c, d]$ according Equation (14);
    Add (a, b), (b, c), and (c, d) to S;
end
return

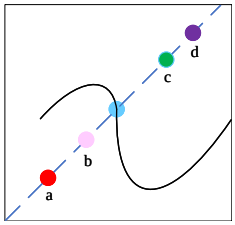


Figure 8. The procedure of edge-guided sampling.

4. Experiments

4.1. Experimental Setup

(1) Datasets

We evaluated the proposed method on three benchmark datasets (NYU-Depth-v2 [46], KITTI [47], and DDAD [48]) in both indoor and outdoor scenarios. About 240,000 RGB and depth image pairs from 464 distinct indoor settings captured with a Microsoft Kinect camera make up the NYU-Depth-v2 dataset. Around 48,000 synchronized RGB and depth image pairs from this dataset were used to train our method. Additionally, 654 images were used for testing. To align with previous literature [10,11], the original images were first reduced to half their size from 640 × 480 pixels. Then, a region of 304 × 228 pixels was centered and cropped from the images to serve as the input to the network. The KITTI dataset, on the other hand, focuses on real-world outdoor scenes and consists of high-resolution outdoor images with dimensions of 375 × 1241 pixels. Similar to Eigen et al. [10], we utilized only the left-side images for our method. The KITTI dataset contains 22,600 training images and 697 test images. The associated Velodyne data points are projected onto the left image plane to create ground-truth depth maps. Missing depth values in the ground-truth depth maps were not taken into account during training or testing. For this study, only the bottom 228 × 912 pixel region was utilized as the LiDAR measurements could not capture the upper part of the image. By using these two diverse datasets, spanning indoor and outdoor scenes,

we ensured a comprehensive evaluation of the proposed method's performance in different environments and scenarios. The DDAD dataset contains 200 driving videos taken in urban scenes. The point clouds that were scanned by LiDAR are given. Compared to the KITTI dataset, the DDAD dataset has almost all vehicles traveling on the road and fewer parked vehicles, which made it more challenging for the model training. We divided the dataset according to the standard training/testing subset, which contained 150 training scenarios (totaling 12,650 images) and 50 validation scenarios (totaling 3950 images). We used the validation scenes for model performance evaluation. During training, the resolution of the images was scaled to 640 pixels $\times$ 384 pixels.

(2) Training details

In our method, the network was trained to predict depth maps using RGB inputs. The network architecture was implemented using the PyTorch framework. We utilized Python 3.7 and CUDA 11.6 on Ubuntu 20.04 to create an MDE model. The model was trained on the NYU-Depth-v2 and KITTI datasets using an NVIDIA RTX 3090 with 24 GB of RAM. Except for the first layer, which included a variable number of input channels, the ResNet weights in the encoder block were initialized using models that had already been trained on the ImageNet [49] dataset. The network was trained using the AdamW [50] optimizer with 100k iterations on each dataset, with the learning rate set to 10^{-4}.

(3) Data Augmentation

Several data augmentation techniques were employed to increase the diversity of the training samples and the network's robustness. These techniques, inspired by Eigen et al. [10], included rotation, scaling, color transformation, flipping, and small translations. By applying these data augmentation techniques, we could generate a more diverse set of training samples, which helped improve the network's generalization capability and robustness. These techniques contributed to the network's ability to estimate depth accurately in various real-world scenarios.

4.2. Evaluation Metrics

We tested and compared the outcomes of our method with various depth estimation methods on the NYU-Depth-v2, KITTI, and DDAD datasets. Some of the results are shown in Figures 8 and 9. A colored map is used to visualize the different depths. Blue indicates a shorter distance, and yellow indicates a longer distance. A well-recognized set of assessment methods proposed by Eigen et al. [10] was used, which included four assessment indicators: root mean square error (*RMSE*), *RMSE* (*log*), absolute relative difference (*ARD*) and squared relative difference (*SRD*). The evaluation metrics are calculated using the following:

$$RMSE(linear) = \sqrt{\frac{1}{|N|}\sum_{i=1}^{N} \|y_i - \hat{y}_i\|^2}, \tag{17}$$

$$RMSE(log) = \sqrt{\frac{1}{|N|}\sum_{i=1}^{N} \|logy_i - log\hat{y}_i\|^2}, \tag{18}$$

$$ARD = \sqrt{\frac{1}{|N|}\sum_{i=1}^{N} \frac{|y_i - \hat{y}_i|}{\hat{y}_i}}, \tag{19}$$

$$SRD = \sqrt{\frac{1}{|N|}\sum_{i=1}^{N} \frac{\|y_i - \hat{y}_i\|^2}{\hat{y}_i}}, \tag{20}$$

where the sum of the pixels is denoted by N. The ith pixel value of the predicted depth image is y_i. The ground truth depth image's ith pixel value is denoted by $\hat{y}_i$.

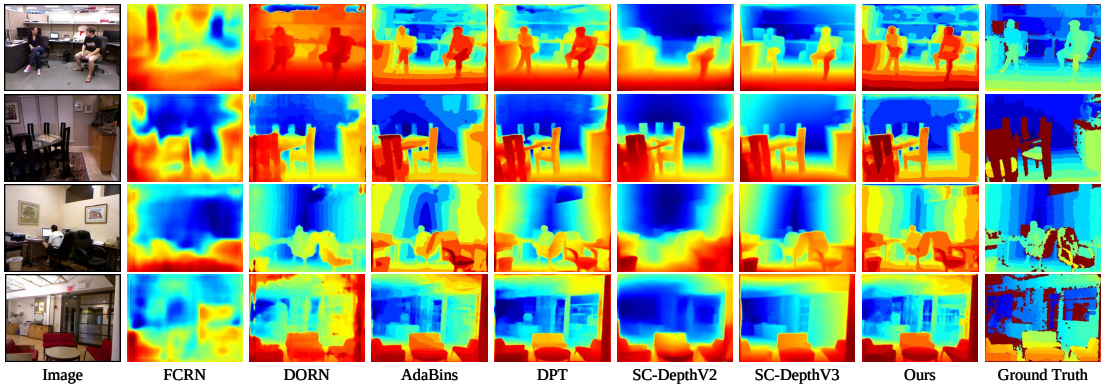


Figure 9. The visualization results of depth estimation on the NYU-Depth-v2 [46] dataset.

4.3. Evaluation Results

Our method is compared with previous state-of-the-art methods, including DORN [20], AdaBins [22], DPT [23], FCRN [11], SC-DepthV1 [40], SC-DepthV2 [16], and SC-DepthV3 [17]. The NYU-Depth-v2 dataset was used to evaluate the proposed method. The performance comparison of the NYU-Depth-V2 dataset is presented in Table 6. As shown in the table, the RMSE (linear), RMSE (log), ARD, and SRD improvements using the proposed method with the NYU-Depth-v2 dataset are more than 4%, 9%, 13%, and 16%, respectively, compared to the published best architecture DPT. The advanced performance of our proposed model on most evaluation measures is due to the architecture and loss function we propose. Additionally, the proposed model outperforms other advanced methods with fewer parameters, such as AdaBins and DPT. The proposed compact pseudo-depth estimation network, in combination with the transformer depth estimation network, successfully helps achieve the accurate and efficient estimation of depth maps. The visualization results are displayed in Figure 9. In comparison to previous methods, our method estimates the depth values of the example image given accurately and is more resistant to changing lighting circumstances.

Table 6. Objective metrics for depth estimation on the NYU-Depth-v2 [46] dataset. The best results are **bolded**.

Method	*Param (M)*	*RMSE (linear)*	*RMSE (log)*	*ARD*	*SRD*	GPU 3090(s)
FCRN	63.6	0.573	0.195	0.152	0.121	83.52
SC-DepthV2	40.9	0.554	0.186	0.142	0.112	37.48
DORN	110.3	0.509	0.172	0.115	0.082	925.22
SC-DepthV3	28.7	0.486	0.165	0.123	0.090	44.37
AdaBins	78	0.364	0.122	0.103	0.070	1377.6
DPT	123.1	0.357	0.121	0.110	0.077	81.85
Ours	**19.8**	**0.342**	**0.110**	**0.095**	**0.064**	**36.24**

The depth estimation results for KITTI are demonstrated in Table 7. The RMSE (linear), RMSE (log), and SRD improvements using the proposed method with the KITTI dataset are more than 8%, 7%, and 15%, respectively, compared to the DPT MDE method. Furthermore, Figure 10 provides the visualization results for the KITTI dataset. SC-DepthV1 and SC-DepthV3 present fuzzy depth maps at the object boundaries, while the methods of DORN, DPT, and AdaBins fail to preserve fine details, such as distant tiny objects. In contrast, our method has richer structural and object details and produces visually pleasing results.

Table 7. Objective metrics for depth estimation on the KITTI [47] dataset. The best results are **bolded**.

Method	*Param (M)*	*RMSE (linear)*	*RMSE (log)*	*ARD*	*SRD*	GPU 3090(s)
SC-DepthV1	27.9	4.997	0.196	0.118	0.870	38.39
SC-DepthV3	28.7	4.699	0.188	0.119	0.756	38.51
DORN	110.3	2.727	0.120	0.072	0.307	985.59
DPT	123.1	2.573	0.092	0.062	0.221	96.91
AdaBins	78	2.360	0.088	**0.058**	0.190	1504.35
Ours	**19.8**	**2.351**	**0.085**	**0.058**	**0.187**	**36.27**

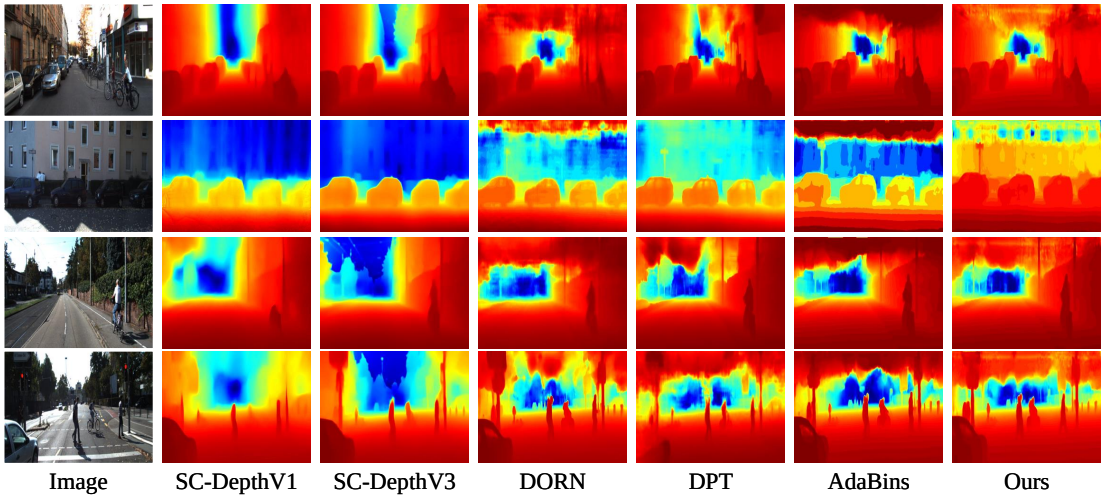


Figure 10. The visualization results of depth estimation on the KITTI [47] dataset.

Comparisons are made with SC-DepthV1 and SC-DepthV3 methods on DDAD. Table 8 lists the performance comparison results on the DDAD dataset. Compared with SC-DepthV1 and SC-DepthV3 methods, the proposed method outperforms all competitors in all metrics by a significant margin. Specifically, in the DDAD dataset, the proposed method improves the RMSE (linear), RMSE (log), ARD, and SRD by 14%, 12%, 17%, and 4%, respectively, compared to SC-DepthV3. A qualitative comparison is shown in Figure 11. The proposed method achieves more accurate and clearer depth estimation results. The cars in all four images in Figure 11 are better recognized by our method, and the reconstructed scenes are satisfactory with clear object boundaries and reasonable depth times, which can be better applied in the field of autonomous driving.

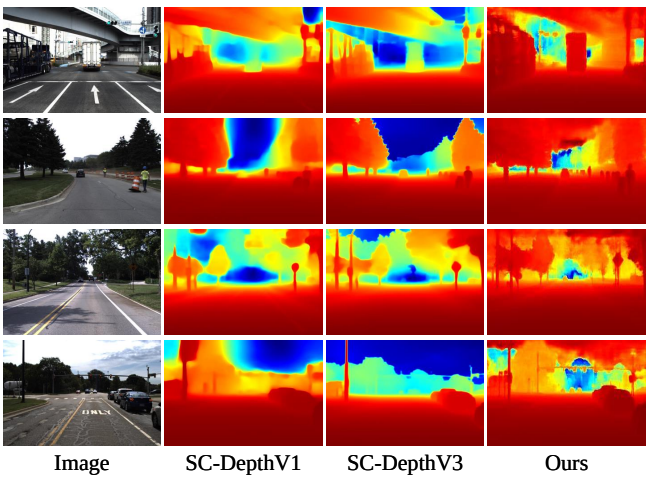


Figure 11. The visualization results of depth estimation on the DDAD [48] dataset.

Table 8. Objective metrics for depth estimation on the DDAD [48] dataset. The best results are **bolded**.

Method	*Param (M)*	*RMSE (linear)*	*RMSE (log)*	*ARD*	*SRD*	GPU 3090(s)
SC-DepthV1	27.9	16.118	0.279	0.168	3.825	88.04
SC-DepthV3	28.7	15.702	0.248	0.143	3.008	88.27
Ours	**19.8**	**13.427**	**0.218**	**0.118**	**2.861**	**82.18**

4.4. Ablation Study

To confirm the efficacy of the proposed method, comparison tests were carried out on the NYU-Depth-V2 dataset, i.e., Pseudo-Depth Net and Depth Net. A network architecture ablation study was conducted. Table 9 presents the results. The first row indicates the proposed method. The second row indicates the results obtained using the method without Pseudo-Depth Net. As shown in the table, the absence of Pseudo-Depth Net degraded the performance of the model. The second row represents the results obtained using the method without Depth Net. The performance of the network without Depth Net was significantly reduced. Table 9 demonstrates that joint Pseudo-Depth Net and Depth Net could greatly improve the accuracy of depth estimation.

Table 9. Ablation experiments for the proposed architecture on the NYU-Depth-v2 [46] dataset. The best results are **bolded**.

Method	*RMSE (linear)*	*RMSE (log)*	*ARD*	*SRD*
Ours	**0.342**	**0.110**	**0.095**	**0.064**
Ours w/o Pseudo-Depth Net	0.378	0.129	0.194	0.095
Ours w/o Depth Net	0.461	0.142	0.253	0.142

4.5. Running Time

In this subsection, we show the results of testing the runtime on NYU-Depth-v2, KITTI, and DDAD datasets. For the NYU-Depth-v2 dataset, the experiments were performed on 654 images of size 304×228. For the KITTI dataset, the experiments were performed on 697 images of size 228×912. For the DDAD dataset, the experiments were performed on 3950 images of size 640×384. The test time results are shown in Tables 6–8. All CNN-based methods were implemented using PyTorch on GPU (GPU was NVIDIA RTX 3090), belonging to NVIDIA of CA, USA, USA. Among all the methods, the proposed method was faster than all the competing methods, which shows that real-time depth estimation is possible.

5. Discussion

In the final section, several aspects of the computational efficiency of our model and its portability to current network architectures are discussed. The transferability of deep networks has garnered increasing interest in recent times. Currently, many methods achieve satisfactory results by training and testing on the same dataset. However, when confronted with diverse datasets from various fields or captured using different cameras, the performance often suffers a significant decline. To tackle this challenge, researchers have turned their focus toward improving the portability of depth networks. One approach involves incorporating camera parameters into the depth estimation framework. By considering specific camera information, like intrinsic and extrinsic parameters, the network can better adapt to different camera setups, resulting in more accurate depth estimation. Furthermore, employing domain adaptation techniques during the training process has emerged as a promising research direction. Domain adaptation aims to bridge the disparity between different datasets by aligning their feature distributions. This enables the network to generalize better and enhance performance on unseen datasets. Through the application of

domain adaptation methods, the network learns to extract more robust and transferable features, leading to improved portability across diverse datasets or camera configurations.

Indeed, while deep networks have demonstrated impressive performance, their high computational requirements pose a significant challenge for practical application, particularly in real-time scenarios. The ability to achieve real-time performance with deep estimation networks holds great importance in their practical usability. Lightweight networks are characterized by their reduced number of parameters, which can impact their overall performance. Thus, a significant research focus lies in improving the accuracy of these networks while ensuring real-time performance. Striking a balance between accuracy and efficiency is a topic that warrants further exploration. Furthermore, questions such as the depth cues learned by deep networks and the specific cues utilized in the estimation process have received little attention in the literature. Investigating these aspects can deepen our understanding of the inner workings of deep networks for depth estimation tasks. To summarize, the development of lightweight networks for real-time performance and further research on the mechanisms underlying monocular depth estimation using deep learning are both worthwhile directions. These areas of study hold promise in advancing the practical application and understanding of deep estimation networks.

6. Conclusions

A deep joint network, which consists of a pseudo-depth and a depth network, is proporsed to provide meaningful coarse and fine features to predict high-quality depth images from a single RGB image. The pseudo-depth network utilizes upsampling mapping, residual modules, and a modified codec to obtain high-resolution depth maps directly. The depth network employs an effective global transformer strategy and Unet depth network to improve the performance, which greatly improves the estimation accuracy. In addition, multiple losses are used jointly to improve the training of the network, and the proposed method attains advanced performance on KITTI and NYU-Depth-v2 datasets.

Author Contributions: Methodology, J.T. and M.G.; Software, X.G.; Formal analysis, T.D.; Writing—original draft, J.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the Open Research Fund of State Key Laboratory of Transient Optics and Photonics, Chinese Academy of Sciences, under Grant SKLST202214 and Grant SKLST202005, in part by the Key R&D project of Shaanxi Province under Grant 2022ZDLGY01-03, in part by Key Scientific Research Program of Shaanxi Provincial Department of Education under Grant 23JY063, and in part by Xian Science and Technology Research Plan under Grant 22GXFW0088.

Data Availability Statement: Data are contained within the article.

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

References

1. Siddiqui, Y.; Porzi, L.; Bulò, S.; Muller, N.; Nießner, M.; Dai, A.; Kontschieder, P. Panoptic lifting for 3d scene understanding with neural fields. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Vancouver, Canada, 18–22 June 2023 ; pp. 9043–9052.
2. Ali, S.; Pandey, A. ArthroNet: A monocular depth estimation technique with 3D segmented maps for knee arthroscopy. *Intell. Med.* **2023**, *3*, 129–138. [CrossRef]
3. Yang, B.; Xu, X.; Ren, J.; Cheng, L.; Guo, L.; Zhang, Z. SAM-Net: Semantic probabilistic and attention mechanisms of dynamic objects for self-supervised depth and camera pose estimation in visual odometry applications. *Pattern Recognit. Lett.* **2022**, *153*, 126–135. [CrossRef]
4. Zhou, C.; Yan, Q.; Shi, Y.; Sun, L. DoubleStar: Long-Range Attack Towards Depth Estimation based Obstacle Avoidance in Autonomous Systems. In Proceedings of the 31st USENIX Security Symposium (USENIX Security 22), Boston, MA, USA, 10–12 August 2022; pp. 1885–1902.
5. Tesla Use pEr-Pixel Depth Estimation with Self-Supervised Learning. Available online: https://youtu.be/hx7BXih7zx8?t=1334 (accessed on 21 April 2020).
6. Tesla AI Day. Available online: https://youtu.be/j0z4FweCy4M?t=5295 (accessed on 20 August 2021).
7. Zheng, X.; Sun, H.; Lu, X.; Xie, W. Rotation-Invariant Attention Network for Hyperspectral Image Classification. *IEEE Trans. Image Process.* **2022**, *31*, 4251–4265. [CrossRef]

8. Zheng, X.; Gong, T.; Li, X.; Lu, X. Generalized Scene Classification from Small-Scale Datasets with Multi-Task Learning. *IEEE Trans. Geosci. Remote Sens.* **2021**, *60*, 1–11.
9. Saxena, A.; Sun, M.; Ng, A.Y. Make3D: Learning 3D scene structure from a single still image. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 824–840. [CrossRef]
10. Eigen, D.; Puhrsch, C.; Fergus, R. Depth map prediction from a single image using a multiscale deep network. In Proceedings of the NeurIPS, Montreal, Canada, 8–13 December 2014; pp. 2366–2374.
11. Laina, I.; Rupprecht, C.; Belagiannis, V.; Tombari, F.; Navab, N. Deeper Depth Prediction with Fully Convolutional Residual Networks. In Proceedings of the 2016 Fourth International Conference on 3D Vision (3DV), Stanford, CA, USA, 25–28 October 2016; pp. 239–248.
12. Hu, J.; Fan, C.; Jiang, H.; Guo, X.; Gao, Y.; Lu, X.; Lam, T. Boosting lightweight depth estimation via knowledge distillation. In Proceedings of the International Conference on Knowledge Science, Engineering and Management, Guangzhou, China, 15–18 August 2023; pp. 27–39.
13. Lopez-Rodriguez, A.; Mikolajczyk, K. Desc: Domain adaptation for depth estimation via semantic consistency. *Int. J. Comput. Vis.* **2023**, *131*, 752–771. [CrossRef]
14. Agarwal, A.; Arora, C. Attention attention everywhere: Monocular depth prediction with skip attention. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, Waikoloa, HI, USA, 2–7 January 2023; pp. 5861–5870.
15. Yin, Z.; Shi, J. GeoNet: Unsupervised learning of dense depth, optical flow and camera pose. In Proceedings of the IEEE/CVF Computer Vision and Pattern Recognition Conference, Salt Lake City, UT, USA, 18–22 June 2018; pp. 1983–1992.
16. Bian, J.; Zhan, H.; Wang, N.; Chin, T.; Shen, C.; Ian, R. Auto-Rectify Network for Unsupervised Indoor Depth Estimation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 9802–9813. [CrossRef] [PubMed]
17. Sun, L.; Bian, J.; Zhan, H.; Yin, W.; Reid, I.; Shen, C. SC-DepthV3: Robust Self-supervised Monocular Depth Estimation for Dynamic Scenes. *arXiv* **2022**, arXiv:2211.03660.
18. Masoumian, A.; Rashwan, H.; Abdulwahab, S.; Cristiano, J.; Puig, D. Gcndepth: Self-supervised monocular depth estimation based on graph convolutional network. *Neurocomputing* **2023**, *517*, 81–92. [CrossRef]
19. Hoyer, L.; Dai, D.; Wang, Q.; Chen, Y.; Gool, L. Improving semi-supervised and domain-adaptive semantic segmentation with self-supervised depth estimation. *Int. J. Comput. Vis.* **2023**, *131*, 2070–2096. [CrossRef]
20. Fu, H.; Gong, M.; Wang, C.; Batmanghelich, K.; Tao, D. Deep Ordinal Regression Network for Monocular Depth Estimation. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 2002–2011.
21. Lee, J.H.; Han, M.-K.; Ko, D.W.; Suh, I.H. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv* **2019**, arXiv:1907.10326.
22. Bhat, S.F.; Alhashim, I.; Wonka, P. AdaBins: Depth Estimation Using Adaptive Bins. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Nashville, TN, USA, 19–25 June 2021; pp. 4008–4017.
23. Ranftl, R.; Bochkovskiy, A.; Koltun, V. Vision Transformers for Dense Prediction. In Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, QC, Canada, 10–17 October 2021; pp. 12159–12168.
24. Saxena, A.; Chung, S.H.; Ng, A.Y. Learning Depth from Single Monocular Images. In Proceedings of NeurIPS, Vancouver, BC, Canada, 5–8 December 2005; pp. 1161–1168.
25. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.H.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An image is worth 16x16 words: Transformers for image recognition at scale. In Proceedings of ICLR, Vienna, Austria, 4–9 May 2021; pp. 1–22.
26. Yang, G.; Tang, H.; Ding, M.; Sebe, N.; Ricci, E. Transformers solve the limited receptive field for monocular depth prediction. *arXiv* **2021**, arXiv:2103.12091.
27. Liu, B.; Gould, S.; Koller, D. Single image depth estimation from predicted semantic labels. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Long Beach, USA, 16–20 June 2010; pp. 1253–1260.
28. Karsch, K.; Liu, C.; Kang, S.B. Depth transfer: Depth extraction from video using non-parametric sampling. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *36*, 2144–2158. [CrossRef] [PubMed]
29. Liu, M.; Salzmann, M.; He, X. Discrete-continuous depth estimation from a single image. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 24–27 June 2014; pp. 716–723.
30. Spencer, J.; Qian, C.; Russell, C.; Hadfield, S.; Graf, E.; Adams, W.; Schofield, A.; Elder, J.; Bowden, R.; Cong, H.; et al. The monocular depth estimation challenge. In Proceedings of the IEEE/CVF Winter Conference Applications of Computer Vision, Waikoloa, Hawaii, USA, 3–7 January 2023; pp. 623–632.
31. Eigen, D.; Fergus, R. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In Proceedings of the ICCV, Santiago, Chile, 11–18 December 2015; pp. 2650–2658.
32. Liu, F.; Shen, C.; Lin, G.; Reid, I. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *38*, 2024–2039. [CrossRef] [PubMed]
33. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the CVPR, Las Vegas, USA, 26–30 June 2016; pp. 770–778.
34. Cao, Y.; Wu, Z.; Shen, C. Estimating depth from monocular images as classfication using deep fully convolutional residual networks. *IEEE Trans. Circuits Syst. Video Technol.* **2017**, *28*, 3174–3182. [CrossRef]

35. Li, B.; Dai, Y.; He, M. Monocular depth estimation with hierarchical fusion of dilated CNNs and soft-weighted-sum inference. *Pattern Recognit.* **2018**, *83*, 328–339. [CrossRef]
36. Hu, J.; Ozay, M.; Zhang, Y.; Okatani, T. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. In Proceedings of the WACV, Hawaii, USA, 7–11 January 2019; pp. 1043–1051.
37. Chen, X.; Chen, X.; Zha, Z.-J. Structure-aware residual pyramid network for monocular depth estimation. *arXiv* **2019**, arXiv:1907.06023.
38. Ye, X.; Chen, S.; Xu, R. DPNet: Detail-preserving network for high quality monocular depth estimation. *Pattern Recognit.* **2021**, *109*, 107578. [CrossRef]
39. Godard, C.; Aodha, O.M.; Brostow, G.J. Unsupervised monocular depth estimation with left-right consistency. In Proceedings of the IEEE Conference Computer Vision and Pattern Recognition (CVPR), Hawaii, USA, 21–26 July 2017; pp. 270–279.
40. Bian, J.; Zhan, H.; Wang, N.; Li, Z.; Zhang, L.; Shen, C.; Cheng, M.; Reid, I. Unsupervised Scale-consistent Depth Learning from Video. In Proceedings of Advances in Neural Information Processing Systems (NeurIPS), Vancouver, Canada, 8–14 December 2019; pp. 1–16.
41. Bian, J.; Li, Z.; Wang, N.; Zhan, H.; Shen, C.; Cheng, M.M.; Reid, I. Unsupervised scale-consistent depth and ego-motion learning from monocular video. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, Canada, 8–14 December 2019; Volume 32, pp. 35–45.
42. Klingner, M.; Termöhlen, J.-A.; Mikolajczyk, J.; Fingscheidt, T. Self-supervised monocular depth estimation: Solving the dynamic object problem by semantic guidance. In Proceedings of the European Conference on Computer Vision; Springer: Glasgow, UK, 23–28 August 2020; pp. 582–600.
43. Heise, P.; Klose, S.; Jensen, B.; Knoll, A. PM-Huber: PatchMatch with Huber Regularization for Stereo Matching. In Proceedings of the IEEE International Conference on Computer Vision, Sydney, NSW, Australia, 1–8 December 2013; pp. 2360–2367.
44. Wang, Z.; Bovik, A.C.; Sheikh, H.R.; Simoncelli, E.P. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Trans. Image Process.* **2004**, *13*, 600–612. [CrossRef]
45. Xian, K.; Zhang, J.; Wang, O.; Mai, L.; Lin, Z.; Cao, Z. Structure-guided ranking loss for single image depth prediction. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, DC, USA, 13–19 June 2020; pp. 611–620.
46. Silberman, N.; Hoiem, D.; Kohli, P.; Fergus, R. Indoor segmentation and support inference from rgbd images. In Proceedings of the European Conference on Computer Vision Workshops (ECCVW), Florence, Italy, Berlin, Germany, 7–13 October 2012; pp. 746–760.
47. Geiger, A.; Lenz, P.; Stiller, C.; Urtasun, R. Vision meets Robotics: The kitti dataset. *Int. J. Robot. Res. (IJRR)* **2013**, *32*, 1231–1237. [CrossRef]
48. Guizilini, V.; Ambrus, R.; Pillai, S.; Raventos, A.; Gaidon, A. 3d packing for self-supervised monocular depth estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, DC, USA, 13–19 June 2020; pp. 2485–2494.
49. Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Li, F.F. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Miami Beach, FL, USA, 20–26 June 2009; pp. 248–255.
50. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. *arXiv* **2017**, arXiv:1711.05101.