BRNet: Exploring Comprehensive Features for Monocular Depth Estimation

Wencheng Han[1], Junbo Yin[2], Xiaogang Jin[3], Xiangdong Dai[4], and Jianbing Shen[1⋆],

[1] SKL-IOTSC, Computer and Information Science, University of Macau
[2] School of Computer Science, Beijing Institute of Technology
[3] State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China
[4] Guangdong OPPO Mobile Telecommunications Corp., Ltd

Abstract. Self-supervised monocular depth estimation has achieved encouraging performance recently. A consensus is that high-resolution inputs often yield better results. However, we find that the performance gap between high and low resolutions in this task mainly lies in the inappropriate feature representation of the widely used U-Net backbone rather than the information difference. In this paper, we address the comprehensive feature representation problem for self-supervised depth estimation by paying attention to both local and global feature representation. Specifically, we first provide an in-depth analysis of the influence of different input resolutions and find out that the receptive fields play a more crucial role than the information disparity between inputs. To this end, we propose a bilateral depth encoder that can fully exploit detailed and global information. It benefits from more broad receptive fields and thus achieves substantial improvements. Furthermore, we propose a residual decoder to facilitate depth regression as well as save computations by focusing on the information difference between different layers. We named our new depth estimation model Bilateral Residual Depth Network (BRNet). Experimental results show that BRNet achieves new state-of-the-art performance on the KITTI benchmark with three types of self-supervision. Codes are available at: `https://github.com/wencheng256/BRNet`

1 Introduction

Depth estimation is a fundamental problem in many applications, which aims to estimate the depth for each pixel in a 2D image. Traditional methods formulate this task as a stereo matching problem [29,38], but the exhausted matching process often limits their deployment. In recent years, deep learning based monocular depth estimation [7] has drawn much attention, which predicts depth only relying on a single-view input image. However, such a method requires large-scale samples with accurate annotation for fully-supervised training, leading to expensive and elaborate manual work[28,5]. An alternative is to apply self-supervised

⋆ Corresponding author: *Jianbing Shen*, email: shenjianbingcg@email.com.

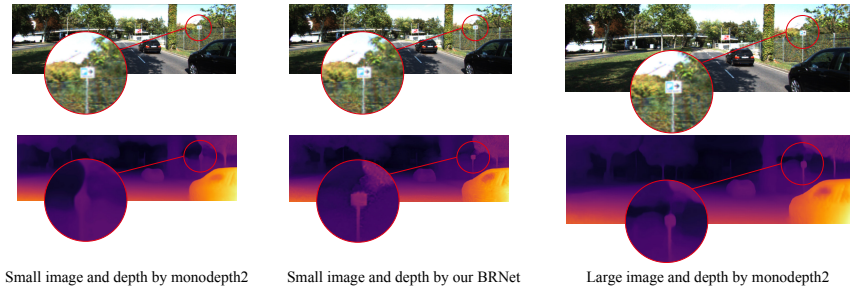


Fig. 1. Predictions from monodepth2 [8] and our BRNet with different input resolutions. The board in the images shows no obvious difference in the two resolutions, but the results of the monodepth2 differ substantially. This motivates us to explore the reason for the different predictions and devise a more suitable model of fully utilizing information in the images for depth estimation.

learning [7] for monocular depth estimation, which requires only video sequences or stereo images to provide supervision signal. This largely decreases the heavy annotation burden and still achieves competitive performance.

Most of the self-supervised learning works, e.g., Monodepth2 [8], HRDepth [17] and PackNet-SfM [9], adopt the U-Net-like [26] architectures. U-Net [26] contains an encoder for extracting hierarchical feature maps and a decoder for predicting depth based on encoded feature maps. It is worth mentioning that almost all these models achieve notably better results when taking higher resolution images as input. Despite this, little work has given deeper attention to how the input resolution influences the performance. Intuitive thought is that the more detailed information in larger images may be the reason. However, as shown in Fig. 1, the boards in the red circles are equally clear in both resolutions, but the model still gives different predictions for them. This observation motivates us to find out the real mechanism behind it.

In this paper, we argue that the performance gap between the high and low-resolution inputs mainly lies in the different receptive fields rather than the information disparity between the inputs. Features extracted from high-resolution images have smaller receptive fields than those extracted from low-resolution ones. The smaller receptive enforces the network to focus more on the detailed information of the images, which is of crucial importance for a depth estimation model. To prove this point, we conduct a heuristic experiment by interpolating the images from a lower resolution to a higher resolution, e.g., the large fake image in Fig. 2, and perform depth estimation based on this fake large image. It turns out that there is little performance decrease in comparison to the input with real large images, suggesting the information disparity is not the main reason for the performance gap. In addition, we find that only smaller receptive fields could not always come with better results, so we should also integrate the feature with large receptive fields, which provides global information such as perspective and object relations [20,7]. To this end, we propose a bilateral

encoder. One branch is designed to encode detailed information with small receptive fields, and the other accounts for global information with large receptive fields. Combining these comprehensive and complementary features, our model significantly outperforms existing U-Net-based encoders [26].

Since the input resolution determines the output resolution, we further investigate how much the output depth map sizes of the decoder affect the performance. There are five decoder layers in U-Net, where each layer upsamples the current feature map to a larger one and fuses it with the corresponding feature map from the encoder. We evaluate the output depth from each decoder layer and surprisingly find that there is no obvious performance gap between them. This is contrary to the intuition that larger outputs should encode more details and be preciser than smaller ones. We infer that the incremental information can not be fully exploited as the decoder layers go deeper. To address this, we propose to add the depth obtained from the previous layers to the input of the current layer and enforce the network to predict residual information between different depth sizes. We call the proposed decoder as residual decoder, which is inspired by the residual operation in ResNet. Our residual decoder effectively focuses on the difference between features maps from different layers and thus facilitates depth regression.

Our main contributions can be summarized as follows:

- We provide a deeper insight into the performance gaps between different input resolutions by thorough and exhaustive heuristic experiments. We find that appropriate receptive fields are critical to the performance of the depth estimation model.
- We propose a new Bilateral Depth Encoder that can fully exploit details and global information simultaneously, providing a more comprehensive feature representation for depth estimation.
- A Residual Decoder is introduced for dense depth regression. It makes full use of the output depth from each decoder layer, focusing on the information difference between features maps of different scales.
- Our depth estimator achieves a new state-of-the-art performance on the KITTI benchmark. The approach remarkably outperforms other approaches with all the types of self-supervision, e.g., achieving 0.097 and 4.378 in terms of Abs Rel and RMSE, respectively, with MS training.

2 Related Work

Current monocular depth estimation approaches can be roughly categorized into two groups. One group applies the fully-supervised learning to regress the ground truth depth maps generated from the LiDAR sensor. Another group provides supervision by leveraging synchronized monocular videos or stereo pairs in order to optimize the depth estimation model in a self-supervised fashion.

2.1 Fully-Supervised monocular depth estimation

The first method for monocular depth estimation based on deep learning is developed by Eigen *et al.* [5], which directly regresses the depth by two components. One is for estimating the global structure of the scene, and the other is used to refine it using local information. The two components are trained separately, and this increases the training expenses. Later this architecture is replaced by a fully convolutional network [14] designed for semantic segmentation by Evan *et al.* [30]. This work enables the depth estimation task to be trained end-to-end with a deeper network, and it performs comparably with the depth sensor. And then, many methods are exploited to improve the performance [2,31,25].

Besides point clouds, some networks also try to exploit extra labels to improve the performance. Klodt *et al.* [12] employ sparse depths and poses from the traditional SLAM system, and Ramirez *et al.* [24] demonstrate that jointly predicting depth and semantic labels can improve the performance of depth estimation. Tosi *et al.* [32] exploit proxy ground truth labels, which are generated by a traditional stereo matching method.

Although these works achieve notable success, they rely on ground truth labels that are expensive to obtain, which limits the training data's scale. In consequence, many works focus on self-supervised methods.

2.2 Self-supervised monocular depth estimation

For self-supervised approaches in monocular depth estimation, there are mainly two categories. One is based on stereo pairs, and the other uses consecutive video frames during the training phase.

Stereo training Garg *et al.* [6] first propose the self-supervised training approach for monocular depth estimation with a photometric-consistency loss between stereo pairs. Specifically, they take the images from one of the views and construct pseudo images for the other view based on the depth predicted and the relative position between the two views. Then a L_2 loss is employed to measure the difference between generated images and the real ones. Godard *et al.* [7] improve the method by replacing the L_2 loss with a L_1 loss and introducing the structural similarity index measure to generate sharper results. Godard *et al.* [8] show that computing projection loss at a higher resolution will improve the depth map quality, and Pillai *et al.* [21] introduce differentiable flip augmentation and subpixel convolutions for increasing the fidelity of the depth map. Watson *et al.* [35] introduce Depth Hints to alleviate the effects of ambiguous reprojections in depth-prediction.

Although some works have achieved satisfying performance [13,16,1,39,22], stereo image pairs require specialized equipment. Therefore, some other works like [10,37,36,18,34] turn to investigate methods based on video sequences.

Video training Zhou *et al.* [41] design an additional network to predict the camera pose between adjacent frames and construct the pseudo frame based on the previous and subsequent frames, respectively. To deal with non-rigid scene motion, they employ a motion explanation mask so that it allows the network to

Experiments	Metrics		
	Abs Rel ↓	RMSE ↓	$\delta < 1.25$ ↑
Input Resolution			
Small	0.115	4.863	0.877
Large (1024 × 320)	0.115	4.701	0.879
Fake Large (1024 × 320)	0.115	4.708	0.880
Encoder			
1/2 Receptive field	0.111	4.672	0.880
1/4 Receptive field	0.116	4.761	0.875
1/8 Receptive field	0.129	5.098	0.844
Decoder			
Extra Layer	0.118	4.826	0.869
Disp0 (full size)	0.115	4.863	0.877
Disp1 (1/2 size)	0.114	4.858	0.877
Disp2 (1/4 size)	0.114	4.834	0.875
Disp3 (1/8 size)	0.116	4.869	0.868

Table 1. Experiments about different input resolutions and different architecture of encoder and decoder. We adopt Abs Rel, RMSE and δ as our metrics. For Abs Rel and RMSE, lower values are better, and higher values are better for $\delta > 1.25$.

ignore specific regions. Godard *et al.* [8] then propose a new strategy to replace this mask and achieve better performance. To fully exploit the view difference between frames, they did not average the loss from the future and past frames but took the minimum value of the loss instead. They also propose a simple auto-masking method to filter out pixels that do not change appearance in a sequence from one frame to the next. Lyu *et al.* [17] prove that predicting more accurate boundaries can improve performance and redesign the skip connection generating high-resolution feature maps to get sharper edges. Zhou *et al.* [42] proposed a lightweight architecture that performs better in a more efficient way. Johnston *et al.* [11] and Bakhanova *et al.* [19] introduce attention mechanism into depth estimation area to improve the performance of models.

These methods show better performances when taking large inputs in their experiments, but little work investigates the reason.

3 Methods

This section first analyses the reason behind the performance gap between the different input resolutions and reveals several limitations of the prevalent depth estimation backbone U-Net [26]. Then, based on our observations, we develop a new depth estimation network, BRNet, which effectively mines the detail and global information of the input image by the Bilateral Encoder and fully exploits the incremental data of different layers by the Residual Decoder. Notably, even taking small inputs, our model significantly outperforms the baseline method taking large inputs.

3.1 Analysis of Current Depth Estimator

As shown in previous works, models with larger inputs always outperform those with small inputs, which is a consensus of the community. An intuitive reason is that larger images keep more real-world information while small ones can not. However, as shown in the Fig. 1, although some objects can be seen clearly in both resolutions, the model still gives different results. Stemming from these observations, we believe that information difference between inputs may not be the main reason leading to the performance gap.

To explore this, we conduct an experiment as shown in Fig. 2. We first down-sample a large input image into a small resolution and then interpolate it back to the original size. The resultant image, called a fake-large image, has the same information as the small image except for a larger size. Afterward, we apply monodepth2 [8] as the baseline model to train and evaluate on fake-large, small and large images, respectively. If the performance gap comes from the information difference, the model's performance with fake-large inputs will be the same as that of small inputs. Otherwise, the result will be similar to the one by taking large inputs. As shown in the Table 1, the model's performance with fake-large inputs is comparable with that of large inputs and is better than the model of small inputs. This experiment proves that most of the performance gap between large and small inputs in the depth-estimation model does not come from the information difference between them.

As input information is not the reason, it may be in the process of feature extraction in the encoder or result generation in the decoder. So we design a series of experiments to find it.

Encoder We find that the bad cases from small inputs are mainly objects that are far from the camera. The prediction for these objects heavily depends on the detailed occlusion relationship with other elements. However, the radical down-sampling strategy of the encoder provides a large receptive field, making it hard to focus on these details and thus neglecting this information. Larger inputs counterbalance the down-sampling and provide relatively smaller receptive fields, which can extract more information, leading to the performance gap between different resolutions. To validate this, we adjust the stride of the convolutions in the encoder and feed small input into it so that the produced feature maps have $\frac{1}{2}$ receptive fields of the origin network and are similar to those of large input. As shown in Table 1, it generates even better results than that of large inputs. The results support our theory that the receptive fields of the model may be the main reason of the performance gap.

On the other hand, as mentioned in Miangole *et al.* [20], higher resolution inputs do not always come with better results. As the resolution gets higher, the network starts losing the overall structure of the scene. As a result, the relative depth between objects is better predicted, but their absolute depth in the whole image is less precise than that of smaller inputs. To verify this conclusion in monocular depth estimation, we prepare another experiment by adjusting more convolutions in the decoder and trying smaller receptive fields. As shown in the

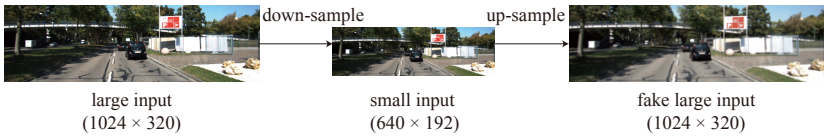


Fig. 2. Experiments in our analysis. An example of our larger inputs, small inputs and fake-large inputs.

Table 1, when reducing the receptive fields into $\frac{1}{4}$ and $\frac{1}{8}$ of the original ones, the performance of this model will degenerate gradually.

In summary, relatively smaller receptive fields can help provide more detailed information, contributing to the better performance of the depth estimator. Larger inputs also lead to smaller receptive fields, which may be the main reason behind the influence of the input resolution. Before we make a conclusion, we still need to clarify the role of the decoder.

Decoder Larger inputs lead to smaller receptive fields, but they also induce larger depth maps. Here is another question, how much does the result size affect the performance. To figure out this question, we prepare another experiment. In this experiment, we feed the network large inputs but down-sample the first feature map into small resolution and send it to the rest of the encoder. The feature maps of the encoder keep the same receptive fields as it takes small inputs. As for the decoder, we insert an extra decoder layer, which upsamples the decoder feature map and fuses it with the large feature map, and we finally obtain the results with the same size as larger inputs as shown in the Fig. 2. As a result, it has similar results with smaller inputs which confirms that the larger output size can obviously improve the performance.

Since a larger result resolution does not affect the performance markedly, does a smaller result affect the performance a lot? To form multi-scale disparity loss, monodepth2 [8] modifies the decoder to generate four results with different sizes. So we evaluate the results produced by different layers of the decoder, which has a full, $\frac{1}{2}$, $\frac{1}{4}$, and $\frac{1}{8}$ resolution, respectively. As shown in the Table 1, they all perform comparably. The result with the full size is even worse than that with $\frac{1}{4}$ size. This observation is counterintuitive because larger depth maps are supposed to keep more details, so they should be preciser than small results. The experiment shows that the decoder does not perform as it is designed. We attribute this problem to that the U-Net decoder designed for semantic segmentation is not well suited to depth estimation and will discuss it in Sec. 3.3.

Summaries. The analysis above leads us to the following conclusion: The receptive field's difference contributes most to the performance gap between different input resolutions rather than the information disparity. Therefore, adjusting the convolutions' stride in the encoder providing more detailed information can lead to better performance. Besides, global information is also indispensable. To achieve better performance, we must fully exploit both detailed and global features. Finally, we find that U-Net decoder can not achieve this as it is designed

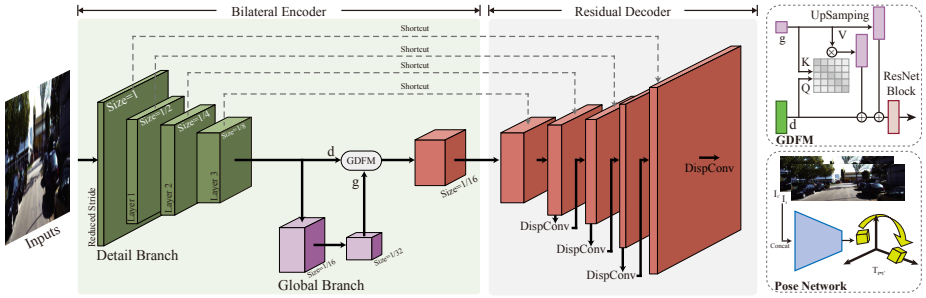


Fig. 3. Overview of the proposed network which mainly contains two parts. One is the bilateral encoder the other is the residual decoder. There is a *detail branch* for extracting detail features for the encoder, a *global branch* to extract high-level information, and finally, a global-detail fuse module (GDFM) that fuses the two feature maps. The pose network is employed to give the relative pose between two frames and is only used during the training phase.

in the depth estimation, so we need a new decoder that is more suitable for this task.

3.2 Bilateral Depth Encoder

According to the conclusions drawn above, a better depth encoder should properly deal with global and detailed information. However, there is a paradox. Detailed information needs the encoder to have relatively smaller receptive fields to focus on occlusion relationships. On the other hand, only larger receptive fields can provide an objective overview. To meet their needs simultaneously, we design a two-branch architecture to extract both global and detailed features and then fuse them into more comprehensive ones.

As shown in the Fig. 3, we reduce the stride of the first convolution so that the whole network downsamples the input more slowly, concentrating more on the details. After several blocks, information such as occlusion relationships will be well encoded. Still, as the receptive fields are reduced, they cannot get enough global information. Thus it will lead to inconsistent predictions, as discussed above. To alleviate this problem, we develop a new branch, i.e., *Global Branch*, after Layer-3. For convenience, we call the original one as *Detail Branch*.

The global branch further down-samples the feature maps to get larger receptive fields while the detail-branch keeps high resolution of the feature maps maintaining detail features. Then the two branches are joined together by GDFM (Global-Detail Fusion Module) to generate comprehensive feature maps, as shown in Fig. 3. Some global information is critical to every location in the image, such as the perspective point and view point. To share this information to every place, we employ a multi-head attention [33] to fuse global and detailed information. Specifically, the detailed features are mapped into query features, and the global features are mapped into key and value features. Then

the query and key features are multiplied to generate an affinity map, and the affinity map is used to aggregate information from value features. In this way, every pixel in the detail feature map can aggregate corresponding global features as they need, even though they are far away from current location. Finally, we add this feature map into a detailed feature map to fuse them. Also, we add a residual shortcut of the global features by upsampling and merging it into the final feature map to facilitate optimization.

Bilateral Depth Encoder can fully exploit hidden information from input images by taking advantage of global and detailed features. With these comprehensive feature maps, our model is able to achieve significant improvement.

3.3 Residual Decoder

Intuitively, larger outputs contain more details and are supposed to perform better than smaller ones. On the contrary, as discussed in Sec. 3.1, the results of different layers in the current depth decoder perform closely, which means that the decoder does not ideally perform as designed. Decoders in most depth estimators are adopted from U-Net [26], which is intended for semantic segmentation. However, semantic segmentation is a dense classification task, while depth estimation is a dense regression task so they are different.

In the classification task, as types are discrete, every pixel needs to be judged independently to generate sharp borders. But in depth estimation, depth values are changed smoothly, and every pixel in the depth map is close to its' neighbors. Thus, we can generate depth for large regions and fine-tune the depth for every pixel in a region.

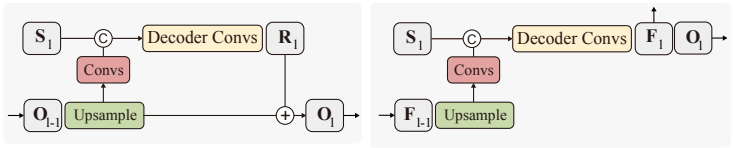
In this work, we regard the outputs of higher layers as residuals:

$$Disp(l) = UpSample(Disp(l-1)) + R(l). \tag{1}$$

If there is no extra information for some regions in the view, their residuals $R(l)$ are close to 0. On the contrary, if there is valuable information, the residuals are used to adjust the predictions made by the former layer. As shown in the Fig. 4, different from the original decoder layer, which upsamples the feature maps from the previous layer, our residual layer upsamples the one-channel depth map to a larger size. Then the depth map is modulated by a convolution, and then we concatenate it with the shortcut feature map extracting the residual using another convolution as:

$$R(l) = f(Disp(l-1), F(l); \theta), \tag{2}$$

where θ is the parameters and $F(l)$ is the shortcut features of layer l. Finally, the residual is added to the previous depth map. Compared to independent decoder layers, residuals can focus on the differences between layers, which helps the current layer to generate better predictions than their predecessors. Benefited from the multi-scale loss [8], every layer in the decoder can be fully optimized during the training phase. Finally, with this residual decoder, our model yields significant improvements compared to the U-Net decoder.



(a) Residual Decoder Layer (b) U-Net Decoder Layer

Fig. 4. Comparison between Residual Decoder layer and U-Net Decoder layer. In a U-Net decoder layer, the feature map from the previous layer F_{l-1} is upsampled and concatenated with the shortcut feature map S_l. Then the feature map is used to generate fused featuremaps F_l and a depth map O_l. In a residual decoder layer, the depth map from the previous layer (O_{l-1}) is upsampled and concatenated with S_l. And the output of this layer is regarded as residual (R_l) and should be added to the previous depth map generating O_l.

3.4 Loss Function

To optimize the proposed model, we formulate our problem as minimising the photometric reprojection error during the training phase. As described in monodepth2 [8], at each pixel, minimization is used to merge results from different sources:

$$L_p = \min_{t'} pe\left(I_t, I_{t'\to t}\right), \tag{3}$$

where I_t is the target image and $I_{t'\to t}$ is the pseudo target image which is generated by:

$$I_{t'\to t} = I_{t'}\left\langle \text{proj}\left(D_t, T_{t\to t'}, K\right)\right\rangle \tag{4}$$

where $I_{t'}$ is the source image and D_t is the depth map generated by our model. K is the pre-computed intrinsics and $T_{t\to t'}$ is the camera pose between $I_{t'}$ and I_t. When training with video sequences, $T_{t\to t'}$ is the result of the pose model or when training with stereo image pairs, it is the pose between two cameras. $pe\left(I_a, I_b\right)$ is our photometric error function, which is designed as:

$$\frac{\alpha}{2}\left(1 - \text{SSIM}\left(I_a, I_b\right)\right) + (1-\alpha)\left\|I_a - I_b\right\|_1. \tag{5}$$

In our experiments, α is set to 0.85. Also, the edge-aware smoothness [7] is employed to encourage the disparities to be locally smooth:

$$L_s = \left|\partial_x d_t^*\right| e^{-\left|\partial_x I_t\right|} + \left|\partial_y d_t^*\right| e^{-\left|\partial_y I_t\right|}. \tag{6}$$

Then the final loss is:

$$L = \mu L_p + \lambda L_s, \tag{7}$$

where μ and λ are the auto-mask introduced by [8]. When training with stereo image pairs, $I_{t'}$ is the image from the other view. For video sequences, $I_{t'} \in \{I_{t-1}, I_{t+1}\}$ where I_{t-1} is the previous frame and I_{t+1} is the subsequent frame.

Method	Resolution	Trian	lower is better				higher is better		
			Abs Rel	Sq Rel	RMSE	RMSE log	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
low resolution									
EPC++ [15]	640×192	M	0.141	1.029	5.350	0.216	0.816	0.941	0.976
Struct2depth [3]	416×128	M	0.141	1.026	5.291	0.215	0.816	0.945	0.979
Monodepth2 [8]	640×192	M	0.115	0.903	4.863	0.193	0.877	0.959	0.981
Monodepth2 R50 [8]	640×192	M	0.110	0.831	4.642	0.187	0.883	0.962	0.982
Johnston [11]	640×192	M	0.106	0.861	4.699	0.185	0.889	0.962	0.982
PackNet-SfM [9]	640×192	M	0.111	0.785	4.601	0.189	0.878	0.960	0.982
HR-Depth [17]	640×192	M	0.109	0.792	4.632	0.185	0.884	0.962	0.983
R-MSFM6 [42]	640×192	M	0.112	0.806	4.704	0.191	0.878	0.960	0.981
BRNet (Ours)	640×192	M	**0.105**	**0.698**	**4.462**	**0.179**	**0.890**	**0.965**	**0.984**
Monodepth R50 [7]	512×256	S	0.133	1.142	5.533	0.230	0.83	0.936	0.970
3Net (R50) [23]	512×256	S	0.129	0.996	5.281	0.223	0.831	0.939	0.974
3Net (VGG) [23]	512×256	S	0.119	1.201	5.888	0.208	0.844	0.941	0.978
Monodepth2 [8]	640×192	S	0.109	0.873	4.960	0.209	0.864	0.948	0.975
BRNet (Ours)	640×192	S	**0.103**	**0.792**	**4.716**	**0.197**	**0.876**	**0.954**	**0.978**
EPC++ [15]	640×192	MS	0.128	0.935	5.011	0.209	0.831	0.945	0.979
Monodepth2 [8]	640×192	MS	0.106	0.818	4.750	0.196	0.874	0.957	0.979
HR-Depth [17]	640×192	MS	0.107	0.785	4.612	0.185	**0.887**	**0.962**	0.982
R-MSFM6 [42]	640×192	MS	0.111	0.787	4.625	0.189	0.882	0.961	0.981
BRNet (Ours)	640×192	MS	**0.099**	**0.685**	**4.453**	**0.183**	0.885	**0.962**	**0.983**
high resolution									
Monodepth2 [8]	1024×320	M	0.115	0.882	4.701	0.190	0.879	0.961	0.982
Zhao *et al.* [40]	832×256	M	0.113	0.704	4.581	0.184	0.871	0.961	0.984
PackNet-SfM [9]	1280×384	M	0.107	0.802	4.538	0.186	0.889	0.962	0.981
HR-Depth [17]	1024×320	M	0.106	0.755	4.472	0.181	**0.892**	**0.966**	0.984
R-MSFM6 [42]	1024×320	M	0.108	0.748	4.470	0.185	0.889	0.963	0.982
BRNet (Ours)	1024×320	M	**0.103**	**0.684**	**4.385**	**0.175**	0.889	0.965	**0.985**
SuperDepth + pp [21]	1024×382	S	0.112	0.875	4.958	0.207	0.852	0.947	0.977
Monodepth2 [8]	1024×320	S	0.107	0.849	4.764	0.201	0.874	0.953	0.977
BRNet (Ours)	1024×320	S	**0.097**	**0.729**	**4.510**	**0.191**	**0.886**	**0.958**	**0.979**
Monodepth2 [8]	1024×320	MS	0.106	0.806	4.630	0.193	0.876	0.958	0.980
HR-Depth [17]	1024×320	MS	0.101	0.716	4.395	**0.179**	**0.899**	0.966	0.983
R-MSFM6 [42]	1024×320	MS	0.108	0.753	4.469	0.185	0.888	0.963	0.982
BRNet (Ours)	1024×320	MS	**0.097**	**0.677**	**4.378**	**0.179**	0.888	0.965	**0.984**

Table 2. Comparison with other SOTA networks on KITTI Eigen split test set. The train column refers to the training data of the models. M means monocular videos only and S means stereo image pairs, and MS means both. The best two results are shown in bold and underlined, respectively.

Architecture	Abs Rel ↓	Sq Rel ↓	RMSE↓	log_{10} ↓
Monodepth	0.544	10.94	11.760	0.193
Monodepth2	0.322	3.589	7.414	0.163
BRNet	**0.302**	**3.133**	**7.068**	**0.156**

Table 3. Make3D results with monocular training and 640×192 inputs.

4 Experiments

We implement our network using Pytorch, and all the training and evaluation are performed on a workstation with an Intel E5-2698 v4CPU, 512G memory, and a single V100 GPU.

4.1 KITTI

The KITTI benchmark is the most widely used dataset in depth estimation, consisting of calibrated videos registered to LiDAR measurements of city scenarios.

The depth evaluation is done on the LiDAR point cloud. We adopt the data split like [4], followed by pre-processing as [41] to remove static frames. Finally, $39,810$ triplets are used for training and $4,424$ for validation. We use the same intrinsics for all images and set the camera's principal point to the image center and the focal length to the average of all the focal lengths in KITTI.

4.2 State-of-the-Art Comparison

Followed by [8], we compare the results of several variants of the proposed model, which is trained on different types of self-supervision. M, S and MS mean monocular video, stereo image pairs only and both, respectively. For the metrics, following [4] we adopt Abs Rel, Sq Rel, RMSE, RMSE log for which lower is better and $\delta < 1.25$, $\delta < 1.25^2$, $\delta < 1.25^3$ for which higher is better.

We compare our model to several state-of-the-art methods. As shown in Table 2, our approach outperforms all existing self-supervised networks with all the three types of self-supervision and two input resolutions. Compared to our baseline model monodepth2 [8], our model with monocular video training and small inputs achieves 0.010, 0.401 and 0.013 improvement in terms of AbsRel, RMSE and $\delta < 1.25$, respectively. As discussed in Sec.3.1, the original U-Net cannot fully exploit the detailed information in small images. Therefore, they cannot fully use the more detailed information carried by the large information either. Different from it, BRNet can extract abundant details from the inputs. When taking larger images, BRNet can extract more information than smaller ones and achieve better results. As a result, our model significantly outperforms our baseline when taking large inputs. We achieve 0.252 improvements on RMSE with 1024×320 inputs and MS training. To show the generality of our model, we follow the setting of [7] and evaluate BRNet on Make3D[27] dataset. According to the results shown in Table.3, BRNet significantly outperform our baseline with monocular training and 640×192 input.

We also compare the qualitative performance with other representative networks. As shown in Fig. 5, our model performs better than other networks, especially for objects far from the camera (marked by the red circles).

4.3 Ablation Study

We perform ablation studies to understand how the proposed components contribute to the overall performance improvements. All the experiments are evaluated on the KITTI Eigen split [4]. There are three components need evaluation:
Detail Branch (DB) The detail branch is responsible for extracting comprehensive features from inputs. As shown in the Table 4, the model taking small inputs with the detail branch even performs better with the baseline accepting large inputs gaining 0.004 improvement in terms of AbsRel and 0.192 in RMSE.
Global Branch (GB) The global branch is incorporated to extract perspective and other global information. In default, GDFM is a part of the global branch and is used to fuse the features from the two branches. To show the efficiency of GDFM, we also show a result without GDFM by replacing GDFM with an

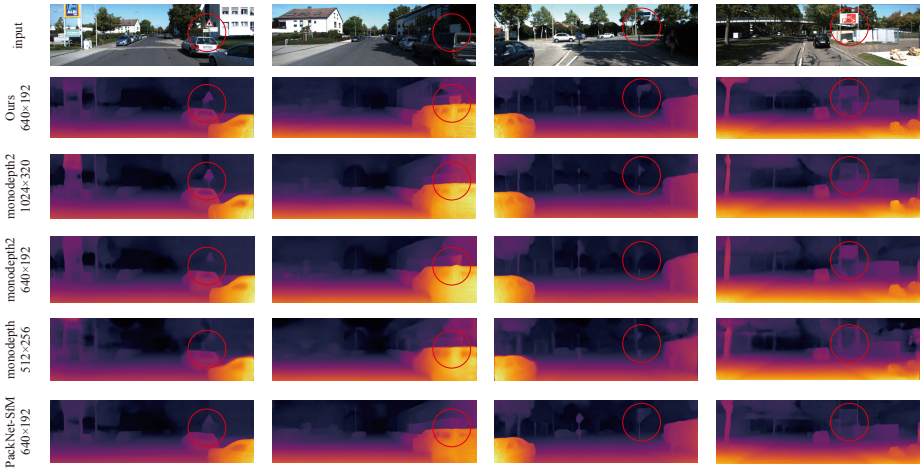


Fig. 5. Qualitative results on the KITTI Eigen split test set. Our model can fully exploit global and detailed information and outperforms previous SOTA networks.

Components				Abs Rel ↓	RMSE ↓	$\delta < 1.25$ ↑
DB	GB	GB w/o GDFM	RD			
				0.115	4.863	0.877
✓				0.111	4.672	0.880
✓		✓		0.109	4.579	0.889
✓	✓			0.108	4.538	0.890
✓	✓		✓	0.105	4.462	0.890
			✓	0.113	4.780	0.877

Table 4. Ablation study of the proposed components We compare the improvements from DB (Detail Branch), GB (Global Branch) RD (Residual Decoder) and Global Branch without GDFM, and the model with all the three components and GDFM achieves our best performance.

element-wise summation. As shown in Table 4, with global branch, the model achieves 0.108 in AbsRel while GDFM provide an obvious improvement.

Residual Decoder (RD) We demonstrate the performance improvement from our decoder. As shown in Table 4, the residual decoder outperforms the original decoder by 0.076 in terms of RMSE. As shown in the table, RD alone can improve the AbsRel by 0.002 and about 0.1 in RMSE.

Efficiency To extract abundant details from the inputs, BRNet introduces more computation than our baseline. To verify the complexity of our model, we compare its efficiency with two representative works in Table 5. As shown in it, BRNet is only slightly slower than its baseline, runing at 14ms (71fps), a speed far beyond real-time requirement. Compared with PackNet-SfM, BRNet largely outperform it with 15% computation, which can prove the improvement of our model does not come from the increasing computation.

Distance Comparison As discussed above, our method mainly improves the performance on objects far from the camera. To verify this, we split the validation

Architecture	Abs Rel ↓	FLOPs(B)	Params(M)	Speed(ms)
Monodepth2	0.115	8	14	10
BRNet	0.105	31	19	14
PackNet-SfM	0.111	205	128	154

Table 5. Comparison of speed, computation. All the speeds are reported on the the same device with a Tesla V100 GPU.

Method	Abs Rel		
	Near	Middle	Far
BRNet	0.059	0.073	0.147
Monodepth2	0.061	0.082	0.159

Table 6. Far objects comparison on different parts over the whole dataset.

(a) Near 0-8m (25%) (b) Middle 8-14m (35%) (c) Far 14-∞m (40%)

Fig. 6. Illustration and the proportion of different depth groups. We split the prediction and ground truth into three groups, Near, Middle and Far, according to the depth of each pixel.

dataset into three groups according to the depth of each pixel and the illustration and proportion of each group are shown in Fig.6. We then compare our model and the baseline model on the groups. In Table 6, two methods perform comparably on the Near part but BRNet outperforms Monodepth2 [8] significantly on the Middle and Far parts, which indicates that for more than 75% pixels in the dataset, our model can improve the performance from our baseline.

5 Conclusion

In this paper, we first make a deep study on the mechanism behind the influence of input resolutions and find out that the receptive field affects the performance rather than information difference between inputs. Indeed, depth estimators need a wide range of receptive fields. Thus they can extract detailed information, such as occlusion and global information like perspective. Based on these findings, we designed a bilateral depth encoder that simultaneously extracts details and global features and fuses them. Finally, we propose a residual decoder focusing on the difference between layers in the decoder, which is able to generate better predictions.

Acknowledgements. This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 62036010, 61972344), and the Start-up Research Grant (SRG) of University of Macau.

References

1. Filippo Aleotti, Fabio Tosi, Matteo Poggi, and Stefano Mattoccia. Generative adversarial networks for unsupervised monocular depth prediction. In *ECCV Workshops*, 2018.
2. Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *CVPR*, 2021.
3. Vincent Casser, Soeren Pirk, Reza Mahjourian, and Anelia Angelova. Depth prediction without the sensors: Leveraging structure for unsupervised learning from monocular videos. In *AAAI*, 2019.
4. David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.
5. David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *arXiv preprint arXiv:1406.2283*, 2014.
6. Ravi Garg, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *ECCV*, 2016.
7. Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.
8. Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *ICCV*, 2019.
9. Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *CVPR*, 2020.
10. Joel Janai, Fatma Guney, Anurag Ranjan, Michael Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *ECCV*, 2018.
11. Adrian Johnston and Gustavo Carneiro. Self-supervised monocular trained depth estimation using self-attention and discrete disparity volume. In *CVPR*, 2020.
12. Maria Klodt and Andrea Vedaldi. Supervising the new with the old: learning sfm from sfm. In *ECCV*, 2018.
13. Yevhen Kuznietsov, Jorg Stuckler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. In *CVPR*, 2017.
14. Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
15. Chenxu Luo, Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, Ram Nevatia, and Alan Yuille. Every pixel counts++: Joint learning of geometry and motion with 3d holistic understanding. *PAMI*, 42(10):2624–2641, 2019.
16. Yue Luo, Jimmy Ren, Mude Lin, Jiahao Pang, Wenxiu Sun, Hongsheng Li, and Liang Lin. Single view stereo matching. In *CVPR*, 2018.
17. Xiaoyang Lyu, Liang Liu, Mengmeng Wang, Xin Kong, Lina Liu, Yong Liu, Xinxin Chen, and Yi Yuan. Hr-depth: high resolution self-supervised monocular depth estimation. *CoRR abs/2012.07356*, 2020.
18. Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3d geometric constraints. In *CVPR*, 2018.
19. Ilya Makarov, Maria Bakhanova, Sergey Nikolenko, and Olga Gerasimova. Self-supervised recurrent depth estimation with attention mechanisms. *PeerJ Computer Science*, 8:e865, 2022.
20. S Mahdi H Miangoleh, Sebastian Dille, Long Mai, Sylvain Paris, and Yagiz Aksoy. Boosting monocular depth estimation models to high-resolution via content-adaptive multi-resolution merging. In *CVPR*, 2021.
21. Sudeep Pillai, Rareş Ambruş, and Adrien Gaidon. Superdepth: Self-supervised, super-resolved monocular depth estimation. In *ICRA*, 2019.

22. Matteo Poggi, Filippo Aleotti, Fabio Tosi, and Stefano Mattoccia. Towards real-time unsupervised monocular depth estimation on cpu. In *IROS*, 2018.
23. Matteo Poggi, Fabio Tosi, and Stefano Mattoccia. Learning monocular depth estimation with unsupervised trinocular assumptions. In *3DV*, 2018.
24. Pierluigi Zama Ramirez, Matteo Poggi, Fabio Tosi, Stefano Mattoccia, and Luigi Di Stefano. Geometry meets semantics for semi-supervised monocular depth estimation. In *ACCV*, 2018.
25. René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *CVPR*, 2021.
26. Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015.
27. Ashutosh Saxena, Min Sun, and Andrew Y Ng. Make3d: Depth perception from a single still image. In *AAAI*, 2008.
28. Ashutosh Saxena, Min Sun, and Andrew Y Ng. Make3d: Learning 3d scene structure from a single still image. *PAMI*, 31(5):824–840, 2008.
29. Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. In *CVPR*, 2003.
30. Evan Shelhamer, Jonathan T Barron, and Trevor Darrell. Scene intrinsics and depth from a single image. In *ICCV Workshops*, 2015.
31. Minsoo Song, Seokjae Lim, and Wonjun Kim. Monocular depth estimation using laplacian pyramid-based depth residuals. *IEEE Transactions on Circuits and Systems for Video Technology*, 2021.
32. Fabio Tosi, Filippo Aleotti, Matteo Poggi, and Stefano Mattoccia. Learning monocular depth estimation infusing traditional stereo knowledge. In *CVPR*, 2019.
33. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
34. Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.
35. Jamie Watson, Michael Firman, Gabriel J Brostow, and Daniyar Turmukhambetov. Self-supervised monocular depth hints. In *ICCV*, 2019.
36. Zhenheng Yang, Peng Wang, Yang Wang, Wei Xu, and Ram Nevatia. Lego: Learning edge with geometry all at once by watching videos. In *CVPR*, 2018.
37. Zhenheng Yang, Peng Wang, Wei Xu, Liang Zhao, and Ramakant Nevatia. Unsupervised learning of geometry with edge-aware depth-normal consistency. *arXiv preprint arXiv:1711.03665*, 2017.
38. Jure Zbontar, Yann LeCun, et al. Stereo matching by training a convolutional neural network to compare image patches. *J. Mach. Learn. Res.*, 17(1):2287–2318, 2016.
39. Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *CVPR*, 2018.
40. Wang Zhao, Shaohui Liu, Yezhi Shu, and Yong-Jin Liu. Towards better generalization: Joint depth-pose learning without posenet. In *CVPR*, 2020.
41. Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.
42. Zhongkai Zhou, Xinnan Fan, Pengfei Shi, and Yuanxue Xin. R-msfm: Recurrent multi-scale feature modulation for monocular depth estimating. In *ICCV*, 2021.