International Journal of Computer Vision (2022) 130:1196–1209
https://doi.org/10.1007/s11263-022-01595-8

Displacement-Invariant Cost Computation for Stereo Matching

Yiran Zhong[1] · Charles Loop[1] · Wonmin Byeon[1] · Stan Birchfield[1] · Yuchao Dai[2] · Kaihao Zhang[3] · Alexey Kamenev[1] · Thomas Breuel[1] · Hongdong Li[3] · Jan Kautz[1]

Received: 18 February 2021 / Accepted: 14 February 2022 / Published online: 16 March 2022

Abstract
Although deep learning-based methods have dominated stereo matching leaderboards by yielding unprecedented disparity accuracy, their inference time is typically slow, *i.e.*, less than 4 FPS for a pair of 540p images. The main reason is that the leading methods employ time-consuming 3D convolutions applied to a 4D feature volume. A common way to speed up the computation is to downsample the feature volume, but this loses high-frequency details. To overcome these challenges, we propose a *displacement-invariant cost computation module* to compute the matching costs without needing a 4D feature volume. Rather, costs are computed by applying the same 2D convolution network on each disparity-shifted feature map pair independently. Unlike previous 2D convolution-based methods that simply perform context mapping between inputs and disparity maps, our proposed approach learns to match features between the two images. We also propose an entropy-based refinement strategy to refine the computed disparity map, which further improves the speed by avoiding the need to compute a second disparity map on the right image. Extensive experiments on standard datasets (SceneFlow, KITTI, ETH3D, and Middlebury) demonstrate that our method achieves competitive accuracy with much less inference time. On typical image sizes (*e.g.*, 540 × 960), our method processes over 100 FPS on a desktop GPU, making our method suitable for time-critical applications such as autonomous driving. We also show that our approach generalizes well to unseen datasets, outperforming 4D-volumetric methods. We will release the source code to ensure the reproducibility.

Keywords Stereo matching · Feature volume · Autonomous driving · Displacement-invariant cost computation

1 Introduction

Deep learning-based methods have achieved state-of-the-art on most of the standard stereo matching benchmarks (*i.e.*, KITTI Menze and Geiger 2015, ETH3D Schöps et al., 2017, and Middlebury Scharstein et al., 2014). This success is achieved by aggregating information in a 4D feature volume (height × width × disparity levels × feature dimension), which is formed by concatenating each feature in one image with its corresponding feature in the other image, across all pixels and disparity levels. To process such a 4D volume, expensive 3D convolutions are utilized, thus making these methods significantly more time- and space-intensive than traditional approaches like semi-global matching (SGM) Hirschmuller (2008) and its variants Hernandez-Juarez et al. (2016). For example, the traditional method known as embedded SGM Hernandez-Juarez et al. (2016) achieves 100 FPS (frames per second) for a pair of 540p images, whereas most deep learning-based methods only manage about 2 FPS. Moreover, since this 4D feature volume grows with the cube of resolution, high-resolution depth estimation is prohibitively expensive.

In this paper, we propose to overcome these limitations with a *displacement-invariant cost computation module* that learns to match features of stereo images *using only 2D convolutions*. Unlike previous 2D convolution-based methods (Mayer et al., 2016; Liang et al., 2018; Pang et al., 2017), however, ours does not rely upon context matching between pixels and disparities; rather, due to its unique design, our network learns to match pixels between the two images. The

Communicated by Akihiro Sugimoto.

This work was done when he was an intern in NVIDIA, Redmond, WA 98052.

✉ Yiran Zhong
zhongyiran@gmail.com

[1] NVIDIA, Redmond, WA 98052, USA

[2] Northwestern Polytechnical University, Xi'an, China

[3] Australia National University, Canberra, Australia

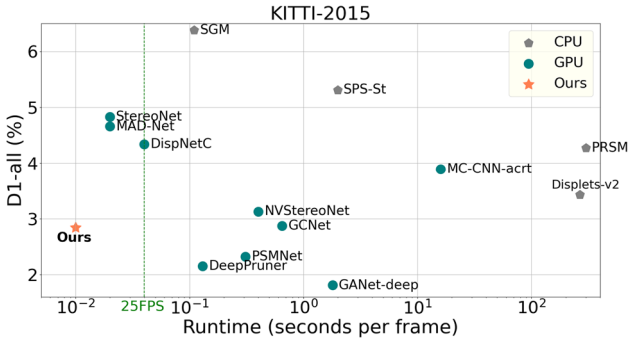


Fig. 1 Our method achieves stereo matching accuracy comparable to state-of-the-art on the KITTI 2015 *test* dataset, while operating at 100 FPS. (All GPU methods are timed on an NVIDIA RTX Titan GPU. For the methods that do not release their source code, we use our own implementation instead. Best viewed on screen.)

key insight behind our approach is to compute the cost of each disparity shift independently using the same 2D convolution-based network. This vastly reduces the number of parameters and memory requirements for training and it also reduces time and memory costs during inference, as it does not need to explicitly store a 4D feature volume before cost computation. Also, we leverage entropy maps that are computed from 3D cost volumes as confidence maps to guide the refinement of disparities. As a result (*cf.*, Fig. 1), our proposed method is not only significantly faster than 3D convolution-based volumetric methods, *e.g.*, GA-Net Zhang et al. (2019), but it also achieves better cross-dataset generalization ability. The entire system is trained end-to-end.

Our paper makes the following contributions:

- A displacement-invariant cost computation module for stereo matching that uses 2D convolutions on disparity-shifted feature map pairs, which achieves significant speedup over standard volumetric methods with much lower memory requirements.
- A new entropy based refinement scheme that bypasses the need for estimating disparity for the right view, which further reduces the processing time and memory consumption.
- State-of-the-art performance on all three benchmarks compared with existing real-time methods, and better generalization performance than existing methods.

2 Related Work

Given a pair of rectified stereo images, stereo matching attempts to find a matching point for each pixel on the corresponding epipolar line. Stereo matching has been extensively studied for decades in computer vision. Here we discuss a few popular and recent methods that are closely related to our approach. Interested readers are referred to recent survey papers such as Scharstein and Szeliski (2002) and Janai et al. (2017).

2.1 Traditional Stereo Matching

Traditional stereo matching methods can be roughly divided into two classes: local methods and global methods. The typical pipeline for a local method has four consecutive steps: (1) compute costs at a given disparity for each pixel; (2) sum up the costs over a window; 3) select the disparity that has the minimal cost; (4) perform a series of post-processing steps to refine the final results. Local methods (Scharstein and Szeliski 2002; Weber et al., 2009) have the advantage of speed. Since each cost within a window can be independently computed, these methods are highly parallelizable. The disadvantage of such methods is that they can only achieve sub-optimal results because they only consider local information. Global methods (Zhang et al., 2015; Taniai et al., 2018) have been proposed to address this issue. They treat the disparity assignment task as a maximum flow / minimum cut problem and try to minimize a global cost function. However, such algorithms are typically too slow for real-time applications. Semi-global matching (SGM) Hirschmuller (2008) is a compromise between these two extremes, which could achieve more accurate results than local methods without sacrificing speed significantly.

2.2 Deep Stereo Matching

Unlike traditional methods, deep stereo matching methods can learn to deal with difficult scenarios, such as repetitive textures or textureless regions, from ground truth data. A deep stereo method is often trained to benefit from one of the following aspects: (1) learning better features or metrics (Žbontar and LeCun 2016; Cai et al., 2020); (2) learning better regularization terms (Seki and Pollefeys 2017); (3) learning better refinement Khamis et al. (2018); (4) learning better cost aggregation (Zhang et al., 2019; Cai and Mordohai 2020); (5) learning direct disparity regression Mayer et al. (2016). Based on the network structure, we divided these methods into two classes:

Direct Regression methods often use an encoder-decoder to directly regress disparity maps from input images (Mayer et al., 2016; Liang et al., 2018; Tonioni et al., 2019; Yin et al., 2019). Such methods learn in a brute force manner, discarding decades of acquired knowledge obtained by classical stereo matching research. When there are enough training data, and the train and test distributions are similar, such methods can work well. For example, iResNet Liang et al. (2018) won first place in the 2018 Robust Vision Challenge. Also, since such methods only employ 2D convolutions, they can easily achieve real-time or near real-time processing and

have low GPU memory consumption. However, these methods lack the ability to generalize effectively, *e.g.*, DispNet Mayer et al. (2016) fails random dot stereo tests Zhong et al. (2018).

Volumetric Methods (Kendall et al., 2017; Zhang et al., 2019; Zhong et al., 2017; Smolyanskiy et al., 2018; Chang et al., 2020; Zhang et al., 2020; Cai and Mordohai 2020; Cai et al., 2020; Badki et al., 2021) build a 4D feature volume using features extracted from stereo pairs, which is then processed with multiple 3D convolutions. These methods leverage the concept of semi-global matching while replacing hand-crafted cost computation and cost aggregation steps with 3D convolutions. Since this type of network is forced to learn matching, volumetric methods easily pass random dot stereo tests Zhong et al. (2018). However, they suffer from high processing times and high GPU memory consumption Zhang et al. (2019). To overcome these issues, some researchers build the feature volume at a lower image resolution to reduce memory footprint (Khamis et al., 2018; Yang et al., 2019), or prune the feature volume by generating a confidence range for each pixel and aggregating costs within it (Duggal et al., 2019; Zhou et al., 2018). However, lowering the resolution of the feature volume makes it difficult to recover high-frequency information. To restore these lost details, volumetric networks typically use color-guided refinement layers as a final processing stage.

2.3 Real-time Stereo Matching

Several real-time deep stereo matching networks have been proposed recently. Khamis et al. (2018) achieves real-time performance at the cost of losing high frequency details due to a low resolution 4D cost volume, *i.e.*, 1/8 or 1/16 of the input images. (Mayer et al., 2016; Tonioni et al., 2019) discard the general pipeline of stereo matching and design a context regression network that directly regresses disparity maps from the input images; these approaches suffer generalization problems as shown in Zhong et al., (2018). Recently, Bi3D Badki et al. (2020) proposes a trade-off strategy to balance the processing latency and depth quality. By regressing a binary mask from a stereo image pair and their matching costs for each disparity level, it can classify objects as being closer or farther than a given distance or provide quantized depth with different quantization. Note that in this case, they cannot generate disparity maps with sub-pixel accuracy. To alleviate this drawback and predict disparity maps with a *full continuous range*, they move back to a 3D encoder-decoder architecture to aggregate information from the neighboring disparity planes and thus loss the real-time processing capability. To overcome the limitations of previous methods, our proposed approach uses a displacement-invariant cost computation module to achieve super real-time inference with *full range continuous* disparity estimation while, at the same time, preserving most high-frequency details by constructing a 3D cost volume on 1/3 of the input resolution. Moreover, experiments in Sect. 4.2 show that our method achieves better generalization ability than 4D volumetric methods.

3 Method

In this section, we start with the overall pipeline of our framework and then provide a detailed analysis of our proposed *displacement-invariant cost computation* module. The details of our network design is also included in this section.

3.1 Framework Overview

As shown in Fig.2, our stereo framework consists of four major components *i.e.*, Feature Net, Matching Net, Projection Layer and Refine Net. The information flow is as follows. Given a pair of stereo images, the Feature Net first embeds them into feature maps. Then the Matching Net takes left features and disparity-shifted right features at each disparity level to generate a 3D cost volume. The Projection Layer projects the cost volume to obtain a disparity map while also computing an entropy map. Finally, the Refine Net takes the left image and the output of the Projection Layer to generate the final disparity map.

The key difference between our framework and previous volumetric methods is the cost computation part. In the existing volumetric methods, they often need to construct a 4D feature volume and use 3D convolutions to compute the matching cost, which are the main barriers for applying volumetric methods to high-resolution images. However, in our solution, we need neither 4D feature volumes nor 3D convolutions to learn the matching cost. By processing each disparity shift independently, our network only needs 2D convolutions to perform cost computation. We also propose a new refinement scheme using entropy, which bypasses the need to estimate disparity for the right view.

3.2 Displacement-Invariant Cost Computation

Existing volumetric methods construct a 4D feature volume, which is processed with 3D convolutions to learn the matching cost, as shown in Fig. 3a. With such an approach, the matching cost for pixel $\mathbf{p}$ at disparity level d is calculated as:

$$c_{3D}(\mathbf{p}, d) = g_{3D}(\phi_{4D}(f(I^L(\mathbf{p})) \,\|\, f(I^R(\mathbf{p} - d)))), \quad (1)$$

where $f(\cdot)$ is a feature network to convert images to feature maps, $\phi_{4D}(\cdot\|\cdot)$ denotes the concatenation of disparity-shifted feature map pairs on every possible disparity shift, and $g_{3D}(\cdot)$ is a 3D convolution-based matching network that computes and aggregates the matching cost based on feature

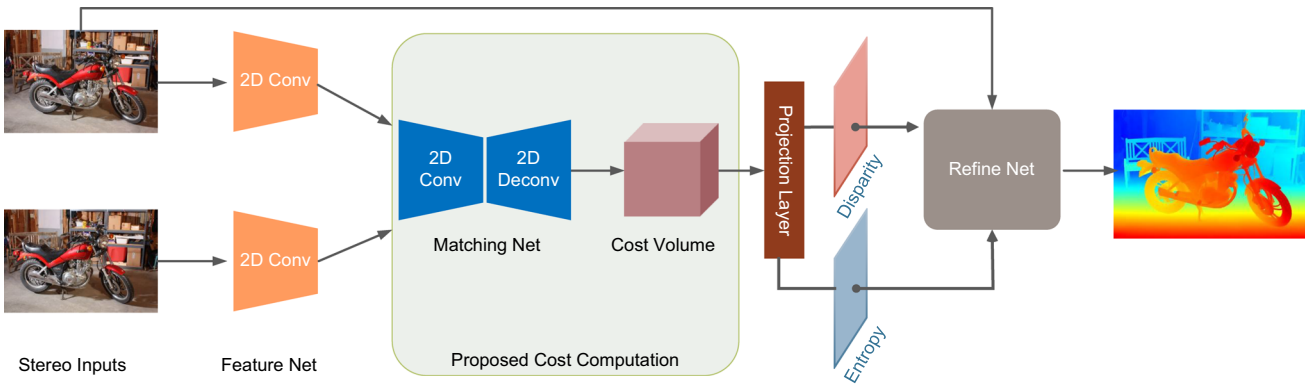
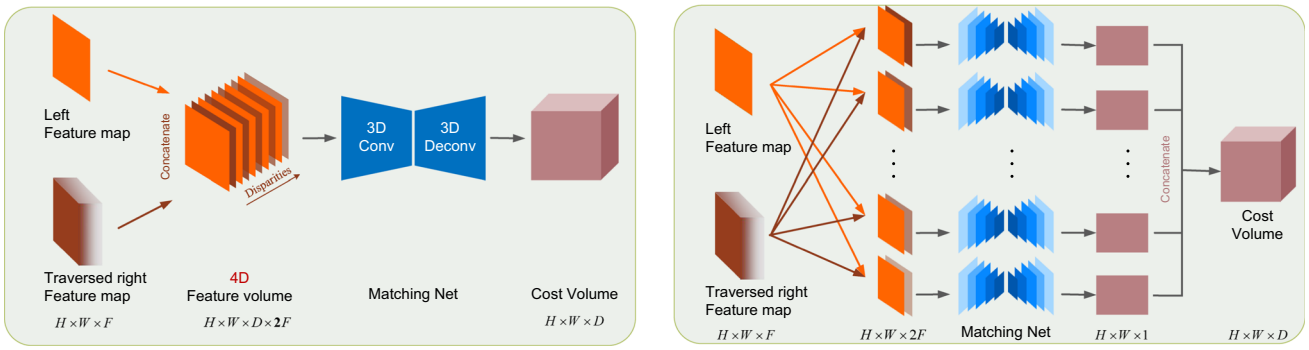


Fig. 2 Overall Architecture of our Stereo Network, consisting of four components: Feature Net, Matching Net, Projection Layer and Refine Net. Given a pair of stereo images, the Feature Net produces feature maps which are processed by the Matching Net to generate a 3D cost volume (see Sect. 3.2). The Projection Layer projects the volume to obtain a disparity map while also computing an entropy map. The Refine Net takes the left image and the output of the Projection Layer to generate the final disparity map.



(a) **(b)**

Fig. 3 Volumetric method and our Proposed method. Our method runs share-weighted Matching Net on disparity shifted feature maps using 2D convolutions before 3D cost volume formation. In contrast, previous volumetric methods perform matching using 3D convolutions after constructing a 4D feature volume.

maps and neighboring disparity shifts. Therefore, the cost will be different if we shuffle the concatenated feature maps along the disparity dimension.[1]

In contrast, our proposed Displacement-Invariant Cost Computation (DICC) only requires 2D convolutions. As shown in Fig. 3b, the exact same matching net is applied to each disparity-shifted feature map pair, thus ensuring that cost computation is only dependent on the current disparity shift. Unlike 3D convolution-based methods, our matching cost will be identical if we shuffle the concatenated feature maps along the disparity dimension.

Specifically, let us consider a matching cost computation for pixel $\mathbf{p}$ at disparity level d. In our method, we compute the matching cost as follows:

$$c_{2D}(\mathbf{p}, d) = g_{2D}(f(I^L(\mathbf{p})), f(I^R(\mathbf{p} - d))), \quad (2)$$

where $f(\cdot)$ is a feature net, and $g_{2D}(\cdot)$ is a matching net that computes the matching cost at each disparity level independently. Similar to MC-CNN Abontar and LeCun (2016), our proposed approach learns a cost from a deep network. However, MC-CNN computes the matching cost based on patches and then uses traditional semi-global cost aggregation steps to generate the disparity map. Our approach, on the other hand, is an end-to-end method that uses a network to perform cost computation and aggregation simultaneously.

To better understand and compare the matching cost computation, we implement *Baseline3D* using Eq. (1) for cost computation, and we implement *Ours* using Eq. (2) to compute the matching cost. Baseline3D and Ours networks share the same feature net and projection layer, but Baseline3D

[1] In our DICC, the total Operations for constructing a cost volume should be $5.4 \times 64 = 345.6$ Gflops. However, since the 64 Matching Nets can process in parallel, we only put the Operations in one Matching Net in the table.

Table 1 Computational resource comparison between Our Matching Net and Baseline3D Matching Net. Runtime is measured on $540(H) \times 960(W)$ resolution stereo pairs with $192(D)$ disparity levels. Note that the Operations are computed on one Matching Net2 .

Methods	Runtime(s)	Params	Memory	Operations
3D Matching Net	0.150	3.84M	$\frac{1}{3}H \times \frac{1}{3}W \times \frac{1}{3}D \times 2F$	733.8 Gflops
Our Matching Net	0.01	1.16M	$\frac{1}{3}H \times \frac{1}{3}W \times 2F$	5.4 Gflops

constructs a 4D feature volume as in Fig. 3a and replaces the 2D convolutions in our Matching Net with 3D convolutions. A detailed comparison can be found in Sect. 4.2.

In Table 1, we compare the computational resources needed for Baseline3D and our proposed Matching Net. Note that in order to build a cost volume, we need to run our Matching Net $\frac{1}{3}D$ times (the cost volume is built on $\frac{1}{3}D$ of the image size.). Since all disparity levels are processed independently, we are allowed to run our Matching Net either in parallel or sequentially. To construct a cost volume of size $\frac{1}{3}H \times \frac{1}{3}W \times \frac{1}{3}D$, the former one will need a memory of $\frac{1}{3}H \times \frac{1}{3}W \times \frac{1}{3}D \times 2F$ but can finish in $0.010s$ (including memory allocation time). The latter one just needs a memory of $\frac{1}{3}H \times \frac{1}{3}W \times 2F$ as all Matching Nets can share the same memory space and can finish in $0.07s$. We ignore the memory cost of convolution layers here as they may vary on different architectures. It is worth noting that the main obstacle for training a volumetric network on high resolution stereo pairs is that it requires the network to initialize a giant 4D feature volume in GPU memory before cost computation and the size of the feature volume increases as the cube of the inputs. However, since the 4D feature volume is not required for our method, we can handle high resolution inputs and large disparity ranges, *i.e.*, our method can process a pair of 1500×1000 images with 400 disparity levels in Middlebury benchmark without further downsampling the feature size to $1/64$ as in HSM Yang et al.(2019).

Another advantage of our method is its better generalization capability. Unlike volumetric methods that utilize 3D convolutions to regularize H, W, D dimensions simultaneously, we force our method to learn matching costs only from H, W dimensions. We argue that connections between $H \times W$ plane and D dimension is a double-edged sword in computing the matching costs. While such connections do improve results on a single dataset, in practice we find that they lead to higher cross-dataset errors as verified in Sect. 4.2.

3.3 Network Architecture Details

Our framework consists of four major parts: (1) Feature Net, (2) Matching Net, (3) Projection Layer, and (4) Refine Net as shown in Fig. 2. Our architecture design follows the general pipeline in deep stereo matching but with significant differences as explained below.

Table 2 Feature Net architecture. Conv_f8 does not have batch normalization or ReLU

Feature Net layer	**k, s, d**	**chns**	**input**
conv_f1	3×3, 1, 1	3/32	image
conv_f2	3×3, 3, 1	32/64	conv_f1
conv_f3	3×3, 1, 4	64/128	conv_f2
conv_f4	3×3, 1, 8	128/128	conv_f3
branch_1	64 × 64 avg. pool 1 × 1, 1, 1, 128/32 bilinear interpolation		conv_f4
branch_2	16 × 16 avg. pool 1 × 1, 1, 1, 128/32 bilinear interpolation		conv_f4
conv_f7	3×3, 1, 1	192/96	conv_f4+branch1+branch2
conv_f8	1×1, 1, 1	96/32	conv_f7

Feature Net. A deeper feature net has a larger receptive field and can extract richer information. Therefore, researchers often use more than 15 layers for feature extraction (Kendall et al., 2017; Khamis et al., 2018; Zhang et al., 2019). In practice, we find that increasing the number of layers in the feature net does not help much for the final accuracy, *e.g.*, MC-CNN Abontar and LeCun (2016). Therefore, in our feature net design, we use a shallow structure that only contains 8 convolutional layers. We first downsample the input images using a convolution layer with a stride of 3. Three dilated convolutions are then applied to enlarge the receptive field. We also adopt a reduced spatial pyramid pooling (SPP) module He et al. (2014) to combine features from different scales to relieve the fixed-size constraint of a CNN. Our SPP module contains two average pooling layers. Each of them follows a convolution and a bilinear upsampling layer. We concatenate feature maps that feed into our SPP module, then pass them to a convolution with output channel size of 96. The final feature map is generated by a convolution without batch normalization and activation functions. Following previous work (Kendall et al., 2017; Khamis et al., 2018; Zhang et al., 2019), we use 32-channel feature maps. A detailed description is shown in Table 2.

Matching Net. Our Matching Net computes a cost map on each disparity level. We adopt a skip-connected U-Net

Table 3 Matching Net architecture. Conv_m14 does not have batch normalization or ReLU.

Matching Net layer	k, s, d	chns	input
conv_m1	3×3, 1, 1	64/32	concat features
conv_m2	3×3, 2, 1	32/48	conv_m1
conv_m3	3×3, 1, 1	48/48	conv_m2
conv_m4	3×3, 2, 1	48/64	conv_m3
conv_m5	3×3, 1, 1	64/64	conv_m4
conv_m6	3×3, 2, 1	64/96	conv_m5
conv_m7	3×3, 1, 1	96/96	conv_m6
conv_m8	3×3, 2, 1	96/128	conv_m7
conv_m9	3×3, 1, 1	128/128	conv_m8
deconv_m1	4×4, 2, 1	128/96	conv_m9
conv_m10	3×3, 1, 1	192/96	deconv_m1+conv_m7
deconv_m2	4×4, 2, 1	96/64	conv_m10
conv_m11	3×3, 1, 1	128/64	deconv_m2+conv_m5
deconv_m3	4×4, 2, 1	64/48	conv_m11
conv_m12	3×3, 1, 1	96/48	deconv_m3+conv_m3
deconv_m4	4×4, 2, 1	48/24	conv_m12
conv_m14	3×3, 1, 1	24/1	deconv_m4

Ronneberger et al. (2015) with some modifications. In particular, we downsample each concatenated feature map four times using 3 × 3 convolutions with a stride of 2. For each scale, we filter the feature maps with one 3 × 3 convolution followed by a batch normalization layer and a ReLU activation layer. We set the feature size of each scale at 48, 64, 96 and 128 respectively. For upsampling layers, we use bi-linear upsampling and a 3 × 3 convolution layers with a stride of 1 and reduce the feature dimension accordingly. A 3 × 3 convolutional layer with feature size of 1, with no batch normalization nor activation is applied to generate the final cost map. Our Matching Net detailed structure is shown in Table 3.

Projection layer. After building a 3D cost volume, we use a projection layer to select the disparity with the lowest matching cost. Similar to previous volumetric methods (Kendall et al., 2017; Zhong et al., 2018; Zhang et al., 2019), we use the soft-argmin operation to generate a disparity. The soft-argmin operation is defined as:

$$\hat{d} := \sum_{d=0}^{D} [d \times \sigma(-c_d)], \tag{3}$$

where c_d is the matching cost at disparity d, D is the preset maximum disparity level and $\sigma(\cdot)$ denotes the softmax operator.

With little additional computation, the projection layer also generates an entropy map to estimate the confidence of each pixel. The entropy of each pixel is defined as:

$$h = -\sum_{d=0}^{D} \sigma(-c_d) \log(\sigma(-c_d)). \tag{4}$$

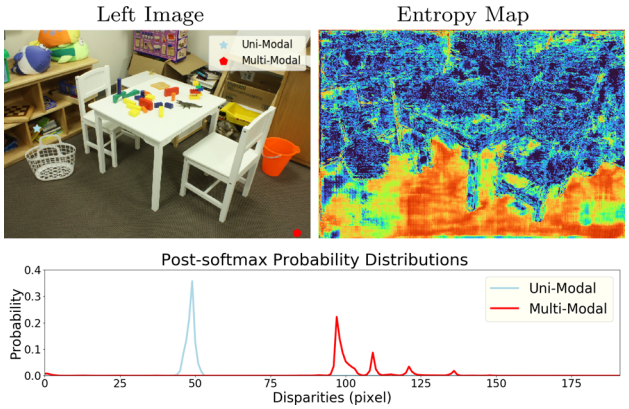


Fig. 4 Entropy Map Analysis. In the entropy map, red color corresponds to high entropy value. Best viewed in color.

We apply a softmax operation on the cost to convert it to a probability distribution before computing the entropy. Fig. 4 shows the resulting entropy map: pixels on textureless areas (*e.g.*, red dot in the carpet) have high entropy, whereas the pixels on the texture-rich areas (*e.g.*, blue dot in the shelf) have low entropy. The bottom of the figure shows the post-softmax probability distribution curves of two selected pixels. A unimodal curve indicates low entropy (high confidence) for the pixel's estimated disparity, whereas a multimodal curve indicates high entropy (low confidence).

Refine Net. There are several choices in designing our refine net. StereoNet Khamis et al. (2018) proposes a refine net that takes the raw disparity map and the left image as inputs. StereoDRNet Chabra et al. (2019) uses a similar refine net but takes the disparity map, the left image, image warping error map and disparity warping error map as inputs. The main drawback of such a design is that computing these error maps requires both left and right disparity maps, which costs extra time and memory. As shown in Fig. 4, the entropy map can provide similar information with less computation and memory. Therefore, we use a similar refine net architecture as StereoDRNet Chabra et al. (2019) (Table 4) but use the disparity map, left image and entropy map as inputs.

There is a long history of leveraging entropy as a certainty measurement in conventional stereo matching algorithms. Scharstein and Szeliski (1998) takes the negative entropy of the probability distribution in the disparity hypotheses to form an explicit local distribution model. This allows the algorithm to control the amount of diffusion based on the disparity confidence. In this paper, we introduce the entropy

Table 4 Refine Net architecture

Refine Net layer	k, s, d	chns	input
conv_r1	3×3, 1, 1	1/16	disparity
conv_r2	3×3, 1, 1	20/32	conv_r1+entropy+left image
ResBlock_1	3×3, 1, 1	32/32	conv_r2
ResBlock_2	3×3, 1, 2	32/32	ResBlock_1
ResBlock_3	3×3, 1, 4	32/32	ResBlock_2
ResBlock_4	3×3, 1, 8	32/32	ResBlock_3
ResBlock_5	3×3, 1, 1	32/32	ResBlock_4
ResBlock_6	3×3, 1, 1	32/32	ResBlock_5
conv_r3	3×3, 1, 1	32/1	ResBlock_6
output	ReLU(conv_r3 + disparity)		

map to a supervised deep framework and serve it as an attention mechanism for disparity refinement.

It is also worth noting that our entropy map is different from the uncertainty map that has been widely used in depth estimation algorithms Poggi et al. (2020). The uncertainty map is often estimated by a network Kim et al. (2019) or from a Bayesian model Poggi et al. (2020). In our framework, we only need a guidance for the Refine Net to indicate the areas that may not be accurately estimated. Both the uncertainty and the entropy map can serve this purpose but the later one has much lower computational complexities.

3.4 Loss Function

Following GA-Net Zhang et al. (2019), we use the smooth ℓ_1 loss as our training loss function, which is robust at disparity discontinuities and has low sensitivity to outliers or noise. Our network outputs two disparity maps: a coarse prediction $\mathbf{d}_{\text{coarse}}$ from the soft-argmin operation and a refined one $\mathbf{d}_{\text{refine}}$ from our refine net. We apply supervision on both of them. Given the ground truth disparity $\mathbf{d}_{gt}$, the total loss function is defined as:

$$\mathcal{L}_{\text{total}} = \ell(\mathbf{d}_{\text{coarse}} - \mathbf{d}_{\text{gt}}) + \lambda\ell(\mathbf{d}_{\text{refine}} - \mathbf{d}_{\text{gt}}), \tag{5}$$

where $\lambda = 1.25$, and

$$\ell(x) = \begin{cases} 0.5x^2, & |x| < 1 \\ |x| - 0.5, & \text{otherwise.} \end{cases} \tag{6}$$

4 Experimental Results

We implemented our stereo matching network in PyTorch. We used the same training strategy and data argumentation as described in GA-Net Zhang et al. (2019) for easy comparison. Our network was trained in an end-to-end manner with the Adam optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$). We randomly cropped the input images to 240×576 and used a batch size of 104 on 8 Nvidia Tesla V100 GPUs. We pre-trained our network from random initialization and a constant learning rate of 1e-3 on the SceneFlow dataset for 60 epochs. Each epoch took around 15 minutes and the total pre-training process took around 15 hours. For inference, our network can run 100 FPS on NVIDIA Titan RTX and occupies less than 2 GB memory for a pair of 384×1280 stereo images. We add different disparity levels as a batch and run the Matching Net over it to achieve parallelism. We comprehensively studied the characteristics of our network in ablation studies and evaluated our network on leading stereo benchmarks. For each benchmark results, we follow the training protocol of GANet Zhang et al. (2019) and finetune our SceneFlow pretrained model on each benchmark separately.

4.1 Datasets

Our method was evaluated on the following datasets:

SceneFlow dataset. SceneFlow Mayer et al. (2016) is a large synthetic dataset containing 35454 training and 4370 testing images with a typical image dimension of 540×960. This dataset provides both left and right disparity maps, but we only use the left ones for training. Note that for some sequences, the maximum disparity level is larger than a preset limit (192 for this dataset), so we exclude these pixels in our training and evaluation. We use this dataset for pre-training and ablation studies.

KITTI 2015 dataset. KITTI 2015 contains 200 real-world drive scene stereo pairs for training and 200 for testing. They have a fixed baseline but the focal lengths could vary. The typical resolution of KITTI images is 376×1240. The semi-dense ground truth disparity maps are generated by Velodyne HDL64E LiDARs with manually inserted 3D CAD models for cars Menze and Geiger (2015). From the training set, we randomly select 160 frames for training and 40 frames for validation. We use a maximum disparity level of 192 in this dataset. We report our results on the test set benchmark, whose ground truth is withheld.

ETH3D stereo dataset. ETH3D is a small dataset that contains 27 images for training and 20 for testing, for both indoor and outdoor scenes. It is a challenging dataset in that most stereo pairs have different lighting conditions. In other words, the Lambertian assumption for stereo matching may not hold in some areas. Moreover, unlike all the other datasets which have color inputs, ETH3D is grayscale. It requires the network to handle different channel statistics and extract features that are robust to lighting conditions. The maximum input resolution is 576×960. Since the maximum disparity of this dataset is very small, we reduce the maximum disparity level to 48 for this dataset.

Middlebury 2014 stereo dataset. Middlebury 2014 is another small dataset. It contains 15 images for training and 15 for testing. This dataset is challenging for deep stereo methods not only because of the small size of the training set, but also due to the high resolution imagery with many thin objects. The full resolution of Middlebury is up to 3000 × 2000 with 800 disparity levels. To fit in GPU memory, most deep stereo methods can only operate on quarter-resolution images. As a consequence, many details are missed at that resolution which leads to a reduced accuracy. We use half resolution and 432 disparity levels.

4.2 Model Design Analysis

In this section, we analyze the components of our network structure and justify the design choices, including the performance differences between our proposed method, hand-crafted cost computation and Baseline3D, the effectiveness of entropy map and the robustness and generalizability of our network. All experiments are conducted on the SceneFlow dataset except the robustness experiment.

To fairly analyze the benefits and disadvantages of our method and volumetric method (Baseline3D), we compare them using exactly the same training strategy, and settings. For the sake of completeness, we also compare the performance with hand-crafted cost using $c_{SSD}(\mathbf{p}, d) = \sum_{\mathbf{q}\in\mathcal{N}_\mathbf{p}} \left\| f(I^L(\mathbf{p})) - f(I^R(\mathbf{p}-d)) \right\|_2^2$, where $\mathcal{N}_\mathbf{p}$ is a local patch around $\mathbf{p}$.

A detailed comparison is shown in Table 5. As expected, using hand-crafted costs fails to generate meaningful qresults. Baseline3D achieves the best performance in accuracy but is 15× slower than Ours. Compared with Baseline3D, the performance of proposed method increases from 0.78 pixels EPE to 1.09 pixels. In fact, our method achieves the top accuracy among all real-time algorithms as shown in Table 6.

A qualitative comparison of Ours and Baseline3D is provided in Fig. 5, where both approaches generate high-quality disparity maps. Baseline3D has better occlusion handling ability: By jointly regularizing spatial and disparity domain, the network can better pre-scale the matching costs to handle the multi-modal situation. However, Baseline3D has a drawback that we will address in the next section. We further analyze the contribution of each component of proposed network, including the use of refine net and the entropy of the cost volume as input to the refine net. In Table 5, we can see that by adding the refine net, the EPE drops from 1.20 to 1.14 and the bad-1.0 drops from 10.31 to 10.16. We also compare our approach with a left-right consistency check strategy (using the color difference between the left image and its corresponding warped right image) and find that it further boosts the accuracy more than 5% but significantly increases the processing time, whereas adding the entropy map can have similar performance without sacrificing the efficiency. We posit that the entropy map provides evidence for the Refine Net as to which parts of the disparity map are unreliable and need to be smoothed using priors.

Table 5 Contributions of each component. The first 3 rows compare the performance (both speed and accuracy) between hand-crafted cost (Dot), Baseline3D and our method. The last 4 rows show the contribution of each components of our method. We also include a comparison between our strategy and a left-right consistency check strategy (L-R Check) for the RefineNet.

Architecture Variant						Inference	SceneFlow	
Dot	3D	2D	RefineNet	Entropy	L-R Check	time(s)	EPE	bad-1.0
✓						0.001	6.05	59.13
	✓					0.150	0.78	8.11
		✓				0.007	1.20	10.31
		✓	✓			0.010	1.14	10.16
		✓	✓		✓	0.020	1.08	9.84
		✓	✓	✓		0.010	1.09	9.67

Table 6 Quantitative results on SceneFlow dataset. Our model is the fastest and the most accurate among real-time algorithms.

	Methods	EPE (pix)	bad-1.0	Params	Runtime(s)
Non-real time	GC-Net Kendall et al. (2017)	1.84	15.60	3.50M	0.65
	CRL Pang et al. (2017)	1.32	-	78.77M	0.26
	PSMNet Chang et al. (2018)	1.09	12.10	5.22M	0.31
	GA-Net Zhang et al. (2019)	0.78	8.70	6.58M	1.80
	DPruner Duggal et al. (2019)	0.86	-	-	0.13
	Baseline3D	0.78	8.11	4.26M	0.15
Realtime	DispNetC Mayer et al. (2016)	1.68	-	-	0.04
	StereoNet Khamis et al. (2018)	1.10	-	-	0.02
	Ours	1.09	9.67	1.70M	0.01

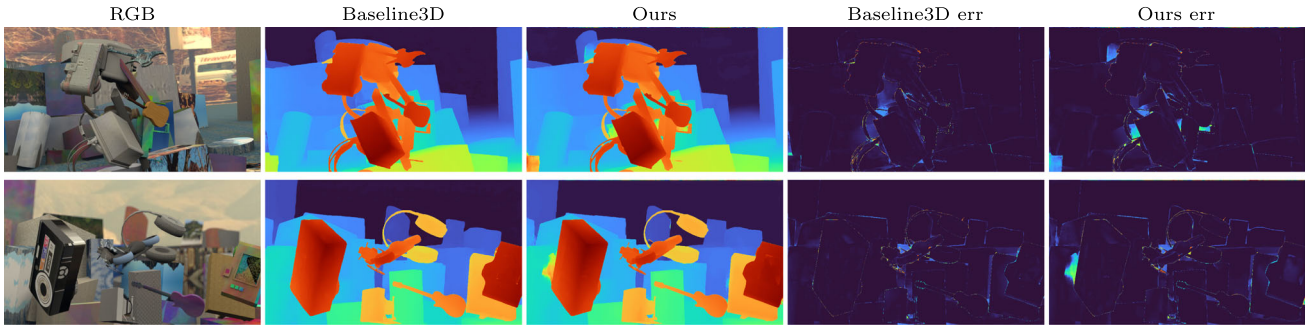


Fig. 5 Qualitative comparison on the SceneFlow dataset.

In Table 6, we compare our methods with SOTA deep stereo methods on the SceneFlow dataset. Baseline3D achieves top performance (in terms of both accuracy and speed) among non-real time methods, while our method with refinement achieves the top performance among real-time methods. It is worth noting that previous 2D convolution based methods (Pang et al., 2017; Liang et al., 2018) need a large number of parameters to learn the context mapping between inputs and disparity maps, *i.e.*, directly regressing disparity from color images. By using the proposed cost computation, our network achieves better accuracy, is significantly faster, and requires a fraction of the parameters as previous 2D convolution based methods.

4.3 Evaluation on Benchmarks

In this section, we provide results on three well-known stereo matching benchmarks: KITTI 2015, ETH3D, and Middlebury. We fine-tuned the SceneFlow pre-trained model on each benchmark. The algorithms are divided into two categories based on runtime: below 10 FPS and above 10 FPS.

KITTI 2015. Table 7 shows the accuracy and runtime of leading algorithms on the KITTI 2015 benchmark. Our method is the fastest and most accurate among real-time algorithms. Fig. 6 visualizes several results on the test set.

ETH3D. Table 8 shows quantitative results of our method on the ETH3D benchmark. Our method achieves competitive performance while being significantly faster than all the other methods. A qualitative comparison of our method with other SOTA algorithms is shown in Fig. 7.

Middlebury 2014. Our method is the only deep learning-based method that can achieve real-time performance on the Middlebury dataset. By accuracy, it is the second best among all competitors, as shown in Table 9. Compared to the most accurate approach (HSM), our method is 12× faster while has comparable accuracy. Qualitative results are shown in Fig. 8.

Table 7 Results on KITTI 2015 *test* set. Bold indicates the best, while underline indicates the second best. All GPU methods are timed on an NVIDIA RTX Titan GPU. For the methods that do not release their source code, we use our own implementation instead.

	Method	Runtime(s)	Non-occ (%)			All (%)		
			bg	fg	all	bg	fg	all
Non-real time	MC-CNN Abontar and LeCun (2016)	16.00	2.48	7.64	3.33	2.89	8.88	3.89
	CRL Pang et al. (2017)	0.26	2.32	3.68	2.36	2.48	3.59	2.67
	PSMnet Chang et al. (2018)	0.31	1.71	4.31	2.14	1.86	4.62	2.32
	GC-Net Kendall et al. (2017)	0.65	2.02	3.12	2.45	2.21	6.16	2.87
	iResNet Liang et al. (2018)	0.08	2.15	2.55	2.22	2.35	3.23	2.50
	HSM Yang et al. (2019)	0.12	1.63	3.40	1.92	1.80	3.85	2.14
	GA-Net-15 Zhang et al. (2019)	0.36	1.40	3.37	1.73	1.55	3.82	1.93
	DPruner_Best Duggal et al. (2019)	0.13	1.71	3.18	1.95	1.87	3.56	2.15
Real-time	StereoNet Khamis et al. (2018)	0.02	-	-	-	4.30	7.45	4.83
	MAD-Net Tonioni et al. (2019)	0.02	3.45	8.41	4.27	3.75	9.20	4.66
	DispNetC Mayer et al. (2016)	0.04	4.11	3.72	4.05	4.32	4.41	4.34
	DeepCostAggr Kuzmin et al. (2017)	0.03	4.82	10.11	5.69	5.34	11.35	6.34
	RTSNet Lee and Shin (2019)	0.02	2.67	5.83	3.19	2.86	6.19	3.41
	Ours	0.01	2.12	3.88	2.42	2.51	4.62	2.86

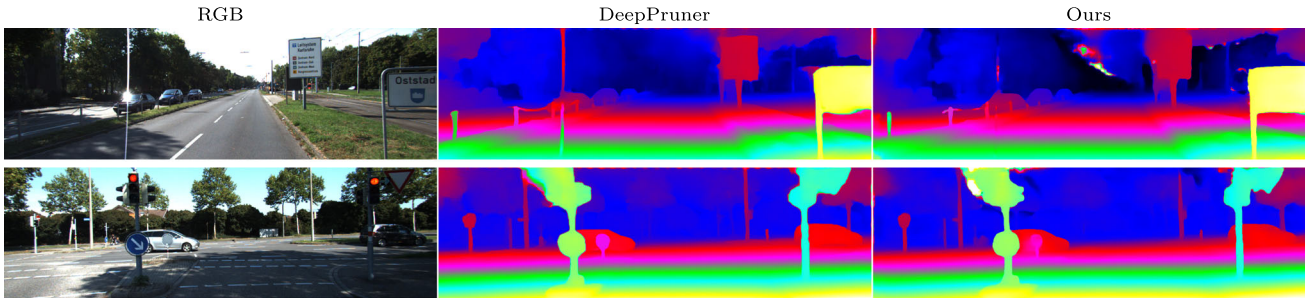


Fig. 6 Qualitative Results on KITTI 2015 *test* dataset. Our method achieves comparable performance with 13 times faster speed.

Table 8 Results on ETH3D *test* dataset. Bold indicates the best, while underline indicates the second best

Methods	time(s)	EPE	rmse	bad-4.0	bad-2.0	bad-1.0	A99
HSM Yang et al. (2019)	0.12	0.29	0.67	0.68	1.48	4.25	3.25
SDRNet Chabra et al. (2019)	0.13	0.34	0.71	0.50	1.66	6.02	3.07
iResNet Liang et al. (2018)	0.15	0.25	0.59	0.34	1.20	4.04	2.70
DPruner Duggal et al. (2019)	0.12	0.28	0.58	0.34	1.04	3.82	2.61
PSMnet Chang et al. (Chang et al., (2018))	0.31	0.36	0.75	0.54	1.31	5.41	3.38
DN-CSS Ilg et al. (Ilg et al., (2018))	0.05	0.24	0.56	0.38	0.96	3.00	2.89
Ours	0.01	0.32	0.63	0.53	1.25	4.82	2.79

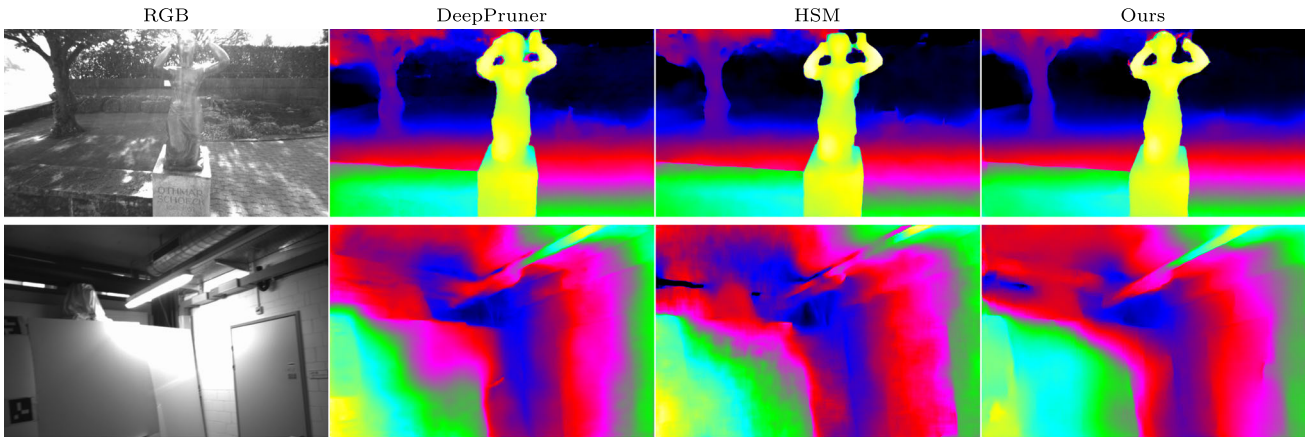


Fig. 7 Qualitative results on ETH3D *test* dataset. Our method achieves comparable performance with 12 times faster speed.

Table 9 Results on Middlebury 2014 *test* dataset. Bold indicates the best, while underline indicates the second best.

Methods	time(s)	EPE	rmse	bad-4.0	bad-2.0	bad-1.0	A99
SGM Hirschmuller (2008)	0.32	5.32	20.0	12.2	18.4	31.1	109
HSM Yang et al. (2019)	0.47	2.07	10.3	4.83	10.2	24.6	39.2
iResNet Liang et al. (2018)	0.28	3.31	11.3	12.6	22.9	38.8	48.6
DPruner Duggal et al. (2019)	0.11	4.80	14.7	15.9	30.1	52.3	67.7
PSMNet chang et al. (2018)	0.45	6.68	19.4	23.5	42.1	63.9	84.5
DN-CSS Ilg et al. (2018)	0.58	4.04	13.9	14.7	22.8	36.0	58.8
Ours	0.04	3.12	13.8	7.22	15.4	35.1	55.6

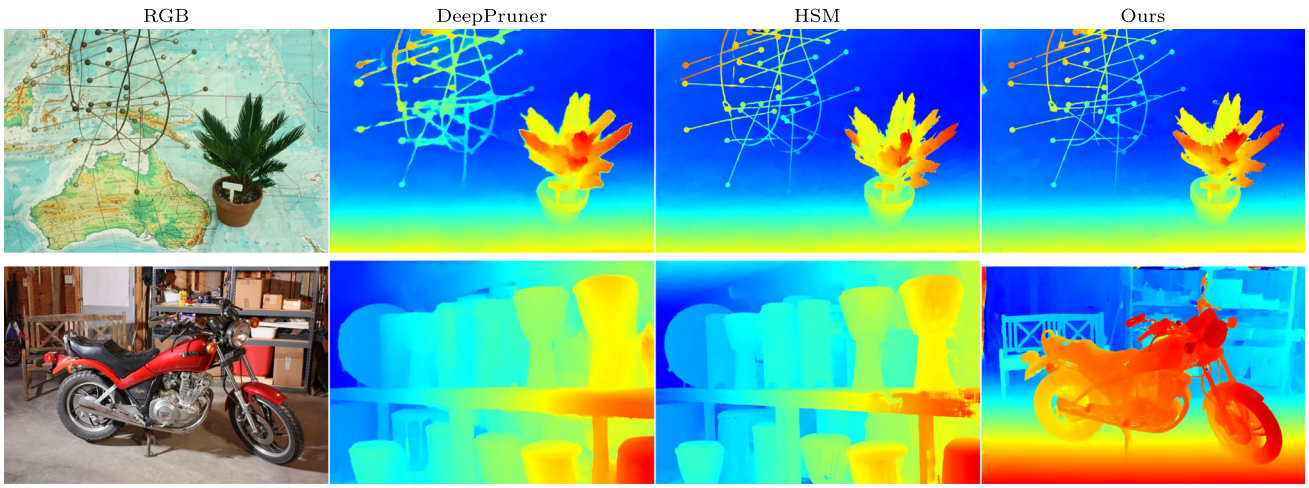


Fig. 8 Qualitative results on Middlebury 2014 *test* dataset. Our method achieves better performance with 3 times faster speed, especially for thin objects.

Table 10 Results on cross-dataset study. SceneFlow pre-trained models were tested on Middlebury 2014 and KITTI 2015 training sets *directly*. For iResNet Liang et al. (2018), for a fair comparison, we retrained their model on SceneFlow dataset with exactly the same data augmentation strategies as other methods, *i.e.*, random crop only. We also add SOTA cross-domain generalization methods for reference (marked with *).

Methods	Middlebury					KITTI 2015			
	bad-1.0	bad-2.0	bad-4.0	avgerr	rms	EPE	bad-1.0	bad-2.0	bad-3.0
*DSMNet Zhang et al. (2020)	–	21.8	–	–	–	–	–	–	6.50%
*MS-GCNet Cai et al. (2020)	-	18.5	–	–	–	–	–	–	6.21%
iResNet Liang et al. (2018)	60.7	40.3	21.6	7.74	23.3	2.49	48.26%	19.59%	11.23%
GA-Net Zhang et al. 2019)	59.9	38.1	19.8	4.94	13.6	1.70	42.35%	18.29%	10.77%
LEAStereo Chang et al. (2020)	42.6	25.5	15.1	8.10	23.9	1.97	43.36%	19.21%	11.11%
Baseline3D	53.0	33.7	20.1	13.5	34.4	1.92	46.76%	21.02%	12.03%
Ours	51.4	33.1	19.6	6.67	19.2	1.63	37.55%	17.54%	10.88%

4.4 Robustness and Generalizability

In the previous section, we showed that Baseline3D yields higher accuracy than Ours, while being significantly slower. In this section, we compared the generalizability of Ours and Baseline3D methods using the SceneFlow-pretrained models, by testing them on the Middlebury and KITTI 2015 training sets, without fine-tuning. Table 10 shows the quantitative results. For better comparison, we use SOTA volumetric method GA-Net Zhang et al. (2019) and direct regression method iResNet Liang et al. (2018) as the reference methods. Note that the generalizability is not only dependent on the architecture itself; the augmentation strategy also effects the generalizability as well. Therefore, for a fair comparison, we report the result of our retrained model of iResNet Liang et al. (2018) with exactly the same data augmentation strategy, *i.e.*, random crop only. Ours consistently achieves better cross-dataset performance on these two datasets. The results in Fig. 9 show that Baseline3D generates false boundaries on the road plane while Ours does not.

4.4.1 Random Dot Stereo

Random Dot Stereograms (RDSs) Julesz (1971) were introduced decades ago to evaluate depth perception in the human visual system. RDSs were key to establishing the fact that the human visual system is capable of estimating depth simply by matching patterns in the left and right images, without utilizing any monocular cues (lighting, color, texture, shape, familiar objects, and so forth). Similarly, we argue that artificial stereo algorithms should be able to process random dot stereograms, and that this ability provides key evidence to understand the actual behavior of the networks, *i.e.*, con-text mapping or matching.

In random dot stereo pairs, there is no semantic context. Therefore, methods that fail this test, but that otherwise do well on benchmarks, are mostly likely relying on monocular, semantic cues for inference. Such methods (including DispNet, and those that build upon DispNet Liang et al., 2018) will struggle to process images whose distribution varies sig-

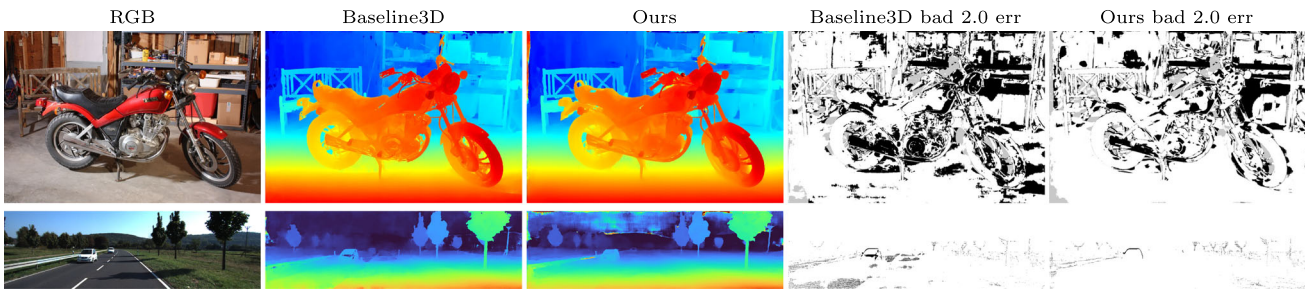


Fig. 9 Qualitative comparison on cross-dataset study. The SceneFlow pre-trained models were tested on Middlebury 2014 and KITTI 2015 training sets *without any finetuning*. Note that Ours generalizes better across datasets than Baseline3D.

Table 11 Quantitative Results on RDS dataset

Methods	time(s)	EPE	bad-1.0	bad-2.0	bad-3.0
DispNet Mayer et al. (2016)	0.07	60.42	99.89	99.77	99.69
Baseline3D	0.28	0.75	3.97	2.71	2.33
Ours	0.02	1.02	5.45	3.59	2.93

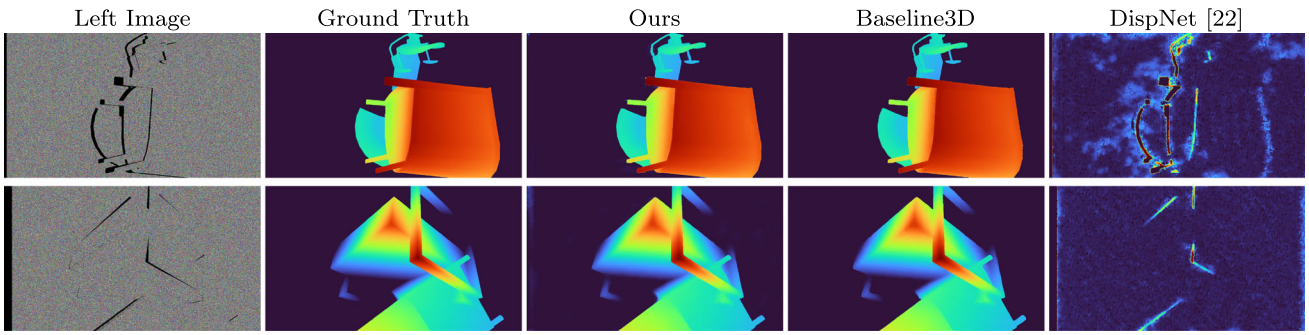


Fig. 10 Qualitative results on the Random Dot Stereo dataset. Our network is able to recover disparities from RDS images while DispNet Mayer et al. (2016) can not. The black holes in the left image is due to the occlusion between the left view and the cyclopean view Henkel (1997).

nificantly from the training distribution, because they do not actually match pixels between images.

To evaluate this claim, we created a Random Dot Stereogram (RDS) dataset. The dataset contains 2000 frames, 1800 for training and 200 for testing. We provide qualitative and quantitative results on the RDS dataset of Ours and DispNet Mayer et al. (2016). We fine-tuned Ours and DispNet Mayer et al. (2016) for 200 epochs with the ground truth. The quantitative results are shown in Table 11. Unlike DispNet, Ours successfully produces high-quality disparity maps, as shown in Fig. 10. This is because these 2D convolution-based methods rely more on context information while our method is trying to find corresponding points with *matching*.

4.5 Discussion

Ours *vs.* R-MVSNet Yao et al. (2019) The design philosophy between our DICC and R-MVSNet Yao et al. (2019) is different, mainly because of the differences between these two tasks. For a MVS task, since there are multiple frames (> 3), the displacements are typically smaller than stereo matching. In this case, with hand-craft costs plus GRU cost aggregation Teed and Deng (2020), it can still find most matching points. When it comes to the stereo matching problem, this strategy does not work well. Also, the primary purpose of R-MVSNet Yao et al. (2019) is to handle memory consumption problems in the MVS task, so they can use GRU to do cost aggregation. On the other hand, ours is focusing on the speed, so we use CNN to learn how to *compute* and aggregate the matching cost.

5 Conclusion

In this paper, we have proposed a highly efficient deep stereo matching network. Our method is not only on par with the state-of-the-art deep stereo matching methods in terms of accuracy, but is also able to run in super real-time, *i.e.*, over 100 FPS on typical image sizes. This makes our method suitable for time-critical applications such as robotics and autonomous driving. The key to our success is Displacement-Invariant Cost Computation, where 2D convolutions based

cost computation is independently applied to all disparity levels to construct a 3D cost volume.

Acknowledgements We would like to thank Nikolai Smolyanskiy for helpful discussions. Y. Dai was supported in part by the Natural Science Foundation of China grants (61871325 and 61671387) and the National Key Research and Development Program of China (2018AAA0102803).

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

References

Badki, A., Gallo, O., Kautz, J., Sen, P. (2021). Binary TTC: A temporal geofence for autonomous navigation. In: CVPR.

Badki, A., Troccoli, A., Kim, K., Kautz, J., Sen, P., Gallo, O. (2020). Bi3d: Stereo depth estimation via binary classifications. In: CVPR.

Cai, C., Mordohai, P. (2020). Do end-to-end stereo algorithms underutilize information? In: 3DV.

Cai, C., Poggi, M., Mattoccia, S., Mordohai, P. (2020). Matching-space stereo networks for cross-domain generalization. In: 3DV.

Chabra, R., Straub, J., Sweeney, C., Newcombe, R., Fuchs, H. (2019). StereoDRNet: Dilated residual StereoNet. In: CVPR.

Chang, J.R., Chen, Y.S. (2018). Pyramid stereo matching network. In: CVPR.

Cheng, X., Zhong, Y., Harandi, M., Dai, Y., Chang, X., Li, H., Drummond, T., Ge, Z. (2020). Hierarchical neural architecture search for deep stereo matching. In: NeurIPS.

Duggal, S., Wang, S., Ma, W.C., Hu, R., Urtasun, R. (2019). Deeppruner: Learning efficient stereo matching via differentiable patchmatch. In: ICCV.

He, K., Zhang, X., Ren, S., Sun, J. (2014). Spatial pyramid pooling in deep convolutional networks for visual recognition. In: ECCV.

Henkel, R.D. (1997). Constructing the cyclopean view. In: ICANN.

Hernandez-Juarez, D., Chacón, A., Espinosa, A., Vázquez, D., Moure, J., & López, A. (2016). Embedded real-time stereo estimation via semi-global matching on the GPU. *Procedia Computer Science, 80,* 143–153.

Hirschmuller, H. (2008). *Stereo processing by semiglobal matching and mutual information*. Intell: IEEE Trans. Pattern Anal. Mach.

Ilg, E., Saikia, T., Keuper, M., Brox, T. (2018). Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In: ECCV.

Janai, J., Güney, F., Behl, A., Geiger, A. (2017). Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. Arxiv.

Julesz, B. (1971). *Foundations of Cyclopean Perception*. Chicago: The University of Chicago Press.

Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., Bry, A. (2017). End-to-end learning of geometry and context for deep stereo regression. In: ICCV.

Khamis, S., Fanello, S., Rhemann, C., Kowdle, A., Valentin, J., Izadi, S. (2018). StereoNet: Guided hierarchical refinement for real-time edge-aware depth prediction. In: ECCV.

Kim, S., Kim, S., Min, D., Sohn, K. (2019). Laf-net: Locally adaptive fusion networks for stereo confidence estimation. In: CVPR.

Kuzmin, A., Mikushin, D., Lempitsky, V. (2017). End-to-end learning of cost-volume aggregation for real-time dense stereo. In: MLSP.

Lee, H., Shin, Y. (2019). Real-time stereo matching network with high accuracy. In: ICIP.

Liang, Z., Feng, Y., Guo, Y., Liu, H., Chen, W., Qiao, L., Zhou, L., Zhang, J. (2018). Learning for disparity estimation through feature constancy. In: CVPR.

Mayer, N., Ilg, E., Häusser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T. (2016). A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: CVPR.

Menze, M., Geiger, A. (2015). Object scene flow for autonomous vehicles. In: CVPR.

Pang, J., Sun, W., Ren, J.S., Yang, C., Yan, Q. (2017). Cascade residual learning: A two-stage convolutional neural network for stereo matching. In: ICCVW.

Poggi, M., Aleotti, F., Tosi, F., Mattoccia, S. (2020). On the uncertainty of self-supervised monocular depth estimation. In: CVPR.

Ronneberger, O., Fischer, P., Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In: Medical Image Computing and Computer-Assisted Intervention.

Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nešić, N., Wang, X., Westling, P. (2014). High-resolution stereo datasets with subpixel-accurate ground truth. In: Pattern Recognition.

Scharstein, D., Szeliski, R. (1998). Stereo matching with nonlinear diffusion. In: IJCV.

Scharstein, D., Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. In: IJCV.

Schöps, T., Schönberger, J.L., Galliani, S., Sattler, T., Schindler, K., Pollefeys, M., Geiger, A. (2017). A multi-view stereo benchmark with high-resolution images and multi-camera videos. In: CVPR.

Seki, A., Pollefeys, M. (2017). SGM-Nets: Semi-global matching with neural networks. In: CVPR.

Smolyanskiy, N., Kamenev, A., Birchfield, S. (2018). On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach. In: CVPRW.

Taniai, T., Matsushita, Y., Sato, Y., Naemura, T. (2018). Continuous 3D Label Stereo Matching using Local Expansion Moves. In: TPAMI.

Teed, Z., Deng, J. (2020). Raft: Recurrent all-pairs field transforms for optical flow. In: ECCV.

Tonioni, A., Tosi, F., Poggi, M., Mattoccia, S., Di Stefano, L. (2019). Real-time self-adaptive deep stereo. In: CVPR.

Weber, M., Humenberger, M., Kubinger, W. (2009). A very fast census-based stereo matching implementation on a graphics processing unit. In: ICCVW.

Yang, G., Manela, J., Happold, M., Ramanan, D. (2019). Hierarchical deep stereo matching on high-resolution images. In: CVPR.

Yao, Y., Luo, Z., Li, S., Shen, T., Fang, T., Quan, L. (2019). Recurrent mvsnet for high-resolution multi-view stereo depth inference. In: CVPR.

Yin, Z., Darrell, T., Yu, F. (2019). Hierarchical discrete distribution decomposition for match density estimation. In: CVPR.

Žbontar, J., & LeCun, Y. (2016). Stereo matching by training a convolutional neural network to compare image patches. *J Mach Learn Res, 17*(1), 2287–2318.

Zhang, C., Li, Z., Cheng, Y., Cai, R., Chao, H., Rui, Y. (2015). Meshstereo: A global stereo model with mesh alignment regularization for view interpolation. In: ICCV.

Zhang, F., Prisacariu, V., Yang, R., Torr, P.H. (2019). GA-Net: Guided aggregation net for end-to-end stereo matching. In: CVPR.
Zhang, F., Qi, X., Yang, R., Prisacariu, V., Wah, B., Torr, P.H.S. (2020). Domain-invariant stereo matching networks. In: ECCV.
Zhong, Y., Dai, Y., Li, H. (2017). Self-supervised learning for stereo matching with self-improving ability. In: arXiv:1709.00930.
Zhong, Y., Li, H., Dai, Y. (2018). Open-world stereo video matching with deep RNN. In: ECCV.
Zhou, H., Ummenhofer, B., Brox, T. (2018). Deeptam: Deep tracking and mapping. In: ECCV.