Parameterized Cost Volume for Stereo Matching

Jiaxi Zeng[1]†, Chengtang Yao[1,3]†, Lidong Yu[3], Yuwei Wu[1]*, Yunde Jia[2],
[1]Beijing Key Laboratory of Intelligent Information Technology,
School of Computer Science & Technology, Beijing Institute of Technology, China
[2]Guangdong Laboratory of Machine Perception and Intelligent Computing,
Shenzhen MSU-BIT University, China
[3]Autonomous Driving Algorithm, Deeproute
{jiaxi,yao.c.t,wuyuwei,jiayunde}@bit.edu.cn
yvlidong@gmail.com

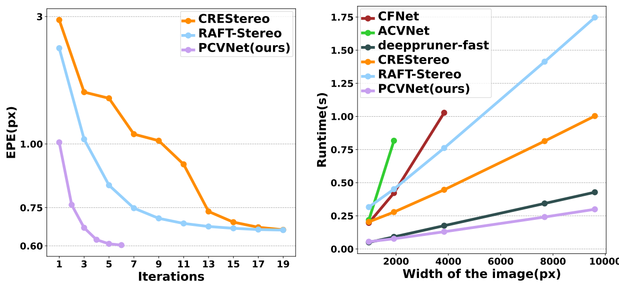
Abstract

Stereo matching becomes computationally challenging when dealing with a large disparity range. Prior methods mainly alleviate the computation through dynamic cost volume by focusing on a local disparity space, but it requires many iterations to get close to the ground truth due to the lack of a global view. We find that the dynamic cost volume approximately encodes the disparity space as a single Gaussian distribution with a fixed and small variance at each iteration, which results in an inadequate global view over disparity space and a small update step at every iteration. In this paper, we propose a parameterized cost volume to encode the entire disparity space using multi-Gaussian distribution. The disparity distribution of each pixel is parameterized by weights, means, and variances. The means and variances are used to sample disparity candidates for cost computation, while the weights and means are used to calculate the disparity output. The above parameters are computed through a JS-divergence-based optimization, which is realized as a gradient descent update in a feed-forward differential module. Experiments show that our method speeds up the runtime of RAFT-Stereo by 4 $\sim$ 15 times, achieving real-time performance and comparable accuracy. The code is available at `https://github.com/jiaxiZeng/Parameterized-Cost-Volume-for-Stereo-Matching`.

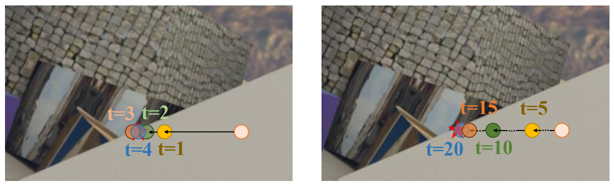
1. Introduction

Stereo matching aims to find the pixel-wise correspondence between paired images within a predefined disparity range. The pixel-wise matching is efficiently realized by existing methods within a small predefined range [1, 26, 30, 6, 5, 20, 27]. However, the matching becomes time- and memory-consuming for a large predefined range, as computational cost increases rapidly with the growing range. This challenge limits the application of stereo matching in the real world, e.g., in high-resolution images.

Prior methods relieve the challenge mainly relying on



(a) The illustration of End-Point-Error (EPE) changing with the increase of iteration on the Sceneflow dataset (left) and the illustration of time cost changing with the increase in image width (right).



Moving Step of Ours   Moving Step of RAFT-Stereo

(b) The visualization of the moving step at each iteration t.

Figure 1: (a) We analyze the EPE at different iterations and time cost with the increase of image width. (b) The moving steps are visualized on the right view image, starting from a disparity value of 0 towards the ground truth. Compared to RAFT-Stereo [13] and CREStereo [11], our method achieves a rapid decrease in EPE and a large moving step from the first iteration, greatly reducing the number of iterations.

†These authors contributed equally to this work.
*Corresponding author.

dynamic cost volume [3, 5, 2, 20, 13, 11]. The dynamic cost volume samples disparity candidates in a local disparity space instead of the entire disparity space. It reduces significant memory costs but requires many iterations to get close to the ground truth, as illustrated in Figure 1a. In this paper, we prove that the dynamic cost volume encodes the disparity space as a single Gaussian distribution with a fixed and small variance. The fixed and small variance results in a limited view over the entire disparity space. Thus, these methods are hard to produce a large update to quickly converge to the ground truth when the initialized disparity is far from the ground truth. Instead, a multi-Gaussian distribution provides a global view after initializing each Gaussian distribution uniformly in the disparity space. The global view allows fast convergence to the vicinity of ground truth at the beginning of the iteration, as shown in Figure 1. Meanwhile, the local view is kept for subsequent fine-grained matching as the learnable parameters (i.e., variances) of multi-Gaussian distribution become small at the convergence stage.

To this end, we propose a parameterized cost volume that encodes the entire disparity space using multi-Gaussian distribution. The disparity distribution of each pixel is parameterized by weights, means, and variances. The computation of these parameters is formulated as a JS-divergence-based optimization problem. We solve the problem through iterative gradient descent updates in a feed-forward differential module, including the following steps. (1) We use the initialized means and variances to sample disparity candidates. (2) The sampled disparity candidates are used to compute the matching cost, which is subsequently used to predict an optimization step. (3) We use the predicted step to update the three kinds of parameters. The updated parameters serve as initialization for the next iteration, while the weighted average of the mean values (i.e., the expectation of multi-Gaussian distribution) is regarded as the disparity output at the current iteration. The above feed-forward optimization is prone to local oscillations at the final convergence stage. Thus, we further design an uncertainty-aware refinement module to localize and correct the incorrect results at the last iteration. We compute uncertainty from weights and variances to measure the reliability of predicted disparity. We then use uncertainty as guidance to propagate disparity from high-reliability areas to low-reliability areas.

We validate our method on both synthetic and real-world datasets. Results show that our method simultaneously achieves real-time performance and SOTA-comparable accuracy on most datasets. We also compare our method with SOTA methods in terms of time growth when the predefined disparity range is enlarged. Experiments show that our method maintains fixed memory costs and requires fewer iterations to achieve comparable or even better accuracy than other methods.

2. Related Work

2.1. Deep Stereo Matching

Stereo matching has been studied for decades [8, 23, 7, 4, 32]. Recently, deep-learning-based methods have shown promising results on all datasets [16, 17, 19]. These methods rely on cost volume, which can be roughly classified as dense volume methods and sparse volume methods.

As one of the dense volumes, the 4D volume [9, 1, 3, 20, 27] is constructed by concatenating the left and right feature for every disparity in the predefined disparity range. They achieve impressive accuracy but require an enormous computational cost due to the heavy 3D convolutions on 4D volume. Some methods construct the 4D volume at a very low-resolution [10, 30] to reduce computation cost. However, the computation grows at the fourth power with the increase of the 4D volume's size. In order to reduce the growth rate, 3D volume is proposed as the base of a lightweight network [16, 14, 12, 29, 26, 24]. The 3D volume is built by computing the correlation between the left and right features within a predefined disparity range. Although they have reduced the computation cost by an order of magnitude, their cost volumes are redundant due to the sparsity of disparity space.

A lot of sparse volume methods are proposed to reduce the computation from the perspective of sparsity [3, 5, 2, 20, 15, 13, 11, 31]. Among them, the dynamic cost volume is accepted by most researchers. The dynamic cost volume reduces huge redundancy in disparity space, thus resulting in low memory cost. However, they are still either time- or memory-consuming when dealing with high-resolution images. Instead, our parameterized cost volume uses a limited number of parameters to encode the entire disparity space achieving both low memory cost and real-time speed.

2.2. Dynamic Cost Volume based Stereo Matching

The methods based on dynamic cost volume [3, 5, 2, 20, 15, 13, 11] only sample a small number of disparity candidates to construct the cost volume, including partially-dense methods and pure-sparse methods. The partially-dense methods [5, 2, 20, 15] first conduct dense matching at the lowest resolution to obtain the initial disparity map. They then use the initial disparity map to predict the sampling range at a higher resolution for dynamic cost volume construction. Among them, CasNet [5] uses a hyperparameter to control the sampling range. ATV-Net [2], CFNet [20], and UASNet [15] propose to use the variance of cost volume along the disparity/depth dimension to predict the sampling range. The partially-dense methods take great advantage of stereo matching's sparsity in disparity space but still rely on dense cost volume for initialization. The complexity of such an initialization grows rapidly with the increase of the predefined disparity range. Later, pure-sparse

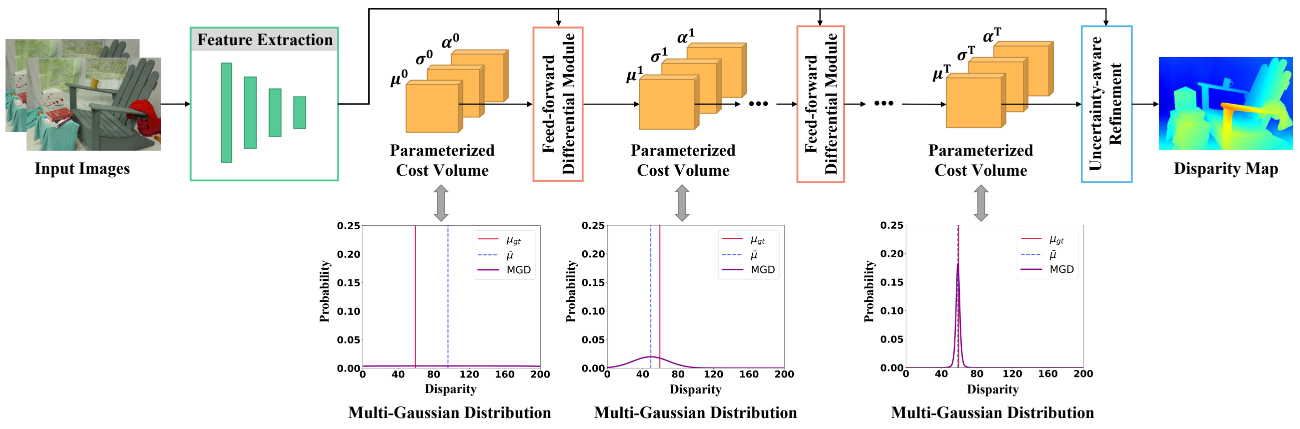

Input Images
Feature Extraction
μ^0 σ^0 α^0
Parameterized Cost Volume
Feed-forward Differential Module
μ^1 σ^1 α^1
Parameterized Cost Volume
Feed-forward Differential Module
μ^T σ^T α^T
Parameterized Cost Volume
Uncertainty-aware Refinement
Disparity Map
Probability
Disparity
μ_{gt}
$\bar{\mu}$
MGD
Multi-Gaussian Distribution
Multi-Gaussian Distribution
Multi-Gaussian Distribution

Figure 2: The pipeline of our method. We first extract the multi-level features from the left and right images. The features and the initial parameterized cost volume are input into a feed-forward differential module to update multi-Gaussian parameters iteratively (means $\boldsymbol{\mu}$, variances $\boldsymbol{\sigma}$, weights $\boldsymbol{\alpha}$). The multi-Gaussian distribution (MGD) gradually converges to the ground truth (symbolized as μ_{gt}) as the iteration goes on. The expectation of MGD (i.e., the weighted sum of means denoted as $\bar{\mu}$) is regarded as the disparity output. The disparity map is further refined through an uncertainty-aware refinement module.

methods [25, 13, 11] are proposed to initialize the disparity as the value of zero without dense matching at a low resolution. These methods update the disparity map in an iteration fashion to gradually converge to the ground truth. At each iteration, an offset is predicted from a dynamic cost volume where disparity candidates are sampled from the existing disparity map with a fixed range. RAFT-Stereo [25, 13] introduce a multi-level GRU to collect context information for fast convergency. CREStereo [11] further introduce a cascade architecture that estimates the disparity map from coarse to fine. The pure-sparse methods significantly reduce the memory cost but at the sacrifice of time speed as they require many iterations to get close to the ground truth when the initialized disparity is far from the ground truth.

Different from the above methods, our parameterized cost volume encodes the entire disparity space using multi-Gaussian distribution. After initializing multi-Gaussian distribution uniformly in the disparity space, a large global view is obtained for quick convergence to get close to the ground truth. At the convergence stage, multi-Gaussian variances become small, allowing for a fine-grained matching in a local view. Thus, our method is able to reduce the number of iterations and speed up the running time without sacrificing the matching accuracy.

3. Method

In this section, we first introduce the parameterized cost volume from the perspective of formulation and optimization. We then present the detailed architecture of our feed-forward differential module and uncertainty-aware refinement module. The whole pipeline is shown in Figure 2.

3.1. Parameterized Cost Volume

Formulation Cost volume $C(x) = c_x^d$ represents the matching cost c_x^d between pixel x on the left image and pixel $x - d$ on the right image, where d is the disparity. Prior methods densely enumerate the disparity candidates d from a discrete disparity distribution $\{0, 1, ..., D-1\}$, which is memory-consuming and redundant. Alternatively, dynamic-cost-volume-based methods only sample from the neighbors of an initialized disparity $d \in N_{\tilde{d}}$, where N denotes neighbors and $\tilde{d}$ is the initialized disparity. In this paper, we propose a parameterized cost volume that uses multi-Gaussian distribution to encode the entire disparity space:

$$\begin{aligned} C(x, \theta) &= \{c_x^{d(\theta)}\}, \\ d(\theta) &\sim \sum_{i=1}^{i=M} \alpha_i \mathcal{N}(\mu_i, \sigma_i^2). \end{aligned} \tag{1}$$

$\theta = \{\alpha_i, \mu_i, \sigma_i\}_{i=1}^{M}$ is the parameter set of multi-Gaussian distributions, including weights $\boldsymbol{\alpha} = \{\alpha_i\}_{i=1}^{M}$, means $\boldsymbol{\mu} = \{\mu_i\}_{i=1}^{M}$, and standard deviations $\boldsymbol{\sigma} = \{\sigma_i\}_{i=1}^{M}$ (we colloquially use 'variance' to refer to σ which is actually the standard deviation). M is the number of Gaussian distributions. $\sim$ represents the sampling from a distribution and α_i is constrained by $\sum_{i=1}^{M} \alpha_i = 1$. Through the parameterized representation of cost volume, we not only maintain the efficiency of dynamic cost volume by sparse sampling from the multi-Gaussian distribution but also acquire an adequate global view by uniformly initializing the multi-Gaussian distribution in the entire disparity space.

Optimization In order to effectively learn the parameters θ of multi-Gaussian distribution, we take the ground-truth disparity as a predefined Gaussian distribution

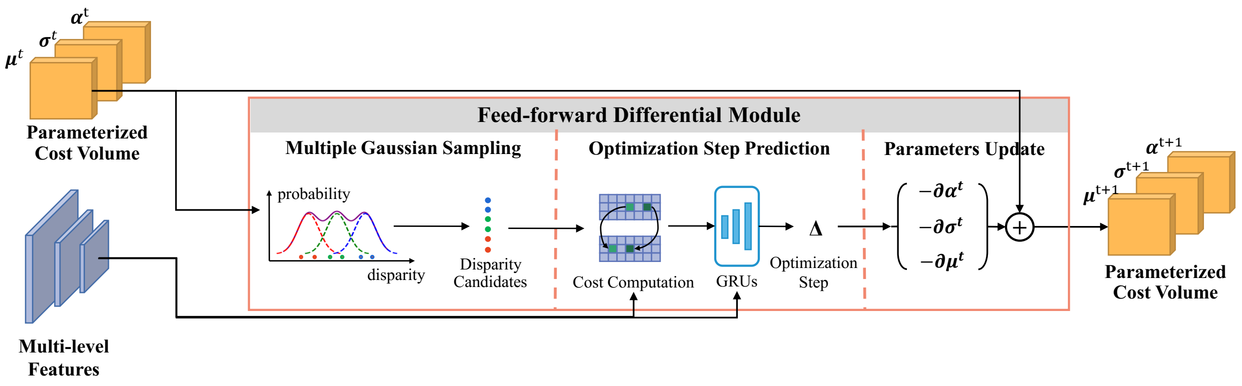


Figure 3: The overview of the feed-forward differential module. The module takes multi-level features and parameterized cost volume (means $\boldsymbol{\mu}$, variances $\boldsymbol{\sigma}$, weights $\boldsymbol{\alpha}$) from the last iteration t as input. It first samples disparity candidates from each Gaussian distribution. It then uses the candidates to compute cost. The cost is further input into GRU with multi-level features to predict the optimization step. The optimization step is used to update parameterized cost volume through the gradient of mean $\partial\boldsymbol{\mu}$, variance $\partial\boldsymbol{\sigma}$, and weight $\partial\boldsymbol{\alpha}$.

$\mathcal{N}(\mu_{gt}, \sigma_{gt})$ and design a JS-divergence-based optimization to force the multi-Gaussian distribution close to it. μ_{gt} is the ground-truth disparity and σ_{gt} is a predefined parameter. The objective function is presented as

$$min\ \frac{1}{2}(F(\mathcal{N}_{gt}||\sum_{i=1}^{i=M}\alpha_i\mathcal{N}_i) + F(\sum_{i=1}^{i=M}\alpha_i\mathcal{N}_i||\mathcal{N}_{gt})), \quad s.t.\ \sum_{i=1}^{i=M}\alpha_i = 1, \tag{2}$$

where $F(P||Q) = \sum_{d\in\mathcal{D}} P(d)log\frac{P(d)}{Q(d)}$ is KL divergence, and $\mathcal{N}_i$ is a short form of $\mathcal{N}(\mu_i, \sigma_i^2)$.

We then derive the optimization into feed-forward gradient decent updates by defining a Lagrangian function L:

$$L = \frac{1}{2}(F(\mathcal{N}_{gt}||\sum_{i=1}^{i=M}\alpha_i\mathcal{N}_i) + F(\sum_{i=1}^{i=M}\alpha_i\mathcal{N}_i||\mathcal{N}_{gt})) + \lambda(\sum_{i=1}^{i=M}\alpha_i - 1), \tag{3}$$

where λ is a Lagrange multiplier. Before proceeding with the derivation, we introduce two lemmas whose specific proofs are provided in our supplementary materials.

Lemma 1 *Given two Gaussian distribution $\mathcal{N}_p$ and $\mathcal{N}_q$, the KL divergence $F(\mathcal{N}_p||\mathcal{N}_q)$ is*

$$F(\mathcal{N}_p||\mathcal{N}_q) = log\frac{\sigma_q}{\sigma_p} + \frac{\sigma_p^2 + (\mu_p - \mu_q)^2}{2\sigma_q^2} - \frac{1}{2}. \tag{4}$$

Lemma 2 *Given two multi-Gaussian distribution $P = \sum_{i=1}^{i=M}\alpha_i^p\mathcal{N}_i^p$ and $Q = \sum_{i=1}^{i=M}\alpha_i^q\mathcal{N}_i^q$, the compact upper bound of KL divergence $F(P||Q)$ is*

$$F(P||Q) \leq \sum_{i=1}^{i=M}F(\alpha_i^p||\alpha_i^q) + \sum_{i=1}^{i=M}\alpha_i^pF(\mathcal{N}_i^p||\mathcal{N}_i^q). \tag{5}$$

Based on the lemma 2, we obtain the upper bound of L through Eq 3 and 6:

$$L \leq \frac{1}{2}(\sum_{i=1}^{i=M}F(\frac{1}{M}||\alpha_i) + \sum_{i=1}^{i=M}\frac{1}{M}F(\mathcal{N}_{gt}||\mathcal{N}_i) + \sum_{i=1}^{i=M}F(\alpha_i||\frac{1}{M}) + \sum_{i=1}^{i=M}\alpha_iF(\mathcal{N}_i||\mathcal{N}_{gt})) + \lambda(\sum_{i=1}^{i=M}\alpha_i - 1), \tag{6}$$

We then turn to minimize the upper bound of L. The gradient of parameters α_i, μ_i, σ_i is derived from Eq 6 and 4 as follows:

$$\begin{aligned}
\partial\sigma_i &= \frac{1}{2}(\frac{\sigma_i^2 - \sigma_{gt}^2 - \Delta^2}{M\sigma_i^3} - \frac{\alpha_i}{\sigma_i} + \frac{\alpha_i\sigma_i}{\sigma_{gt}^2}),\\
\partial\mu_i &= -\frac{\Delta}{2}(\frac{1}{M\sigma_i^2} + \frac{\alpha_i}{\sigma_{gt}^2}),\\
\partial\alpha_i &= \beta_i + \lambda,\\
\beta_i &= \frac{1}{2}(-\frac{1}{M\alpha_i} + log\frac{\sigma_{gt}M\alpha_i}{\sigma_i} + \frac{\sigma_i^2 + \Delta^2}{2\sigma_{gt}^2} + \frac{1}{2}),\\
\lambda &= -\frac{1}{M}\sum_{i=1}^{i=M}\beta_i,\\
\Delta &= \mu_{gt} - \mu_i.
\end{aligned} \tag{7}$$

As μ_{gt} is not available during inference, we approximate the optimization step Δ through a network similar to RAFT-Stereo [13]. The updates to α_i, μ_i, σ_i at iteration t is then set as

$$\begin{aligned}
\sigma_i^{t+1} &= \sigma_i^t - \partial\sigma_i^t,\\
\mu_i^{t+1} &= \mu_i^t - \partial\mu_i^t,\\
\alpha_i^{t+1} &= \alpha_i^t - \partial\alpha_i^t.
\end{aligned} \tag{8}$$

Dynamic-cost-volume-based methods such as RAFT-Stereo predict the step size to move towards the ground truth at each iteration. The step size is added to the disparity from the previous iteration to update the result. According to the Eq. 7 and 8, these methods are equivalent to approximating a single Gaussian distribution with a fixed variance of $\sigma^2 = \sigma_{gt}^2/(2\sigma_{gt}^2 - 1)$. Due to the limited view of a single Gaussian distribution, such methods are difficult to capture a global view in a large disparity range. In contrast, our method employs multi-Gaussian distribution to enlarge the view size and facilitate information interaction between the different distributions during optimization, resulting in efficient convergence to the ground truth.

3.2. Feed-forward Differential Module

We design a feed-forward differential module to implement the JS-based optimization. As shown in Figure 3, the module first samples the disparity candidates from the current multi-Gaussian distribution. Subsequently, matching costs are computed based on these disparity candidates, and the module utilizes multi-level GRUs [13] to predict the optimization step. Finally, the optimization step is used to calculate the gradients for the parameters update.

Multiple Gaussian Sampling We sample the disparity candidates from the current multi-Gaussian distribution, with each Gaussian distribution independently sampled. Specifically, for the i-th Gaussian distribution, the candidates are uniformly sampled within the range $[\mu_i - 3\sigma_i, \mu_i + 3\sigma_i]$.

Optimization Step Prediction We compute the matching cost through correlation according to the disparity candidates. Firstly, the cost from different Gaussian distributions is independently encoded by several 2D convolutional layers whose weights are shared among the distributions. Subsequently, the encoded costs, along with the means $\boldsymbol{\mu}$, variances $\boldsymbol{\sigma}$, and weights $\boldsymbol{\alpha}$, are concatenated as input to the multi-level GRUs [13]. The hidden state of the highest-level GRU is used to predict the optimization step Δ via a two-layer convolution.

Parameters Update We use the gradient descent algorithm to update the parameters. Due to the numerical instability of the gradients in Eq. 7, we apply gradient clipping before the update. The clipped gradients are then employed for parameter updates, as specified in Eq. 8. To restrict the updated $\boldsymbol{\alpha}$ between 0 and 1, we perform clipping and normalization on $\boldsymbol{\alpha}$ as follows:

$$\hat{\alpha}_i^{t+1} = \frac{\min(\max(\alpha_i^{t+1}, 0), 1)}{\sum_i \min(\max(\alpha_i^{t+1}, 0), 1)}. \tag{9}$$

The updated parameters are used for the next iteration. Subsequently, we compute the expectation of the multi-Gaussian distribution as the disparity output using the following equation:

$$\bar{\mu}^{t+1} = \sum_{i=1}^{M} \hat{\alpha}_i^{t+1} \mu_i^{t+1}. \tag{10}$$

3.3. Uncertainty-aware Refinement Module

As the feed-forward optimization is prone to local oscillations at the convergence stage, we design an uncertainty-aware refinement module to improve the disparity of detailed areas. We first input weights $\boldsymbol{\alpha}$, variances $\boldsymbol{\sigma}$, and means $\boldsymbol{\mu}$ into a series of convolutions following a sigmoid function to estimate an uncertainty map U. Afterward, we concatenate the uncertainty map with the disparity map and left features to predict a residual map R through convolutions where a leaky-relu function is used at each layer except the last one. Finally, we use the uncertainty map U to guide the fusion of residual map R and disparity map $\bar{\mu}$ as follows:

$$\hat{\mu} = \bar{\mu} + R \cdot U. \tag{11}$$

For detailed architecture, please refer to our supplementary materials.

3.4. Loss

We design a loss function to enforce the multi-Gaussian distribution to be close to the ground truth distribution. Given the ground truth disparity μ_{gt}, we use the L1 loss function to minimize the differences between the mean of each Gaussian distribution μ_i^t and ground truth μ_{gt} for each iteration t:

$$\mathcal{L}_m^t = \sum_{i=1}^{M} \|\mu_i^t - \mu_{gt}\|_1, \tag{12}$$

The L1 loss function is also applied to the output disparity map $\bar{\mu}^t$ at each iteration t:

$$\mathcal{L}_d^t = \|\bar{\mu}^t - \mu_{gt}\|_1. \tag{13}$$

Besides, the refinement of disparity $\hat{\mu}$ is supervised as follows:

$$\mathcal{L}_r = \|\hat{\mu} - \mu_{gt}\|_1. \tag{14}$$

The final loss is the weighted combination of $\mathcal{L}_m^t$, $\mathcal{L}_d^t$, and $\mathcal{L}_r$:

$$\mathcal{L} = \sum_{t=1}^{T} \gamma^t (\mathcal{L}_m^t + \mathcal{L}_d^t) + \lambda \mathcal{L}_r, \tag{15}$$

where γ^t and λ are the balancing scalars.

4. Experiments

4.1. Datasets

SceneFlow SceneFlow [16] is a synthetic dataset consisting of 35454 image pairs for training and 4370 for testing. It contains three sub-datasets covering indoor and outdoor scenes. The resolution of the images is 540×960.

Model	Multi-Gaussian	Adaptive variance	Uncertainty-aware refinement	EPE (px)	1px-error rate (%)	3px-error rate (%)
$SGFV$				0.79	8.96	5.33
$SGAV$		✓		0.82	10.29	6.05
$MGFV$	✓			0.74	8.93	5.12
PCV	✓	✓		0.72	8.16	4.89
$PCV + Refine$	✓	✓	✓	0.71	7.98	4.79

Table 1: Ablation study on SceneFlow dataset. We present the results of all methods at the fourth iteration. $SGFV$ is a single Gaussian distribution with a fixed variance. $SGAV$ represents the single Gaussian distribution with the adaptive variance. $MGFV$ denotes multi-Gaussian with a fixed variance. PCV is our parameterized cost volume. $Refine$ is our uncertainty-aware refinement. All of the models in the ablation study are trained for 50k steps with a batch size of 48 on 8 NVIDIA A100 GPUs.

KITTI 2015 KITTI 2015 [17] is a real-world dataset for autonomous driving scenes. It contains 200 training and 200 testing image pairs with a resolution of 376×1240.

Middlebury Middlebury [19] is a real-world indoor dataset including 33 high-resolution image pairs. The maximum image size is 1896×3000. It collects image pairs from the same scene under different lighting, exposure, and calibration conditions to evaluate the robustness of the methods.

Booster Booster [18] is a novel dataset with high-resolution image pairs. Different from the Middlebury dataset, the Booster dataset provides more (228 for training and 191 for testing) labeled and higher-resolution (3008×4112) image pairs. It also includes more challenging cases in stereo matching, such as non-Lambertian surfaces.

4.2. Implementation Details

Our model is built based on RAFT-Stereo [13]. We use $M = 4$ Gaussian distributions and initialize our parameterized cost volume with $\mu \in \{0, 64, 128, 192\}$, $\sigma = 32$, and $\alpha = 0.25$. We set iteration number T as 6 during training and 4 during inference. As aforementioned in Section 3.1, we transform the ground truth into a predefined Gaussian distribution where μ_{gt} is the ground-truth disparity, and σ_{gt} is set to 2. We then set $\gamma^t = 0.2 + 0.2t$ and $\lambda = 1.4$ for the loss function in Eq. 15. We use AdamW optimizer and one-cycle learning rate schedule [22] where the maximum learning rate is set to 0.0002.

We pre-train our model with a batch size of 16 for 200k steps on the SceneFlow dataset. For the KITTI dataset, we combine the KITTI2015 and KITTI2012 to fine-tune the Sceneflow pre-trained model for 5k steps with a batch size of 12. For the Middlebury dataset, we fine-tune the pre-trained model for 4k steps with a batch size of 4. For the Booster dataset, we employ the cascade inference strategy proposed in CREStereo [11] to enlarge the receptive field for high-resolution inputs. Specifically, we down-sample the images by 0.25 and feed them to the model. After 4 iterations, we up-sample the parameterized cost volume to initialize another 4 iterations at the original resolution. The learning rate for fine-tuning is set to 0.00001.

We apply the data augmentation used in RAFT-Stereo [13] to all experiments. The slow-fast GRU [13] is used in our implementation. All of the experiments except for the ablation study are conducted on 2 NVIDIA GeForce RTX 3090 GPUs.

4.3. Ablation Study and Analysis

Parametrized Cost Volume We illustrate the effectiveness of our parameterized cost volume by comparing it with three kinds of variants: a single Gaussian distribution with a fixed variance ($SGFV$), a single Gaussian distribution with an adaptive variance ($SGAV$), and a multi-Gaussian distribution with a fixed variance ($MGFV$). As aforementioned in section 3.1, RAFT-Stereo is equal to an approximation of single Gaussian distribution with a fixed variance. We take RAFT-Stereo as the baseline method and implement the other two kinds of methods on it. We integrate thin volume [5, 20] with the baseline to realize the second kind of method. The third kind of method is realized by initializing the points uniformly within a pre-defined range and then sampling their nearby disparities to update themselves independently. The weighted average at the last iteration is regarded as the final disparity. For more details about the last two kinds of methods, please refer to our supplemental materials.

As shown in Table 1, our parameterized cost volume (PCV) is superior to the above three kinds of methods, which proves the effectiveness of the parameterized cost volume. It is worth noting that $SGAV$ is worse than $SGFV$. This might be caused by the limited view of a single Gaussian distribution, which tends to a rapid decrease in variance, further resulting in fast convergence to a local optimum trap. $MGFV$ shows a decrease in EPE but limited improvement in 1-px accuracy compared to $SGFV$. This might be because the multi-Gaussian distribution provides a global view but the fixed variance prevents fine-grained matching. Instead, our parameterized cost volume works in a coarse-to-fine manner. At the beginning iteration, it provides a global view by the uniformly initialized multi-Gaussian distribution with large variances for fast conver-

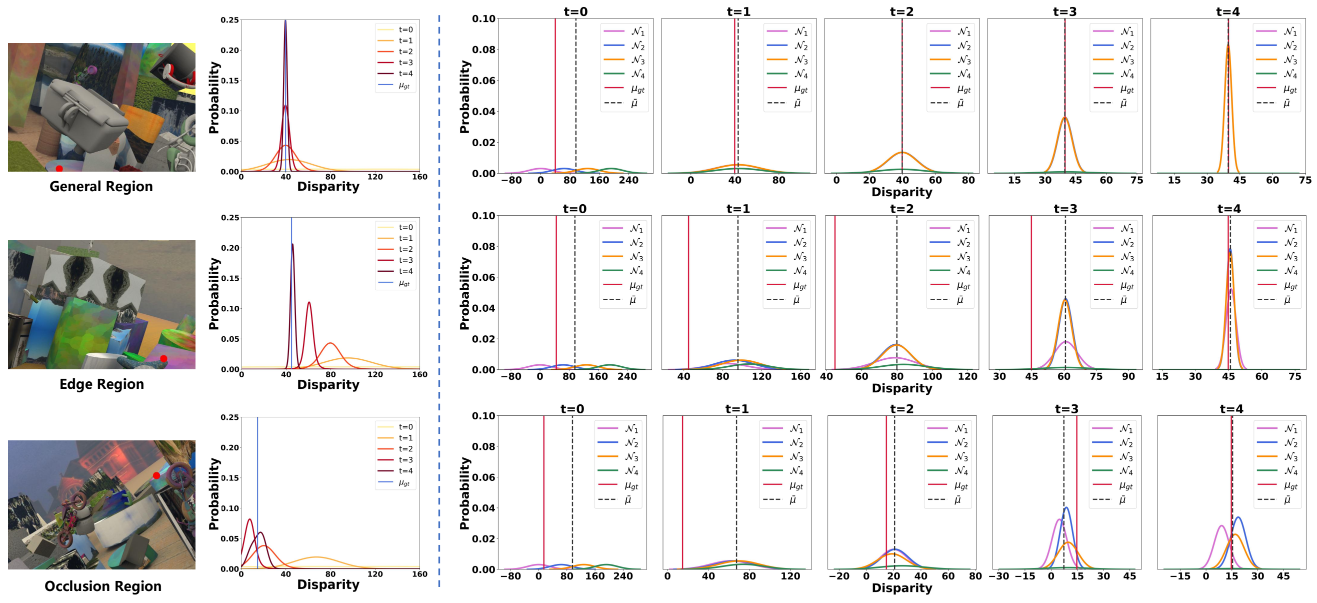


Figure 4: The visualization of the entire multi-Gaussian distribution (column two) and each weighted Gaussian distribution (columns three to seven) the at each iteration. The distributions correspond to the red points on the left images which are sampled from the general, edge, and occlusion regions, respectively.

Method	D1-bg (%)	D1-fg (%)	D1-all (%)	Time (ms)
PCW-Net [21]	1.37	3.16	1.67	440
DeepPruner(best) [3]	1.87	3.56	2.15	180
CREStereo [11]	1.45	2.86	1.69	410
RAFT-Stereo [13]	1.58	3.05	1.82	380
AANet+ [29]	1.65	3.96	2.03	60
DeepPruner(fast) [3]	2.32	3.91	2.59	60
HITNet [24]	1.74	3.2	1.98	20
Dec-Net [31]	2.07	3.87	2.37	50
ACVNet-fast [27]	-	-	2.34	48
PCVNet (ours)	1.68	3.19	1.93	56

Table 2: Benchmark results on all pixel areas of the KITTI 2015 dataset, categorized into two groups based on the runtime magnitude of the methods.

Method	Bad2.0 (%)	Bad4.0 (%)	avgerr (px)	rms (px)	A95 (px)	A99 (px)	time/MP (s/MP)	time/GD (s/Gdisp)
CFNet[20]	16.1	11.3	5.07	18.2	34.7	88.1	0.52	1.39
DeepPruner[3]	36.4	21.9	6.56	18.0	33.1	83.7	0.41	4.38
CREStereo[11]	8.13	5.05	2.10	10.5	5.48	49.7	0.77	2.22
RAFT-stereo[13]	9.37	6.42	2.71	12.6	8.89	64.4	2.19	5.76
HITNet[24]	12.8	8.66	3.29	14.5	11.4	77.7	0.11	0.29
HSM-Net[30]	16.5	9.68	3.44	13.4	17.6	63.8	0.10	0.22
PCVNet (ours)	13.6	7.77	2.71	11.9	8.83	58.9	0.14	0.37

Table 3: Benchmark results on all pixel areas of the Middlebury 2014 dataset, categorized into two groups based on the time cost of the methods.

gence to the ground truth. As the iteration goes on, the means of Gaussian distributions converge and their variances decrease, thus it becomes to focus on the local disparity space for fine-grained matching.

Uncertainty-aware Refinement We compute uncertainty from weights, variances, and means to guide the improvement of the disparity map at the last iteration. As presented in Table 1, our refinement slightly promotes the results in EPE error and achieves better improvement on 1px-error and 3px-error rates. These results show that our refinement works in cases when optimization oscillates locally around the ground truth. We also visualize uncertainty maps to illustrate the uncertainty-aware characteristics of the module in the supplementary materials.

Large Disparity Range We analyzed the effect of the disparity range growth on the time cost of various methods by stretching the image width. We maintain the structure of the models and increased the predefined disparity range for each method based on the image stretching ratio, then recorded the runtime. As shown in Figure 1a (right), our method exhibits the slowest time cost increase rate compared to other methods. Conversely, both RAFT-Stereo and CREStereo experience significant increases in time cost when dealing with the large disparity range. This result proves that our method relieves the rapid increase of time cost with the growing disparity range.

Convergence Speed We analyze the convergence speed of our method in the following two aspects. (1) We illustrate the average error changing with the increase of iteration in Figure 1a (left). Compared to RAFT-Stereo and

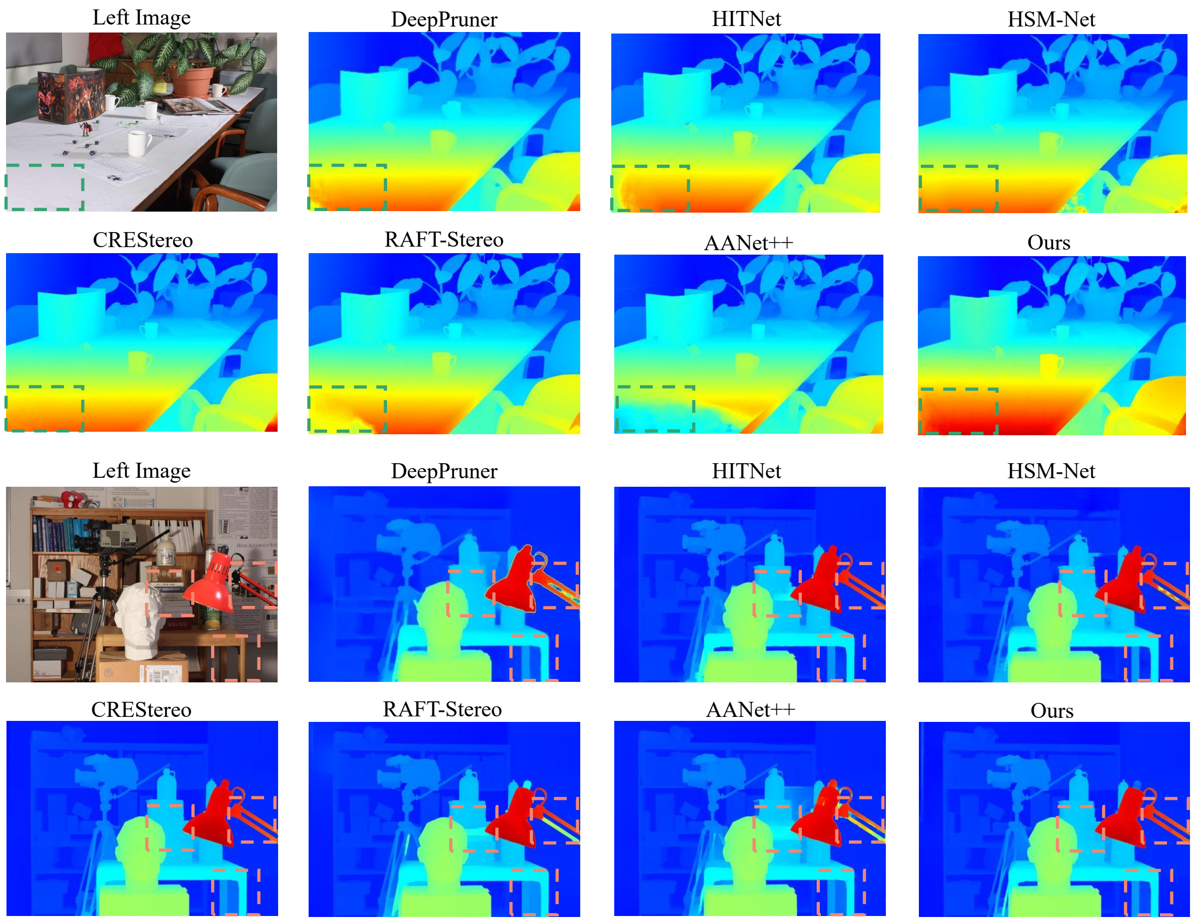


Figure 5: The visualization of results on the Middlebury 2014 dataset.

Setting	FLOPS (G)	Memory (MB)	time (ms)	EPE (px)
RAFT-Stereo ($T = 32$)	3176.51	2761	205	0.69
RAFT-Stereo*,† ($T = 4$)	678.45	2347	39	0.79
Ours*,† ($T = 4$)	798.23	2503	41	0.71

Table 4: The ablation results of computation cost on SceneFlow. * represents sharing the backbone for contextual and matching features. † denotes using the slow-fast GRU. The computation costs are measured on an NVIDIA A100 GPU.

CREStereo, our method achieves the smallest number of iterations with even better EPE. In practice, we only use 4 iterations for inference whereas RAFT-Stereo requires 32 iterations (and even up to 80 iterations for certain datasets), and CREStereo requires 20 iterations. Our method reduces the iteration of RAFT-Stereo by $8 \sim 20$ times and that of CREStereo by 5 times, which proves the fast convergence speed of our method. (2) We conduct a comprehensive analysis of the multi-Gaussian distribution visualization at each iteration. Figure 4 illustrates that, in general regions, the distribution rapidly converges to the ground truth in the initial iterations. The moving step and the variance of distribution progressively decrease as the iteration progresses, which enables fine-grained matching and indicates increasing confidence in the predictions. In the case of edge regions, the distribution gradually converges towards the ground truth, exemplifying the process of edge refinement. The adaptive variance of PCV accelerates the process of convergence, often requiring only a few iterations. For ill-posed regions like occlusion, the means of distributions diverge, and the variances exhibit relatively higher values, consequently yielding a large variance for the overall distribution. It reveals that the variance serves as a good indicator of uncertainty, which is beneficial to improve the disparity of ill-posed areas.

Efficiency Analysis Our method is compared with RAFT-Stereo [13] in terms of FLOPS, memory consumption, and speed, while considering the accuracy. The training strategy of the settings is the same as the ablation study. As presented in Table 4, with the same 4 iterations, PCV brings a slight cost increase (17.6% FLOPS, 6.6% memory, and 5.1% inference time) and a significant EPE improvement (10.1%) compared to the RAFT-Stereo. As for the RAFT-Stereo with 32 iterations, PCV greatly reduces the computation cost (74.8% FLOPS and 80.0% inference time) with comparable performance (-2% EPE). The results

Method	Deeppruner (best) [3]	ACVNet [27]	RAFT-Stereo [13]	IGEV-Stereo [28]	Deeppruner (fast) [3]	ACVNet (fast) [27]	PCVNet (Ours)
EPE (px)	0.86	0.48	0.65	0.47	0.97	0.77	0.62
Runtime (ms)	130	201	254	370	62	48	62

Table 5: The results of Sceneflow dataset. The methods are categorized into two groups based on the runtime. All the runtimes were measured on an NVIDIA 3090 GPU.

Method	Bad 2 (%)	Bad 4 (%)	Bad 6 (%)	Bad 8 (%)	MAE (px)	RMSE (px)	time/MP (s/MP)
CFNet [20]	66.85	46.02	35.48	29.74	19.65	43.01	0.30
RAFT-Stereo [13]	38.66	23.32	17.65	14.55	7.56	17.39	1.01
PCVNet (ours)	27.41	14.91	11.06	9.03	6.21	15.42	0.14

Table 6: Benchmark results of the Booster dataset. The metric time/MP is measured on an NVIDIA 3090 GPU.

reveal that our method can significantly speed up the convergence of dynamic cost volume at an affordable computation cost.

4.4. Benchmark Results

SceneFlow We compare our method with SOTA methods on the Sceneflow dataset [16]. Table 5 illustrates that our method gets better performance on the EPE metric while speeding up the RAFT-Stereo by 4 times. Overall, our method achieves the best trade-off between time and accuracy.

KITTI 2015 We present a comprehensive comparison of our method with SOTA approaches on the KITTI 2015 dataset [17], showcasing the effectiveness of our approach in real-world outdoor scenes. The results on all pixel areas are depicted in Figure 2, while supplementary materials provide detailed results on non-occluded pixel areas. As observed in Table 2, our method achieves nearly the highest accuracy among methods with runtimes smaller than 100 ms. Compared to other approaches, our method demonstrates comparable performance with significantly improved computational speed. In particular, our method outperforms DeepPruner (best) in both accuracy and runtime metrics and achieves a 7 times acceleration of RAFT-Stereo. As for CREStereo, we use no additional data but only the SceneFlow dataset for pre-training and the KITTI dataset for fine-tuning, attaining a 0.24% accuracy decrease on D1-all while achieving a remarkable $7 \sim 8$ times speedup.

Middlebury We also evaluate our method on the Middlebury dataset [19] to verify the effectiveness in real-world indoor scenes. The results on all pixel areas are presented in Figure 3. For the results on non-occluded pixel areas and other evaluation metrics, please refer to our supplementary materials. As shown in Table 3, the accuracy of our method is comparable to the results of CREStereo, while our running speed is $5 \sim 7$ times faster than CREStereo's. As for RAFT-Stereo, we achieve comparable results on Bad4.0 and avgerr and better results on rms, A95, and A99. Furthermore, our speed is almost 15 times faster than RAFT-Stereo's. Among the other approaches, our method is almost the best one. Besides the quantitative analysis, we also provide the visualization of the disparity map in Figure 5. The results of our method are comparable to CREStereo's disparity maps and are more accurate and smoother than the others.

Booster We evaluate our method on the Booster dataset [18]. The results in Table 6 demonstrate the superiority of our approach, exhibiting significantly better performance across all evaluation metrics compared to other methods on the benchmark. Moreover, the time cost of our method is merely 1/2 that of CFNet and 1/7 that of RAFT-Stereo. Additional visualizations of this dataset can be found in the supplementary materials.

4.5. Limitations and Discussion

As demonstrated in the aforementioned experiments, our method achieves significant runtime acceleration without compromising accuracy. Nevertheless, it is crucial to acknowledge certain limitations that persist, such as the use of naive gradient descent updates for JS-divergence-based optimization. This naive update scheme lacks robustness and may lead to model collapse under certain circumstances.

5. Conclusion

In this paper, we have proposed a parameterized cost volume that efficiently encodes the entire disparity space utilizing multi-Gaussian distribution. We formulated the parameter computation of multiple Gaussian as a JS-divergence-based optimization problem and solved it through a feed-forward differential module. Our parameterized cost volume with the feed-forward differential module can enable a global view for fast convergence at the beginning iteration and provide a local view for fine-grained matching as the iterations progress. The experimental results demonstrate that our method can speed up the runtime by $4 \sim 15$ times without sacrificing accuracy.

Acknowledgement This work was supported by the Natural Science Foundation of China (NSFC) under Grants No.62176021.

References

[1] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5410–5418, 2018.

[2] Shuo Cheng, Zexiang Xu, Shilin Zhu, Zhuwen Li, Li Erran Li, Ravi Ramamoorthi, and Hao Su. Deep stereo using adaptive thin volume representation with uncertainty awareness. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2524–2534, 2020.

[3] Shivam Duggal, Shenlong Wang, Wei-Chiu Ma, Rui Hu, and Raquel Urtasun. Deeppruner: Learning efficient stereo matching via differentiable patchmatch. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4384–4393, 2019.

[4] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient large-scale stereo matching. In *Asian conference on computer vision*, pages 25–38. Springer, 2010.

[5] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2495–2504, 2020.

[6] Xiaoyang Guo, Kai Yang, Wukui Yang, Xiaogang Wang, and Hongsheng Li. Group-wise correlation stereo network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3273–3282, 2019.

[7] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(2):328–341, 2008.

[8] Takeo Kanade and Masatoshi Okutomi. A stereo matching algorithm with an adaptive window: Theory and experiment. *IEEE transactions on pattern analysis and machine intelligence*, 16(9):920–932, 1994.

[9] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 66–75, 2017.

[10] Sameh Khamis, Sean Fanello, Christoph Rhemann, Adarsh Kowdle, Julien Valentin, and Shahram Izadi. Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 573–590, 2018.

[11] Jiankun Li, Peisen Wang, Pengfei Xiong, Tao Cai, Ziwei Yan, Lei Yang, Jiangyu Liu, Haoqiang Fan, and Shuaicheng Liu. Practical stereo matching via cascaded recurrent network with adaptive correlation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16263–16272, 2022.

[12] Zhengfa Liang, Yiliu Feng, Yulan Guo, Hengzhu Liu, Linbo Qiao, Wei Chen, Li Zhou, and Jianfeng Zhang. Learning deep correspondence through prior and posterior feature constancy. *arXiv preprint arXiv:1712.01039*, 2017.

[13] Lahav Lipson, Zachary Teed, and Jia Deng. Raft-stereo: Multilevel recurrent field transforms for stereo matching. In *2021 International Conference on 3D Vision (3DV)*, pages 218–227. IEEE, 2021.

[14] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5695–5703, 2016.

[15] Yamin Mao, Zhihua Liu, Weiming Li, Yuchao Dai, Qiang Wang, Yun-Tae Kim, and Hong-Seok Lee. Uasnet: Uncertainty adaptive sampling network for deep stereo matching. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6311–6319, 2021.

[16] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016.

[17] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070, 2015.

[18] Pierluigi Zama Ramirez, Fabio Tosi, Matteo Poggi, Samuele Salti, Stefano Mattoccia, and Luigi Di Stefano. Open challenges in deep stereo: the booster dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21168–21178, 2022.

[19] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition (GCPR))*, pages 31–42, 2014.

[20] Zhelun Shen, Yuchao Dai, and Zhibo Rao. Cfnet: Cascade and fused cost volume for robust stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13906–13915, 2021.

[21] Zhelun Shen, Yuchao Dai, Xibin Song, Zhibo Rao, Dingfu Zhou, and Liangjun Zhang. Pcw-net: Pyramid combination and warping cost volume for stereo matching. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII*, pages 280–297. Springer, 2022.

[22] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. *arXiv: Learning*, 2017.

[23] Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 25(7):787–800, 2003.

[24] Vladimir Tankovich, Christian Hane, Yinda Zhang, Adarsh Kowdle, Sean Fanello, and Sofien Bouaziz. Hitnet: Hierarchical iterative tile refinement network for real-time stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14362–14372, 2021.

[25] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–*

28, 2020, Proceedings, Part II 16, pages 402–419. Springer, 2020.

[26] Alessio Tonioni, Fabio Tosi, Matteo Poggi, Stefano Mattoccia, and Luigi Di Stefano. Real-time self-adaptive deep stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 195–204, 2019.

[27] Gangwei Xu, Junda Cheng, Peng Guo, and Xin Yang. Acvnet: Attention concatenation volume for accurate and efficient stereo matching. *arXiv preprint arXiv:2203.02146*, 2022.

[28] Gangwei Xu, Xianqi Wang, Xiaohuan Ding, and Xin Yang. Iterative geometry encoding volume for stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21919–21928, 2023.

[29] Haofei Xu and Juyong Zhang. Aanet: Adaptive aggregation network for efficient stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1959–1968, 2020.

[30] Gengshan Yang, Joshua Manela, Michael Happold, and Deva Ramanan. Hierarchical deep stereo matching on high-resolution images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5515–5524, 2019.

[31] Chengtang Yao, Yunde Jia, Huijun Di, Pengxiang Li, and Yuwei Wu. A decomposition model for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6091–6100, 2021.

[32] Jure Zbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1592–1599, 2015.