Multi-Level Context Ultra-Aggregation for Stereo Matching

Guang-Yu Nie[1] Ming-Ming Cheng[2] Yun Liu[2] Zhengfa Liang[3]
Deng-Ping Fan[2] Yue Liu[1,4 *] Yongtian Wang[1,4]
[1] Beijing Institute of Technology [2] TKLNDST, CS, Nankai University
[3] National Key Laboratory of Science and Technology on Blind Signal Processing
[4] AICFVE, Beijing Film Academy
http://mmcheng.net/mcua/

Abstract

Exploiting multi-level context information to cost volume can improve the performance of learning-based stereo matching methods. In recent years, 3-D Convolution Neural Networks (3-D CNNs) show the advantages in regularizing cost volume but are limited by unary features learning in matching cost computation. However, existing methods only use features from plain convolution layers or a simple aggregation of multi-level features to calculate cost volume, which is insufficient because stereo matching requires discriminative features to identify corresponding pixels in rectified stereo image pairs. In this paper, we propose a unary features descriptor using multi-level context ultra-aggregation (MCUA), which encapsulates all convolutional features into a more discriminative representation by intra- and inter-level features combination. Specifically, a child module that takes low-resolution images as input captures larger context information; the larger context information from each layer is densely connected to the main branch of the network. MCUA makes good usage of multi-level features with richer context and performs the image-to-image prediction holistically. We introduce our MCUA scheme for cost volume calculation and test it on PSM-Net. We also evaluate our method on Scene Flow and KITTI 2012/2015 stereo datasets. Experimental results show that our method outperforms state-of-the-art methods by a notable margin and effectively improves the accuracy of stereo matching.

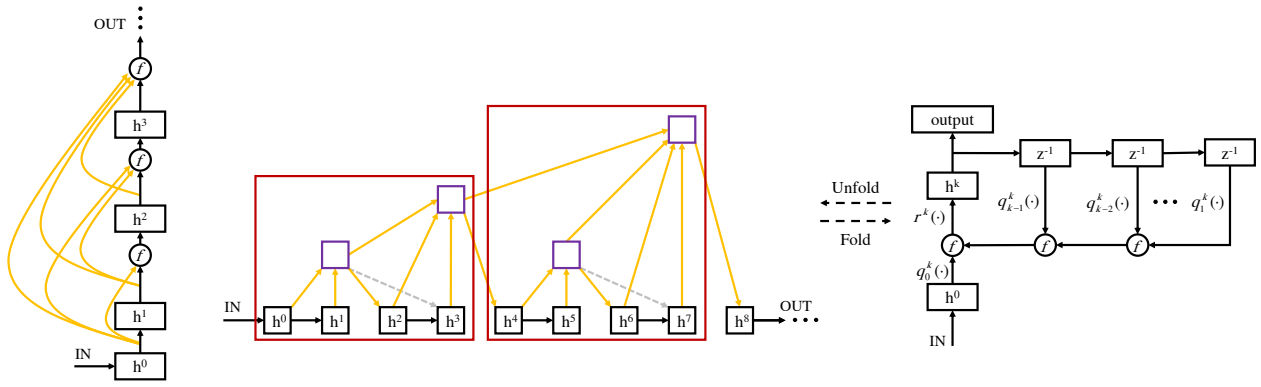
1. Introduction

Stereo matching, also known as disparity estimation, aims to find corresponding points in a pair of rectified stereo images. It serves as an essential subclass of computer vision [26, 28]. Cost volume plays a vital role for Convolution Neural Networks (CNNs) based stereo matching methods, which has been validated by [28]. Traditional 1-D correlation along the disparity line enables to generate a 3-D stereo cost volume [14, 15], but it loses lots of information due to its multiplicative approximation to the volume. As an improvement, a simple concatenation, instead of 1-D correlation, is implemented to combine the unary features from left and right inputs across each disparity level to generate a 4-D cost volume, and then 3-D CNNs are incorporated in the context to regularize this 4-D cost volume [9]. 4-D cost volume based methods [9, 2] usually outperform 3-D cost volume [14, 11] based methods, because 4-D cost volume can preserve the feature dimensions.

Skip connection [7, 17] in CNNs encourages the integration of hierarchical representations, and may also contribute to stereo matching for the improvement of the cost volume [29, 4]. Stereo matching is a regression problem which aims to achieve pixel-wise dense prediction, but it usually generates discontinuity in the occluded areas, and it suffers from aperture problem in texture-less regions such as sky or other flat areas [9], so it is more concerned with the merge of multi-level context information. In DenseNets [8] and DLA [25], large receptive fields are achieved at deep stages of a network, but they only refer to intra-level combination of features and enable not to obtain large receptive field at shallow stages. Therefore, it lacks enough global information for more context information when using dense connection or DLA scheme on the matching cost calculation in the stereo matching task. This problem makes these two architectures be limited when learning context information.

To solve this problem, we improve the discriminative ability of unary features for matching cost calculation by introducing ***M**ulti-level **C**ontext **U**ltra-**A**ggregation* (**MCUA**) scheme which combines the features at the shallowest, smallest scale and deeper, larger scales using just "shallow" skip connections. Except for intra-level combination inspired by DenseNets [8] and DLA [25], MCUA contains an *independent child module* which introduces the inter-level

*Yue Liu (liuyue@bit.edu.cn) is the corresponding author.



(a) DenseNets with shared weights (b) Deep Layer Aggregation with shared weights (c) Networks in Higher Order RNNs form

Figure 1. DenseNets and DLA belong to the family of Higher Order RNNs. (a) Dense connection scheme; (b) DLA scheme between neighboring groups (red box), consisting of HDA (combining stages in groups) and IDA (combining groups); (c) Higher Order RNNs framework. The orange solid lines indicate the skip connections between each two stages.

combination scheme. The main contributions of this paper include i) we propose MCUA for both intra- and inter-level features aggregation and formulate it as a Higher Order RNN; ii) the experimental results show that MCUA improves matching cost calculation significantly.

2. Related Work

Stereo matching can be implemented using multistage techniques [1] which typically include four main steps, i.e., matching cost computation, cost aggregation, disparity computation and optimization, and disparity refinement [20]. Early learning-based methods adopted neural networks to replace one or more stages in the traditional stereo pipeline [27, 26, 19, 14, 21]. Some approaches achieve better performance by integrating all steps into a whole network for joint optimization. Mayer et al. [15] introduced a 1-D correlation layer to integrate the unary features along the disparity line, which can provide a 3-D cost volume for end-to-end training. Pand et al. [18] proposed a cascaded CNN architecture by first obtaining an initial disparity map, and then employing residual learning for refinement. Liang et al. [11] presented feature constancy to measure the correspondence between two input images, which is then used to refine the disparity. EdgeStereo, developed by song et al. [23], introduces a multi-task architecture to generate the final disparity map by integrating a one-stage stereo network and a proposed edge detection network. SegStereo, proposed in [24], introduces two incorporation strategies of semantic cues, including semantic information embedding and semantic loss regularization added to softmax loss.

Since 1-D correlation is a multiplicative approximation to the stereo cost volume, it will lose some useful information and is thus harmful to context learning. GC-Net [9] introduces the 4-D cost volume to incorporate context in cost volume regularization. This method does not collapse the feature dimension when generating stereo cost volume. Recently, PSM-Net [2] exploit the context information for stereo matching by applying an SPP module [6] on cost volume calculation and utilizing three stacked 3-D hourglass networks to regularize this 4-D cost volume. StereoNet [10] is a real-time end-to-end network for stereo matching, in which a cost volume with meager resolution but encoding all information is first used to obtain an initial disparity map, and then a learned upsampling function is used for refinement. In our work, we apply a novel aggregation pattern, MCUA, to generate the unary features with better context support. The experimental results demonstrate the effectiveness of MCUA in stereo matching.

3. Reviewing Feature Aggregation Schemes

In this section, we first review DenseNets [8] and DLA [25], and formulate these two aggregation schemes with Higher Order RNNs [22, 12, 3]. Then, we discuss the limitations of features aggregation when applying these schemes into stereo matching.

3.1. DenseNets

DenseNets [8] apply a dense connection scheme on the group in which feature maps generated by all stages have the same resolution and scale. As shown in Fig. 1(a), the signal "h^k" indicates k-th stage of this block, it receives the feature maps from all preceding stages, $h^0, ..., h^{k-1}$, and shares its feature maps with all its subsequent stages. It can be formulated as follows:

$$h^k = r^k[f_{t=0}^{k-1} q_t^k(h^t)] \tag{1}$$

where $q_t^k(h^t)$ is the feature extraction function, $r^k(\cdot)$ is the transmit function to transform the gathered information before this information flowing into the k-th stage, and f denotes the concatenation operation for data fusion.

Fig. 1(c) shows the framework of Higher Order RNNs, where the signal "h^k" indicates the hidden state of the

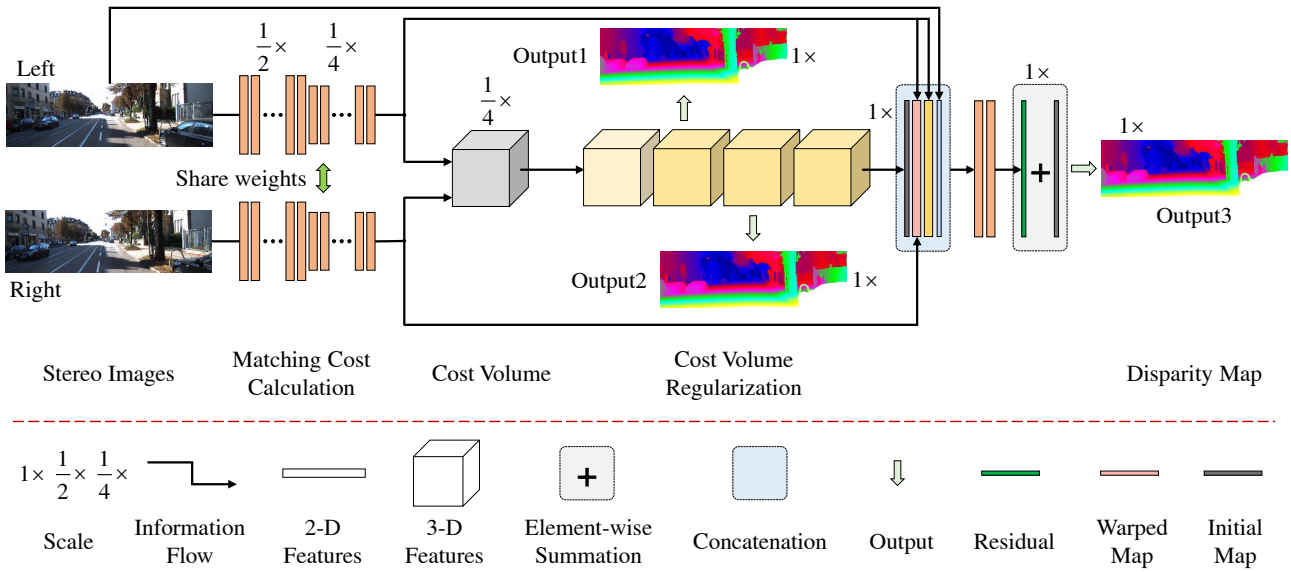


Figure 2. The diagrammatic sketch of our proposed network (EMCUA). It is constructed based on PSM-Net [2] by applying MCUA on the architecture of matching cost calculation and adding a residual module at the end. A pair of stereo images (i.e., Left, Right) pass through the network for disparity prediction (i.e., Output3). Fig. 3 shows the detail of the updated architecture of matching cost calculation.

RNNs at k-th step, $r^k(\cdot)$ indicates a transform function, the symbol "z^{-1}" indicates a time-delay unit, and "f" denotes the operation for aggregation (e.g., summation, concatenation, etc.). In the Higher Order RNNs, all functions share the same weights, i.e., $\forall t, k,\ q_t^k(\cdot) \equiv q_t(\cdot)$ and $\forall k,\ r^k(\cdot) \equiv r(\cdot)$. When the signals share parameters [3], DenseNets can be represented as Higher Order RNNs, which shows that DenseNets belong to the family of Higher Order RNNs. DenseNets cannot merge features across scales and resolutions, which loses lots of low-level information. In this paper, we develop a general feature aggregation scheme to solve this problem.

3.2. DLA

As shown in Fig. 1(b), a network with nine stages is designed as the backbone, on which we apply DLA scheme. Due to different scales of output features, stages of this backbone can be divided into three groups (represented by red boxes): $h^0, \ldots, h^3$ for the first group, $h^4, \ldots, h^7$ for the second group, and h^8 for the third group. DLA consists of two aggregation schemes [25]: (i) the Iterative Deep Aggregation (IDA) merges features across scales and resolutions, in which the outputs of aggregation nodes are downsampled before merging with other features. (ii) the Hierarchical Deep Aggregation (HDA) merges the outputs of the aggregation nodes into the backbone serving as the inputs to the next sub-tree. This makes each stage only selectively use a subset of outputs from all previous stages, as illustrated in Fig. 1(b), deleting the short connections with gray dashed lines by taking $q_t^k(\cdot) = 0$. We follow DenseNets shown in Eq. (1) to describe DLA as follows:

$$h^k = \begin{cases} r^k[\sum_{t=0}^{k-1} q_t^k(h^t)], k = 4n \\ r^k[q_{k-1}^k(h^{k-1})], k = 4n+1 \\ r^k[q_{k-2}^k(h^{k-2}) + q_{k-1}^k(h^{k-1})], k = 4n+2 \\ r^k[q_{k-1}^k(h^{k-1})], k = 4n+3 \end{cases} \tag{2}$$

where $n = 0, 1, 2, \ldots$ indicates the index of the group. Similarly, the DLA scheme can also be represented as the form of Higher Order RNNs. However, the fusion in DLA only refers to the intra-level combination. To overcome this disadvantage, we introduce an independent child module to fuse features with the inter-level combination, where large receptive fields can be obtained at shallow stages.

4. Network Architecture

In this section, we introduce each part of the proposed network which is developed from PSM-Net [2]. An overall illustration is shown in Fig. 2.

4.1. MCUA Scheme

We apply the proposed **MCUA** scheme (in Fig. 3) to PSM-Net [2] for matching cost computation. The branch (a) of MCUA can be regarded as the backbone. It is a 2D-CNN which is the same as the matching cost computation network in PSM-Net. We divide the backbone into nine stages based on the layer definitions in [2]: The first seven stages, $F_0, \ldots, F_6$, correspond to conv0_1, conv0_2, conv0_3, conv1_x, conv2_x, conv3_x, and conv4_x, respectively; The eighth stage, F_7, contains the SPP module fol-

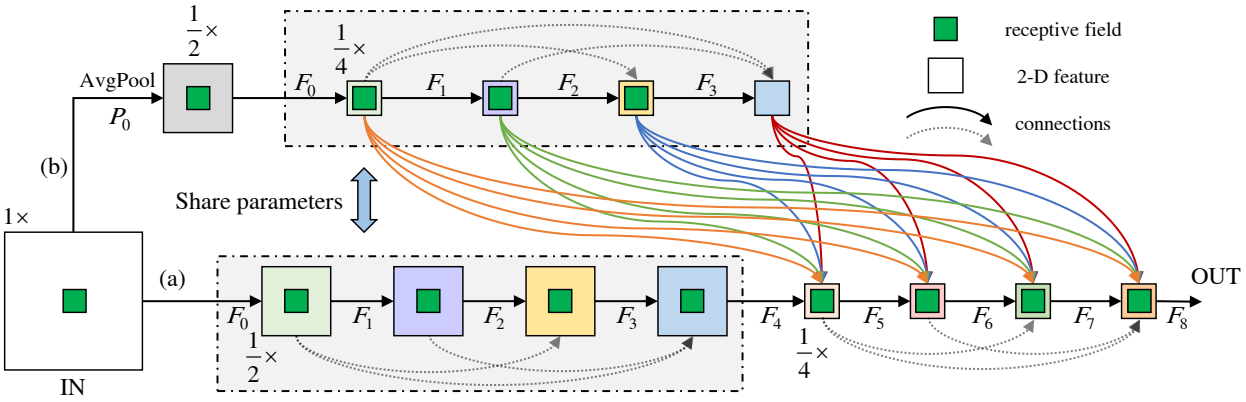


Figure 3. Illustration of MCUA scheme. Branch (a) is the backbone, while branch (b) is the independent child module. Each colored block represents the feature map generated by one stage, while each green block denotes the receptive field that the next stage has. The intra-level combination is described by dashed gray lines, while the inter-level combination is depicted by solid color lines. The unary features generated by F_8 is the final output of this architecture. Tab. 1 shows the layer-wise definition of MCUA.

lowed by a 3×3 convolution operation; The ninth stage, F_8, is a 1×1 convolution operation which aims to fuse the combined features. We use the output of the last layer of each stage as the feature information for other operations. This design is natural since the deepest layer of each stage should have the most reliable features. According to the sizes of feature maps, the backbone can be divided into two groups: Stages $F_0, \ldots, F_3$ belong to the first group, whose output feature maps have a size of $\frac{1}{2}\times$ scale, and stages $F_4, \ldots, F_8$ belong to the second group whose output feature maps have a size of $\frac{1}{4}\times$ scale.

Fig. 3 and Tab. 1 illustrates the details of MCUA. MCUA allows each stage to receive the features from all previous stages and enables its outputs to pass through all subsequent stages. In details, features (i.e., $h^1, h^2, \ldots$) from the previous stages are first aggregated by element-wise summation, and then pre-activated before passing through the next stage. We formulate MCUA as follows:

$$h_1^k = r^k[\sum_{t=0}^{k-1} q_t^k(h_1^t)](0 \le k \le m), \quad (3)$$

$$h_2^k = r^k[\sum_{t=0}^{m-1} q_t^k(\alpha h_1^t) + q_m^{m+1}(h_1^m)](k = m+1), \quad (4)$$

$$h_2^k = r^k[\sum_{t=0}^{m-1} q_t^k(\alpha h_1^t) + q_m^{m+1}(h_1^m) + \sum_{t=m+1}^{k-1} q_t^k(h_2^t)] \quad (m+2 \le k \le n), \quad (5)$$

where $m = 4$, $n = 8$, "h_1^k" denotes the output of stage F_k with the feature maps scale of $\frac{1}{2}\times$ *input size*, and "h_2^k" denotes the output of stage F_k with the scale of $\frac{1}{4}\times$ *input size*. Among all $n+1$ stages, F_m is a special stage which receives the feature maps with $\frac{1}{2}\times$ *input size* and outputs the feature maps with $\frac{1}{4}\times$ *input size*. α ($\alpha > 1$) is the expanding factor to control the ratio of the increased area, so that one bigger receptive field captures more information than a smaller one.

4.1.1 Intra-level Combination

The intra-level combination fuses feature maps in each group, in which dense connection, described by dashed lines in Fig. 3, are applied between each of the two stages. In details, features are transformed by a linear function, $q_t^k(x) = \beta x$ where β is defined as a linear coefficient. This transformation is achieved by a 1×1 convolution operation [13] to make the feature maps match with each other in dimensions. The transformed features from previous stages are integrated by element-wise summation and pre-activated, and then, flow to the next stage. For instance, the number of channels of the feature map generated by stage F_4 is 64, while that generated by stage $F_{5,6,7}$ is 128. Before merging and flowing to stage F_8, the feature map of stage F_4 needs to be linearly transformed into an immediate map with 128 channels.

4.1.2 Inter-level Combination

As shown in Fig. 3, we use an independent child module to introduce inter-level aggregation which is represented by the solid color lines. The independent child module first adopts an average pooling operation, P_0, to reduce the size of input by half, and then uses four stages (i.e., $F_0, \ldots, F_3$) to learn unary features. Each of these four stages shares the same internal architecture with the first group of backbone, and parameters of corresponding layers are tied. Generally, large receptive fields are usually achieved at deep stages of a network. By using the independent child module, it can obtain large receptive fields at shallow stages, which can

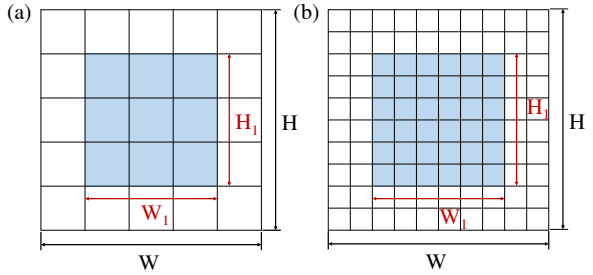


Figure 4. A fix-sized receptive field (blue block) in CNNs enables to filter larger region when decreasing the scales of input (scale of b is half of the scale of a). H and W denote the height and width of the area in feature maps, and H_1 and W_1 denote the height and width of the receptive field, respectively.

be explained by Fig. 4: the receptive field with a size of $H_1 \times W_1$ enables to capture more visual information from the downsampled inputs (i.e., Fig. 4 (b)) than that on raw inputs (i.e., Fig. 4 (a)). Since the child module shares parameters with backbone, we have $h_2^t = \alpha h_1^t$ in Eq. (5), in which α ($\alpha > 1$) indicates the spatial information increased by applying a fixed-size receptive field on a different area of feature maps. Besides, linear transformations are also applied in dense path. We set the parameter β, which adopts the same strategy as features intra-level combination, to make features adapt to the dimensions of subsequent stages. For stereo matching, the independent child module can provide more context information for the features to calculate cost volume, which usually occurs at shallow stages. In Sec. 6.2, we will show the importance of the independent child module for learning contextual information and improving the performance of stereo matching.

4.2. Disparity Regression

The *soft argmin* is a valid operation to regress values over probability volumes regularized by 3-D CNNs [9], because it is fully differentiable and enables back-propagation training. The regressed value for each pixel is calculated by a weighted average of all modes, which can be shown as

$$D_{h,w} = \sum_{d=0}^{D_{max}} d \times \sigma(-c_{d,h,w}) \tag{6}$$

where $c_{d,h,w}$, $\sigma_{d,h,w}$ and d correspond to the cost value, softmax operation for each pixel, and the disparity value, respectively.

4.3. Outputs

As shown in Fig. 2, MCUA contains three hourglass networks, each of which generates a disparity map. These three outputs are used to calculate loss when training the network, and the last output is used for testing. The output of the third hourglass network is considered as an initial disparity map. To refine the foreground of initial prediction, a residual module is added at the end of the network. It first generates a residual map and then combine with the initial disparity map using element-wise summation to obtain the final output, i.e., Output3. As shown in Fig. 2, the residual module contains three convolution layers with a kernel size of 5 and stride of 2. The layer definitions of the residual module is shown in supplementary materials. The whole network is named EMCUA, which is slightly different from MCUA in the last output.

Table 1. Architecture of MCUA

Stage	Type	K	S	P	D	N	R	Output	Dim.	I/O	Input
IN								input	3	–/1	IN
Backbone											
F_0	Conv.	3	1	1	1	1	3	C01	3/32	1/2	input
F_1	Conv.	3	1	1	1	1	5	C02	32/32	2/2	C01
F_2	Conv.	3	1	1	1	1	7	C03	32/32	2/2	C01 + C02
F_3	Conv.	3	1	1	1	3	13	C1x	32/32	2/2	C01 + C02 + C03
Independent Child Module (i.e., Branch(b))											
P_0	AvgP	2	2	0	0	1	2	P20	3/3	1/2	IN
F_0	Conv.	3	1	1	1	1	6	C201	3/32	2/4	P20
F_1	Conv.	3	1	1	1	1	11	C202	32/32	4/4	C201
F_2	Conv.	3	1	1	1	1	16	C203	32/32	4/4	C201 + C202
F_3	Conv.	3	1	1	1	3	31	C21x	32/32	4/4	C201 + C202 + C203
Backbone											
F_4	Conv.	3	1	1	1	16	45	C2x	32/64	2/4	C01 + C02 + C03 + C1x
F_5	Conv.	3	1	1	1	3	51	C3x	64/128	4/4	C201 + C202 + C203 + C21x + C2x
F_6	Conv.	3	1	1	1	3	57	C4x	128/128	4/4	C201 + C202 + C203 + C21x + C2x + C3x
–	AvgP	[64 32 16 8]	[64 32 16 8]	0	1	1	–	B1 B2 B3 B4	128/32	4/4	C201 + C202 + C203 + C21x + C2x + C3x + C4x
	Conv.	1	1	1	1	1	–				
	Ups.	–	–	–	–	–	–				
–	ConC	–	–	–	–	–	–	M1	128/128	4/4	B1, B2, B3, B4, C2x, C4x
F_7	Conv.	3	1	1	1	1	59	FSPP	320/128	4/4	M1
F_8	Conv.	1	1	0	1	1	59	fusion	128/32	4/4	C201 + C202 + C203 + C21x + C2x + C3x + C4x + FSPP

K, S, P, D, N, R: kernel size, stride, padding, dilation, number, and receptive field of convolutional layer; **Dim.**: dimension of input/output feature maps; **I/O**: scale of input/output feature maps; Symbol "**+/-**": element-wise summation/subtraction operation; **ConC**: concatenation operation.

4.4. Loss Function

We train the whole network end-to-end with supervised learning by adopting *Smooth L1 Loss* which creates a criterion that uses a squared term if the absolute element-wise error falls below 1 and an **L1** term otherwise. This loss is

Table 2. KITTI2015 Results

Mod.	All (%)			Noc (%)		
	D1-bg	D1-fg	D1-all	D1-bg	D1-fg	D1-all
SegStereo	1.88	4.07	2.25	1.76	3.70	2.08
iResNet	2.25	**3.40**	2.44	2.07	**2.76**	2.19
CRL	2.48	3.59	2.67	2.32	3.12	2.45
GC-Net [9]	2.21	6.16	2.87	2.02	5.58	2.61
PSM-Net	1.86	4.62	2.32	1.71	4.31	2.14
MCUA	1.69	4.38	2.14	1.55	3.90	1.93
EMCUA	**1.66**	4.27	**2.09**	**1.50**	3.88	**1.90**

"**All**" and "**Noc**" : percentage of outliers averaged over ground truth pixels of all/non-occluded regions. "**D1-bg**", "**D1-fg**", and "**D1-all**": percentage of outliers averaged only over background regions, foreground regions, and all ground truth pixels.

less sensitive to outliers than **L1 Loss** and in some cases prevents exploding gradients. The loss is defined as:

$$Loss(x, y) = \frac{1}{n} \sum_{i} z_i \tag{7}$$

$$z_i = \begin{cases} 0.5(x_i - y_i)^2, if\ |x_i - y_i| < 1 \\ |x_i - y_i| - 0.5, otherwise \end{cases} \tag{8}$$

where x_i and y_i denote the ground truth and predicted disparities for each pixel *i*, respectively. The loss weights for the three intermediate supervision are 0.5, 0.7 and 1.0, respectively, which are the same with PSM-Net [2].

5. Experiments

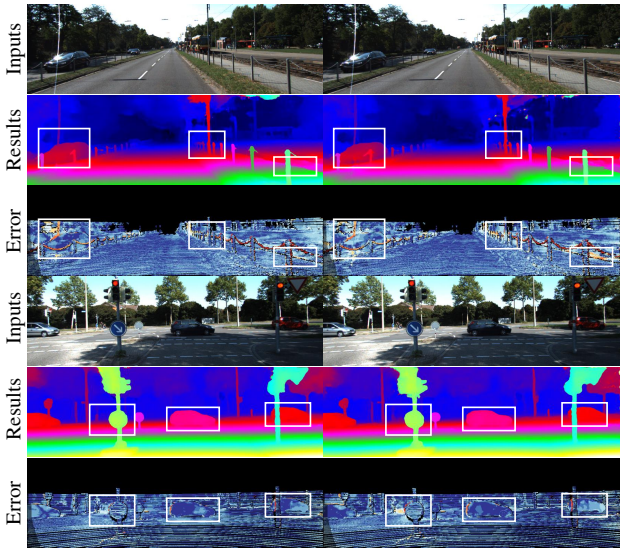
We test our proposed model on three datasets and compare it with the state-of-the-art architectures.

5.1. Implementation Details

We implement our proposed model using PyTorch and conduct experiments on four NVIDIA TITAN Xp GPUs.

Datasets We adopted three publicly available datasets for training and testing: **The Scene Flow datasets** [15] contain stereo images in 960×540 pixel resolution with 35454 for training and 4370 for testing, and all image pairs are rendered from various synthetic sequences, i.e., FlyingThings3D, Driving, and Monkaa. **KITTI2015/2012 datasets** consist of KITTI2015 dataset [16] (200 training and 200 test scenes in 1242×375 pixel resolution) and KITTI2012 dataset [5] (194 training and 195 test scenes in 1242×375 pixel resolution). These images were captured by driving in rural areas and on highways. For both KITTI training sets, we use 160 image pairs for training and the remains for validation.

Training The training process of EMCUA contains two steps. The first step is to train the updated model that



(a) Ours (b) PSM-Net

Figure 5. Results of our model and PSM-Net in KITTI2015 dataset

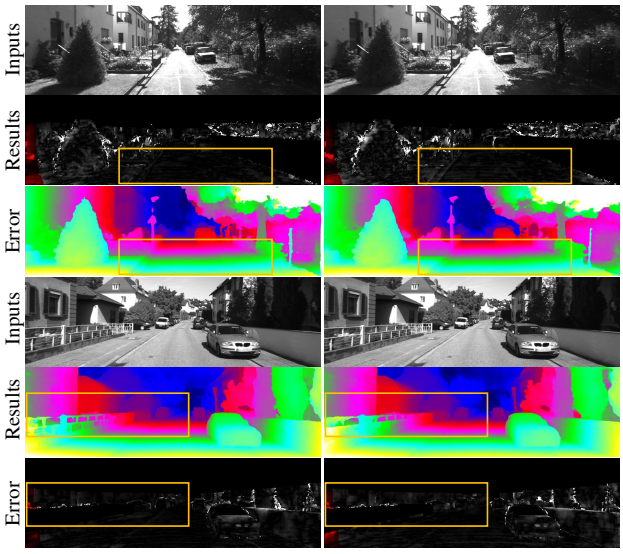
MCUA scheme is applied on the architecture of matching cost computation in PSM-Net. Before inputting to the network, each raw image is first processed by color normalization and then randomly cropped into patches with 256×512 resolution. The network is optimized end-to-end using Adam (Adaptive Moment Estimation) with β_1 of 0.9 and β_2 of 0.999. The batch size and maximum disparity (D) set to 8 and 192 *pixels*, respectively. We first train MCUA on Scene Flow datasets with a fixed learning rate of 0.001 for 20 epochs, then we fine-tune the network on KITTI2015/2012 dataset with stepped learning rates of 0.001 for 600 epochs and 0.0001 for another 400 epochs. Furthermore, for Scene Flow dataset, we extend the training to 70 epochs to get the final results. The second step refers to training the EMCUA in which a residual module is added at the end of MCUA. We first train EMCUA on Scene Flow datasets by 1 epoch using the trained parameters from MCUA on KITTI2015/2012 datasets, then continue to fine-tune EMCUA on KITTI2015/2012 datasets, respectively. The parameter settings in EMCUA training are as same as that in MCUA training.

Validating/Testing As shown in Fig. 2, **Output3**, the last of three outputs, is selected as the final result of the whole network, and we estimate the performance of both MCUA and EMCUA on both Scene Flow test and KITTI2015/2012 validating sets. To implement estimation, based on groundtruth we calculate the **end-point-error** of the results of each epoch for Scene Flow test set, while **three-pixel-error** of that for KITTI2015/2012 validating sets, respectively. After finishing the estimation, we use the trained parameters with the lowest error to predict the

Table 3. KITTI2012 Results

Mod	$> 2px$		$> 3px$		$> 4px$		$> 5px$		ME(px)	
	Noc	All	Noc	All	Noc	All	Noc	All	AN	AA
SegStereo	2.66	3.19	1.68	2.03	1.25	1.52	1.00	1.21	0.5	0.6
iResNet	2.69	3.34	1.71	2.16	1.30	1.63	1.06	1.32	0.5	0.6
GC-Net	2.71	3.46	1.77	2.30	1.36	1.77	1.12	1.46	0.6	0.7
PSM-net	2.44	3.01	1.49	1.89	1.12	1.42	0.90	1.15	0.5	0.6
MCUA	2.07	2.64	1.30	1.70	0.98	1.29	0.80	1.04	0.5	0.5
EMCUA	**2.02**	**2.56**	**1.26**	**1.64**	**0.95**	**1.24**	**0.76**	**0.99**	**0.4**	**0.5**

"**Noc**" and "**All**": percentage of erroneous pixels in non-occluded areas, and in total. "**AN**" and "**AA**": average disparity/end-point error in non-occluded areas, and in total. "**ME**": mean error.



(a) Ours (b) PSM-Net

Figure 6. Results of our model and PSM-Net in KITTI2012 dataset

Table 4. Performance comparison on Scene Flow test set

Mod.	EPE	Mod.	EPE	Mod.	EPE
MCUA	**0.56**	PSM-Net [2]	1.09	StereoNet [10]	1.10
CRL. [18]	1.32	iResNet [11]	1.40	SegStereo [24]	1.45

Mod.: model; **EPE**: end-point-error;

disparity maps for KITTI2015/2012 test sets and submit the results to the KITTI evaluation server for competition. The batch size is set to 4 when validating and testing the performance of both MCUA and EMCUA.

5.2. Performance on KITTI2015/2012 Datasets

Compared with Scene Flow datasets [15] which only consist of synthetic scenes, KITTI2015/2012 datasets [16, 5] contain real-world image data collected from scenes such as urban, rural, and highways, which has higher credit for the algorithm evaluation. As a result, we choose KITTI2015/2012 datasets to evaluate the contribution of applying MCUA scheme and the additional residual module to the improvement of performance.

We compare both EMCUA and MCUA with PSM-Net and other recently published approaches on KITTI 2015/2012 test sets. The evaluated results (reported by KITTI server) are illustrated in Tab. 2 and Tab. 3, respectively. EMCUA has the overall three-pixel-error of 2.09%/1.64% on KITTI2015/2012 dataset, and achieves 9.9%/13.2% decrease compared to PSM-Net, while MCUA has that of 2.14%/1.70%, and achieves 7.8%/10.1% decrease compared to PSM-Net. The results show that both EMCUA and MCUA outperform the state-of-the-art method (i.e., SegStereo), and the performance gain mainly comes from MCUA scheme. Furthermore, as shown in Tab. 2, EMCUA has the overall three-pixel-error of foreground/background of 4.27%/1.66% on KITTI2015 dataset, which achieves 2.5%/1.8% decrease compared to MCUA. It shows that the residual module is mainly used to improve the performance of the accuracy of the foreground. Furthermore, Fig. 5 and Fig. 6 illustrate some examples of final results generated by EMCUA on KITTI2015/2012 datasets, respectively.

5.3. Performance on Scene Flow Datasets

As we know, EMCUA is the updated model that is adding a residual module at the end of MCUA, which aims to enhance the performance of MCUA. To show the effect of applying MCUA scheme, we only compare MCUA with PSM-Net and other four existing approaches on Scene Flow test set. As shown in Tab. 4, the end-point-error of MCUA is 0.56 *pixels*, which has a 50% increase over PSM-Net, and outperforms the state-of-the-art approach. Two of testing examples are illustrated in Fig. 7, as shown in blue boxes, applying ultra-aggregation scheme helps the model to learn robust context information and accurately predicts disparity especially for overlapped objects.

6. Model Design Analysis

In this section, we qualitatively evaluate MCUA scheme. We first train models on the Scene Flow training datasets with 20 epochs, and then fine-tune on the KITTI2015 training set with 1000 epochs. We evaluate the resulting models on the Scene Flow validation set and KITTI2015 validation set.

6.1. Aggregation Schemes

The first experiment in Tab. 5 compares MCUA with DenseNets [8] and DLA [25] for stereo matching by replacing the 2-D CNNs branch of PSM-Net with three aggregation schemes. From Tab. 5, we can see that MCUA performs significantly better than DenseNets and DLA. We also observe that MCUA enables to learn contextual information effectively and improve the sharpness and accuracy for the disparity map (Fig. 7). Moreover, MCUA outper-

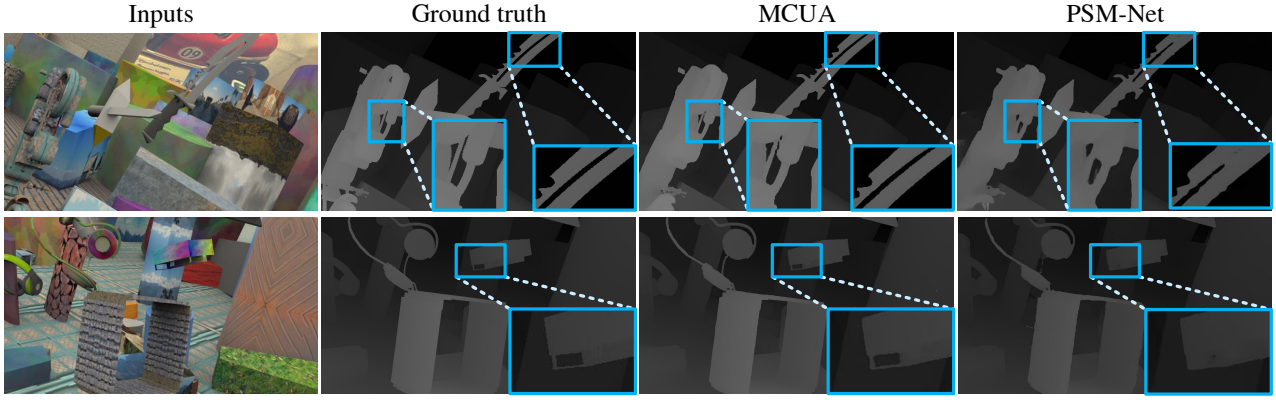


Figure 7. MCUA produces the state-of-the-art performance on Scene Flow Datasets. The left column shows the left image of the stereo images. The second shows the ground truth disparity. The third shows the prediction of our method. The fourth shows the disparity produced by PSM-Net [2].

Table 5. Ablation study

Mod.	Scene Flow				KITTI2015	Para.
	$> 1px$	$> 3px$	$> 5px$	EPE	VE (%)	
Compare of aggregation patterns						
PSM-Net	–	–	–	1.119	1.83	5.22M
DenseNets	8.526	3.329	2.286	0.794	1.698	5.27M
DLA	8.586	3.337	2.280	0.806	1.685	5.32M
MCUA	**7.885**	**3.108**	**2.148**	**0.758**	**1.579**	5.31M
Compare of architecture components						
UChi	8.185	3.153	**2.147**	**0.755**	1.635	5.39M
Chi	8.133	3.242	2.226	0.777	1.642	5.29M
DenPool	8.187	3.187	2.179	0.761	1.628	5.31M
MCUA	**7.885**	**3.108**	2.148	0.758	**1.579**	5.31M

$> t px$: EPE; **VE**: three-pixel-error; **Para.**: number of parameters.

forms the plain model by aggregating much richer contexts without significantly increasing computation burden.

6.2. Effect of MCUA

Tab. 5 shows the results of several control experiments, which are used to evaluate each part of MCUA scheme. In the first ablation study, we untie the relationship between child module and branch (a) in MCUA, which means that the branch (a) do not share parameters with child module. This new model is denoted as **UChi**. As shown in Tab. 5, although the number of parameters for Uchi increases by 0.08M after untying, the performance has no significant improvement compared with the original MCUA.

The second ablation model, **Chi**, only applies the intra-level combination on the network for the matching cost calculation in PSM-Net by removing the dashed lines but remaining the color lines in Fig. 3. As shown in Tab. 5, the performance of Chi decreases compared with the original MCUA, which indicates that the inter-level combination through child module makes an important contribution to the whole model.

The third ablation model densely connect all stages in the backbone itself. We use pooling operation to match features with different scales. The resulting architecture is represented as **DenPool**. It is clear from Tab. 5 that, using an independent child module (i.e., MCUA) is better than without using it (i.e., DenPool). Hence intra-level feature aggregation is insufficient to capture enough contextual information. However, our independent child module introduces inter-level feature aggregation, enlarges the receptive fields, captures more context information, improves the cost volume, and thus achieves better stereo matching results.

7. Conclusion

In this paper, we propose a general feature aggregation scheme, MCUA, which contains both intra- and inter-level feature aggregation, while DenseNets and DLA contain only intra-level aggregation. We formulates these models as Higher Order RNNs to clearly show this difference. We use an independent child module to introduce inter-level aggregation, which enlarges the receptive fields and captures more context information. The experimental results demonstrate the effectiveness of MCUA scheme for the context learning. Our approach outperforms the state-of-the-art methods on the Scene Flow datasets and KITTI2015/2012 benchmarks. In the future, we plan to make exploration on the improvement of soft argmin operation which is another limitation in stereo matching.

Acknowledgements. This research was supported by the National Natural Science Foundation of China (No.61731003, No.61572264), the national youth talent support program, Tianjin Natural Science Foundation (17JCJQJC43700, 18ZXZNGX00110) and the Fundamental Research Funds for the Central Universities (Nankai University, NO. 63191501).

References

[1] S. T. Barnard and M. A. Fischler. Computational stereo. *ACM Computing Surveys (CSUR)*, 14(4):553–572, 1982. 2

[2] J.-R. Chang and Y.-S. Chen. Pyramid stereo matching network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5410–5418, 2018. 1, 2, 3, 6, 7, 8

[3] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng. Dual path networks. In *Adv. Neural Inform. Process. Syst.*, pages 4467–4475, 2017. 2, 3

[4] J. Fu, J. Liu, Y. Wang, J. Zhou, C. Wang, and H. Lu. Stacked deconvolutional network for semantic segmentation. *IEEE Transactions on Image Processing*, 2019. 1

[5] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2012. 6, 7

[6] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(9):1904–1916, 2015. 2

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, 2016. 1

[8] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, volume 1, page 3, 2017. 1, 2, 7

[9] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-end learning of geometry and context for deep stereo regression. In *Int. Conf. Comput. Vis.*, pages 66–75, 2017. 1, 2, 5, 6

[10] S. Khamis, S. Fanello, C. Rhemann, A. Kowdle, J. Valentin, and S. Izadi. Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *Eur. Conf. Comput. Vis.*, pages 573–590, 2018. 2, 7

[11] Z. Liang, Y. Feng, Y. G. H. L. W. Chen, and L. Q. L. Z. J. Zhang. Learning for disparity estimation through feature constancy. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2811–2820, 2018. 1, 2, 7

[12] Q. Liao and T. Poggio. Bridging the gaps between residual learning, recurrent neural networks and visual cortex. *arXiv preprint arXiv:1604.03640*, 2016. 2

[13] M. Lin, Q. Chen, and S. Yan. Network in network. In *Int. Conf. Learn. Represent.*, pages 1–10, 2014. 4

[14] W. Luo, A. G. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5695–5703, 2016. 1, 2

[15] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4040–4048, 2016. 1, 2, 6, 7

[16] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3061–3070, 2015. 6, 7

[17] A. E. Orhan and X. Pitkow. Skip connections eliminate singularities. In *Int. Conf. Learn. Represent.*, pages 1–22, 2018. 1

[18] J. Pang, W. Sun, J. S. Ren, C. Yang, and Q. Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *IEEE Conf. Comput. Vis. Pattern Recog.*, volume 7, 2017. 2, 7

[19] H. Park and K. M. Lee. Look wider to match image patches with convolutional neural networks. *IEEE Signal Processing Letters*, 24(12):1788–1792, 2017. 2

[20] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vis.*, 47(1-3):7–42, 2002. 2

[21] A. Shaked and L. Wolf. Improved stereo matching with constant highway networks and reflective confidence learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4641–4650, 2017. 2

[22] R. Soltani and H. Jiang. Higher order recurrent neural networks. *arXiv preprint arXiv:1605.00064*, 2016. 2

[23] X. Song, X. Zhao, H. Hu, and L. Fang. Edgestereo: A context integrated residual pyramid network for stereo matching. *ACCV*, 2018. 2

[24] G. Yang, H. Zhao, J. Shi, Z. Deng, and J. Jia. Segstereo: Exploiting semantic information for disparity estimation. In *Eur. Conf. Comput. Vis.*, pages 1–16, 2018. 2, 7

[25] F. Yu, D. Wang, E. Shelhamer, and T. Darrell. Deep layer aggregation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2403–2412, 2018. 1, 2, 3, 7

[26] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4353–4361, 2015. 1, 2

[27] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1592–1599, 2015. 2

[28] J. Zbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1-32):2, 2016. 1

[29] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu. Residual dense network for image super-resolution. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2472–2481, 2018. 1