Cascade Residual Learning: A Two-stage Convolutional Neural Network for Stereo Matching

Jiahao Pang[*] Wenxiu Sun[*] Jimmy SJ. Ren Chengxi Yang Qiong Yan
SenseTime Group Limited
{pangjiahao, sunwenxiu, rensijie, yangchengxi, yanqiong}@sensetime.com

Abstract

Leveraging on the recent developments in convolutional neural networks (CNNs), matching dense correspondence from a stereo pair has been cast as a learning problem, with performance exceeding traditional approaches. However, it remains challenging to generate high-quality disparities for the inherently ill-posed regions. To tackle this problem, we propose a novel cascade CNN architecture composing of two stages. The first stage advances the recently proposed DispNet by equipping it with extra up-convolution modules, leading to disparity images with more details. The second stage explicitly rectifies the disparity initialized by the first stage; it couples with the first-stage and generates residual signals across multiple scales. The summation of the outputs from the two stages gives the final disparity. As opposed to directly learning the disparity at the second stage, we show that residual learning provides more effective refinement. Moreover, it also benefits the training of the overall cascade network. Experimentation shows that our cascade residual learning scheme provides state-of-the-art performance for matching stereo correspondence. By the time of the submission of this paper, our method ranks first in the KITTI 2015 stereo benchmark, surpassing the prior works by a noteworthy margin.

1. Introduction

Dense depth data is indispensable for reconstructing or understanding a 3D scene. Although active 3D sensors such as Lidar, ToF, or structured light can be employed, sensing depth from stereo cameras is typically a more cost-effective approach. Given a rectified stereo pair, depth can be estimated by matching corresponding pixels on the two images along the same scan-line. Particularly, for an arbitrary pixel (x, y) in left image, suppose its correspondence is found at location $(x + d, y)$ in the right image, we can compute its depth by $f \cdot l/d$, where f is the focal length, l is the baseline distance, and d is often referred to as disparity. As depth is inversely proportional to disparity, a stereo matching system is targeted to produce an accurate dense disparity instead.

从一个重要性的父命题中引出子命题，即用立体相机感知深度。

说明如何通过视差求解深度

知道了视差和深度的关系，所以立体匹配系统的目标就是产生稠密视差

Stereo matching is traditionally formulated as a problem with several stages of optimization. Until recent years with the developments in convolutional neural networks (CNNs) [18], it is cast as a learning task. Taking advantages of the vast available data, correspondence matching with CNNs achieves considerable gain compared to traditional approaches in terms of both accuracy and speed. Nevertheless, it is still difficult to find the correct correspondence at inherently ill-posed regions, such as object occlusions, repeated patterns, or textureless regions. For a pixel appearing in one image yet occluded in the other, its correspondence cannot be identified; while for repeated patterns and textureless regions, many potential correspondences exists. All these issues lead to erroneous disparity estimations.

一笔带过传统的立体匹配方法（例如四阶段），过度到CNN方法取得的进步

引入矛盾，即CNN在不适定区域也难找到正确的对应性，引出下文的解决方案

To alleviate the aforementioned problems, we propose a *cascade residual learning* (CRL) framework, composing of two stages of convolutional neural networks with hour-glass structure [5, 12]. At the first-stage network, an simple-yet-nontrivial up-convolution module is introduced to produce *fine-grained* disparities, setting up a good starting point for the residual learning at the second stage. At the second stage, the disparity is explicitly rectified with the residual signals produced at multiple scales. It is easier to learn the residual than to learn the disparity directly, similar to the mechanism of ResNet [10]. To the extreme where the initial disparity is already optimal, the second-stage network can simply generate zero residual to keep the optimality. However, the building blocks of [10]—residual blocks—are cascaded one-by-one, in which the residuals cannot be directly supervised. Different from [10] and other works along its line (*e.g.*, [4]), we embed the residual learning mechanism across multiple scales, where the residuals are explicitly supervised by the difference between the ground-truth disparity and the initial disparity, leading to *superior* disparity refinement.

介绍本文的算法

The proposed CRL scheme is trained end-to-end, integrating the traditional pipeline [24] from matching cost

本算法的特点和优势

[*] Both authors contributed equally.

computation, cost aggregation, disparity optimization, to disparity refinement by a stack of non-linear layers. The two stages of CRL boost the performance together and achieve state-of-the-art stereo matching results. It *ranks first* in the KITTI 2015 stereo benchmark [21].

Our paper is structured as follows. We review related works in Section 2. Then we elaborate our CRL framework and discuss our network architecture in Section 3. In Section 4 and Section 5, experimentation and conclusions are presented respectively.

2. Related Works

指名要将评述的重点放在近期采用CNNs的立体匹配方法上

There exist a large body of literature on stereo matching. We hereby review a few of them, with emphasis placed on those recent methods employing convolutional neural networks (CNNs).

由典型立体匹配的四步骤，引出CNN方法包含上述步骤的部分或全部。并说明CNN方法分为三类，引出下文

A typical stereo matching algorithm, *e.g.,* [11, 26], consists of four steps [24]: 1) matching cost computation; 2) cost aggregation; 3) disparity optimization (derive the disparity from the cost volume); and 4) disparity refinement (post-process the disparity). In contrast, CNN-based approaches estimate disparities reflecting part or all of the aforementioned four steps. These approaches can be roughly divided into the three categories.

Matching cost learning: In contrast to hand-crafted matching cost metrics, such as sum of absolute difference (SAD), normalized cross correlation (NCC) and Birchfield-Tomasi cost [1], CNNs are utilized to measure the similarity between image patches. Han *et al.* [9] presented a Siamese network called MatchNet, which extracts features from a pair of patches followed by a decision module for measuring similarity. Concurrently, Zagoruyko *et al.* [28] and Zbontar *et al.* [29] investigated a series of CNN architectures for binary classification of pairwise matching and applied in disparity estimation. In contrast to an independent binary classification scheme between image patches, Luo *et al.* [19] proposed to learn a probability distribution over all disparity values. This strategy employs a diverse set of training samples without concerning about the unbalanced training samples. Though the data-driven similarity measurements out-perform the traditional hand-crafted ones, a number of post-processing steps (*e.g.,* steps 2) to 4) in the traditional stereo matching pipeline) are still necessary for producing compelling results.

CNN的匹配代价学习好于传统方法， 但后处理步骤依旧必不可少，此为一矛盾

Regularity learning: Based on the observation that disparity images are generally piecewise smooth, some existing works imposes smoothness constraints in the learning process. Menze *et al.* [21] applied adaptive smoothness constraints using texture and edge information for a dense stereo estimation. By discovering locally inconsistent labeled pixels, Gidaris *et al.* [6] propose the detect, replace, refine framework. However, discarding unreliable disparities with new ones results in a wasted computation. Disparity can also be regularized by incorporating with mid- or high-level vision tasks. For instance, disparity was estimated concurrently by solving the problem of semantic segmentation, *e.g.*, [2, 17, 27]. Guney and Geiger raised Displets in [8], which utilizes object recognition and semantic segmentation for finding stereo correspondence.

对于正则化学习，认识不够，需要阅读部分文献。

End-to-end disparity learning: By carefully designing and supervising the network, a fine disparity is able to be end-to-end learned with stereo inputs. Mayer *et al.* [20] presented a novel approach called DispNet, where an end-to-end CNN is trained using synthetic stereo pairs. In parallel with the proposal of DispNet, similar CNN architectures are also applied to optical flow estimation, leading to FlowNet [5] and its successor, FlowNet 2.0 [12]. A very recent method, GC-NET [15], manages to employ contextual information with 3D convolutions for learning disparity. For monocular depth estimation, end-to-end semi-supervised [16] and unsupervised [7] approaches were also proposed, which connect stereo images with the estimated disparity and require a very limited amount of training data that has ground-truth disparity.

Our work belongs to the third category. In spite of the superior performance of the CNN-based approaches, it remains very challenging to produce accurate disparities at ill-posed regions. Unlike existing works, we present a cascade residual learning scheme to tackle the aforementioned problem. Particularly, we adopt a two-stage CNN, in which the first stage delivers a high-quality initialized disparity map. After that, the second stage performs further refinement/rectification by producing residual signals across multiple scales. Our experimentation shows that, the proposed cascade residual learning scheme provides state-of-the-art disparity estimates with an acceptable runtime.

3. Cascade Residual Learning

This section illustrates our cascade residual learning (CRL) scheme in detail.

3.1. Two-stage Disparity Computation

In general, low-level vision tasks, *e.g.*, denoising and deblurring, can be improved with post-facto iterative refinement [22], and disparity/flow estimation is no exception [3]. Recently, Ilg *et al.* [12] introduced FlowNet 2.0, which uses stacking CNNs for optical flow refinement and achieves reasonable gain. The lessons of the previous works inspire us to employ a two-stage CNN for disparity estimation.

Akin to the proposal of DispNetC ("C" indicates the network has a correlation layer) [20], the first stage of our CNN has an hour-glass structure with skip connections. However, DispNetC outputs disparity image at half the resolution of the input stereo pair. Differently, our network includes extra deconvolution modules to magnify the disparity, leading to disparity estimates at the same size of the input images. We

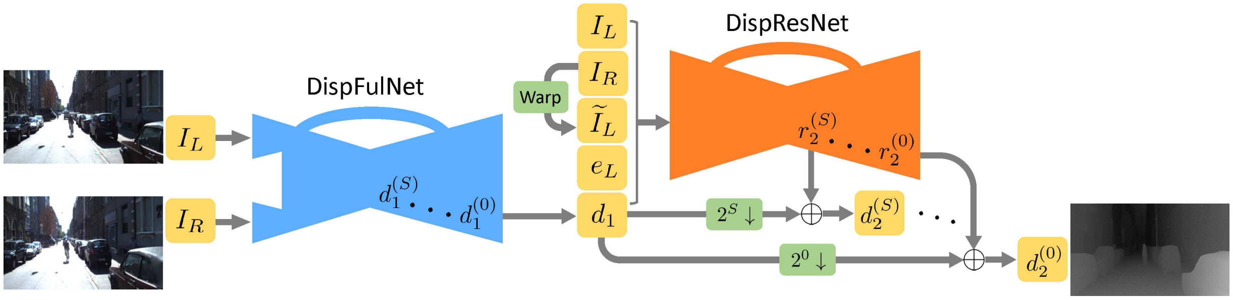


Figure 1. Network architecture of our *cascade residual learning* (CRL) scheme. The first stage is *DispFulNet* and the second stage is *DispResNet* with multiscale residual learning. The module "$2^s \downarrow$" is the downsampling layer to shrink d_1 for 2^s times, while "Warp" denotes the warping layer.

call our first stage network *DispFulNet* ("Ful" means full-resolution). As shown later in Section 4, our DispFulNet provides extra details and sharp transitions at object boundaries, serving as an ideal starting point for the second-stage refinement.

Note that in our network, the two stages are cascaded in a way recommended by [12]. Specifically, the first network takes as input the stereo pair I_L and I_R and produces the initial disparity d_1 (of the left image). We then warp the right image I_R according to disparity d_1 and obtain a synthesized left image, *i.e.*,

$$\widetilde{I}_L(x, y) = I_R(x + d_1(x, y), y). \tag{1}$$

Then the input to the second network is the concatenation of $I_L, I_R, d_1, \widetilde{I}_L(x, y)$ and the error $e_L = |I_L - \widetilde{I}_L(x, y)|$. The warping operation is differentiable for bilinear interpolation [12, 13], hence our network can be trained end-to-end.

3.2. Mutiscale Residual Learning

For the second-stage refinement/rectification, we propose to adopt the residual learning scheme of He *et al.* [10]. Particularly, given the initial disparity d_1 obtained with the first stage, the second network outputs the corresponding residual signal r_2, then the new disparity d_2 is given by $d_1 + r_2$. In this way, we relieve the "burden" of the second-stage network, letting it only focus on learning the highly nonlinear residual. On par with the spirit in [10], in the extreme case when the first stage already produces the optimal disparity, the second-stage network only needs to output zero residual to retain the optimality.

The second-stage of our architecture also takes an hourglass structure, producing residual signals across multiple scales. We call our second-stage network *DispResNet* ("Res" means residual). In the expanding part of DispResNet, the residuals are produced across several scales. They are denoted as $\{r_2^{(s)}\}_{s=0}^{S}$ where 0 denotes the scale of full resolution. The summation of $r_2^{(s)}$ with the downsampled disparity $d_1^{(s)}$ leads to the new disparity at scale s, *i.e.*,

$$d_2^{(s)} = d_1^{(s)} + r_2^{(s)}, 0 \leq s \leq S. \tag{2}$$

To train DispResNet, we supervise the estimated disparities $\{d_2^{(s)}\}_{s=0}^{S}$ across $S + 1$ scales. Hence, differs from the off-the-shelf residual block structure proposed in [10], our network explicitly supervise the residual signals, leading to effective disparity refinement.

In fact, a straightforward application of FlowNet 2.0 [12] for disparity estimation is to adopt DispNetS [20]—a variation of DispNetC without correlation layer and "S" means simple—to *directly* learn the disparity. Nevertheless, our comparisons in Section 4 show that incorporating residual learning brings more gain than its direct learning counterpart, *i.e.*, DispNetS. Furthermore, residual learning also benefits the finetuning of the overall network, as it alleviates the problem of over-fitting [10, 12], while using DispNetS harms the performance after overall finetuning.

3.3. Network Architecture

Our CRL architecture is illustrated in Fig. 1, where $d_1 = d_1^{(0)}$, and the final disparity output is $d_2^{(0)}$. To obtain the downsampled disparity images $\{d_1^{(s)}\}_{s=0}^{S}$, we have implemented a differentiable bilinear downsampling layer, similar to the sampler module in the spatial transformer networks [13].

The first stage, DispFulNet, enlarges the half-resolution disparity estimates of DispNetC [20]. For a concise presentation, the detailed architecture of DispFulNet is not provided here. In general, it shares similar spirits with DispNetC. Though differently, we append extra up-convolutions to the last two convolution layers of DispNetC, the output of the upconvolutions are then concatenated with the left image. By applying one more convolution (with one output channel) to the concatenated 3-D array, we arrive at the output of DispFulNet—a full-resolution disparity image. The full-resolution disparity image, along with the other intermediate disparity images at six different scales, are super-

Layer	K	S	Channels	I	O	Input Channels
conv1	5	1	13/64	1	1	left+right+left_s+err+pr_s1
conv2	5	2	64/128	1	2	conv1
conv2_1	3	1	128/128	2	2	conv2
conv3	3	2	128/256	2	4	conv_3_1
conv3_1	3	1	256/256	4	4	conv3
conv4	3	2	256/512	4	8	conv3_1
conv4_1	3	1	512/512	8	8	conv4
conv5	3	2	512/1024	8	16	conv4_1
conv5_1	3	1	1024/1024	16	16	conv5
res_16	3	1	1024/1	16	16	conv5_1
pr_s1_16	-	-	1/1	1	16	pr_s1
pr_s2_16	-	-	1/1	16	16	pr_s1_16+res_16
upconv4	4	2	1024/512	16	8	conv5_1
iconv4	3	1	1025/512	8	8	upconv4+conv4_1+pr_s2_16
res_8	3	1	512/1	8	8	iconv4
pr_s1_8	-	-	1/1	1	8	pr_s1
pr_s2_8	-	-	1/1	8	8	pr_s1_8+res_8
upconv3	4	2	512/256	8	4	iconv4
iconv3	3	1	513/256	4	4	upconv3+conv3_1+pr_s2_8
res_4	3	1	256/1	4	4	iconv3
pr_s1_4	-	-	1/1	1	4	pr_s1
pr_s2_4	-	-	1/1	4	4	pr_s1_4+res_4
upconv2	4	2	256/128	4	2	iconv3
iconv2	3	1	257/128	2	2	upconv2+conv2_1+pr_s2_4
res_2	3	1	128/1	2	2	iconv2
pr_s1_2	-	-	1/1	1	2	pr_s1
pr_s2_2	-	-	1/1	2	2	pr_s1_2+res_2
upconv1	4	2	128/64	2	1	iconv2
res_1	5	1	129/1	1	1	upconv1+conv1+pr_s2_2
pr_s2	-	-	1/1	1	1	pr_s1+res_1

Table 1. Detailed architecture of the proposed *DispResNet*. Layers with prefix `pr_s1` are downsampling layers applying on the predictions of the first stage; while layers with prefix `pr_s2` are element-wise summation layers leading to predictions of the second stage. **K** means kernel size, **S** means stride, and **Channels** is the number of input and output channels. **I** and **O** are the input and output downsampling factor relative to the input. The symbol + means summation for element-wise summation layers; otherwise it means concatenation.

vised by the ground-truth through computing the ℓ_1 loss.

The detailed specification of the second stage, DispResNet, is provided in Table 1. Note that at a certain scale, say, $1/4$, the bilinear downsampling layer `pr_s1_4` shrinks `pr_s1`, the disparity prediction of DispFulNet, by a factor of 4. The downsampled disparity is then added to the learned residual `res_4` by the element-wise summation layer `pr_s2_4`, leading to the disparity prediction at scale $1/4$. We follow the typical supervised learning paradigm and compute an ℓ_1 loss between the disparity estimate and the ground-truth disparity at each scale.

One may raise a straightforward question about our design: if a two-stage cascade architecture performs well, why not stacking more stages? First, adding more stages translates to higher computational cost and memory consumption, which is unrealistic for many practical applications. Second, in this paper, we aim at developing a two-stage network, where the first one manages to produce full-resolution initializations; while the second stage tries its best to refine/remedy the initial disparities with residual learning. The two stages play their own roles and couple with each other to provide satisfactory results. As to be seen in Section 4.3, our two-stage network estimate high-quality disparity images with an acceptable execution time: it takes 0.47 sec with an Nvidia GTX 1080 GPU to obtain a disparity image in the KITTI 2015 stereo dataset.

Target Dataset	Training Schedule	
	Separate	Overall
FlyingThings3D	`1F-2F`	`1F-2F-0F`
Middlebury	`1F-2F`	`1F-2F-0F`
KITTI	`1F-1K-2F-2K`	`1F-1K-2F-2K-0K`

Table 2. Training schedules of a two-stage network with different target datasets. When overall finetuning is adopted, the whole network is finetuned on the target dataset at the end.

4. Experiments

Experimental setup and results are presented in this section. To evaluate the effectiveness of our design, we replace the two stages of our network with the plain DispNetC and/or DispNetS [20] for comparisons. We also compare our proposal with other state-of-the-art approaches, *e.g.*, [27, 29].

4.1. Experimental Settings

Datasets: Three publicly available datasets are adopted for training and testing in this work:

(i) *FlyingThings3D* [20]: a large scale synthetic dataset containing more than 22k synthetic stereo pairs for training and 4k for testing. We found this dataset has a few images with unreasonably large disparities (*e.g.*, greater than 10^3), therefore we perform a simple screening on this dataset before using it. Particularly, for a disparity image, if more than 25% of its disparity values are greater than 300, this disparity image (and the corresponding stereo pair) is removed.

(ii) *Middlebury 2014* [23]: a small dataset capturing various high-resolution in-door scenes, which has 23 stereo pairs with given ground-truth. We only use this dataset for testing.

(iii) *KITTI 2015* [21]: a real-world dataset with dynamic street views from the perspective of a driving car. It provides 200 stereo pairs with sparse ground-truth disparities and 200 pairs for evaluation through its online leaderboard. Similar to the practice in [6], we divide its training set into a training split and a validation split, where the training split occupies 85% of the data and the validation split occupies the rest.

Architecture		Dataset											
Stage 1	Stage 2	FlyingThings3D				Middlebury 2014				KITTI 2015			
		Separate		Overall		Separate		Overall		Separate		Overall	
DispNetC	-	1.84	9.67	-	-	1.95	15.42	-	-	0.77	3.16	-	-
DispNetC	DispNetS	1.74	7.98	1.77	9.04	1.94	14.72	1.96	14.91	0.75	2.71	0.78	2.82
DispNetC	DispResNet	1.63	7.76	1.60	7.67	1.86	14.69	1.88	14.71	0.72	2.66	0.71	2.63
DispFulNet	-	1.75	8.61	-	-	1.73	12.82	-	-	0.73	2.41	-	-
DispFulNet	DispNetS	1.51	6.93	1.53	7.09	1.52	9.69	1.51	10.04	0.72	2.29	0.73	2.33
DispFulNet	DispResNet	1.35	6.34	**1.32**	**6.20**	1.46	9.35	**1.40**	**9.13**	0.69	2.12	**0.68**	**2.10**

Table 3. Comparisons of our CRL architecture (DispFulNet+DispResNet) with other similar networks. In each cell, the corresponding endpoint-error (EPE) and three-pixel-error (3PE) are presented, respectively.

Training: The Caffe framework [14] is used to implement our CRL scheme. Generally speaking, we first train the DispFulNet, then by fixing its weights, the DispResNet is trained. After that, we optionally finetune the overall network. Depending on the targeting dataset for testing, different training schedules are employed. For presentation, we hereby encode every training schedule with a string. A segment of such string contains two characters `ND`, meaning that stage `N` is trained on dataset `D`, with stage `0` denotes the whole network. For instance, `1F-1K` means the first stage is trained on the FlyingThings3D, then it is finetuned on KITTI. The training schedules for the three datasets are presented in Table. 2. Note that the networks trained for FlyingThings3D are directly applied on the Middlebury data (at the quarter scale).

We adopt a batch size of 4 when training the first or the second stage, and a batch size of 2 when finetuning the overall network due to limited GPU memory. We employ the parameters provided in [20] when training the first stage or the second stage on the FlyingThings3D dataset. During finetuning, we train the model for 200 K iterations; however, when the target dataset is KITTI 2015, we only optimize for 100 K iterations to lessen the problem of over-fitting. Since some of the ground-truth disparities are not available for the KITTI dataset, we neglect them when computing the ℓ_1 loss.

Testing: We test our networks on the aforementioned datasets, with two widely used metrics for evaluation:

(i) *Endpoint-error* (EPE): the average Euclidean distance between the estimated disparity and the ground-truth.

(ii) *Three-pixel-error* (3PE): computes the percentage of pixels with endpoint error more than 3. We call it three-pixel-error in this work.

4.2. Architecture Comparisons

We first compare our design with several similar network architectures. Particularly, we use either DispNetC or DispFulNet as the first-stage network; while at the second stage, we use either DispNetS (with direct learning) or DispResNet (with residual learning) for improving the disparity estimates. The plain DispNetC and DispFulNet (with only one stage) are also considered in our evaluation. For DispNetC, we adopted the model released by Dosovitskiy *et al.* [5]; while DispFulNet are trained in a similar manner as that in [5] (*e.g.*, with multi-scale loss functions). During the training process, we follow the schedules shown in Table 2, hence 20 different network models are obtained for comparisons.

Objective performance of the networks on the three datasets are presented in Table. 3. We have the following observations:

(i) Using our DispFulNet as the first stage provides *higher accuracy* compared to DispNetC.

(ii) Though appending a second-stage network improves the results, our DispResNet bring *extra gain* compared to DispNetS (the propposal in [12]).

(iii) When DispNetS is served as the second stage, the performance deteriorates after overall finetuning, in accordance with [12]. In contrast, when DispResNet is used, overall optimization further *improves* the performance in most cases (except for Middlebury which is not used for training). From [10], that is because learning the residual is less easy to over-fit the training data, making the network more stable for overall optimization.

As a whole, our CRL scheme (DispFulNet+DispResNet) with overall finetuning achieves the best objective qualities in all the three datasets. In the following, we use this network model for further comparisons.

Fig. 2 shows the outputs of our CRL scheme and its first stage, DispFulNet, as well as their absolute differences between the ground-truth disparities. The three rows are segments taken from the FlyingThings3D, Middlebury and KITTI datasets, respectively. We see that not only the disparities at object boundaries are greatly improved by the second-stage (DispResNet), some of the occlusion and textureless regions are also rectified. For instance, the regions

Metric	SGM	SPS-St	MC-CNN-fst	DispNetC	CRL
EPE	4.50	3.98	3.79	1.84	1.32
3PE	12.54	12.84	13.70	9.67	6.20

Table 4. Objective performance of our work (CRL), along with those of the competing methods on the FlyingThings3D dataset.

within the red boxes (on the ground-truth) are corrected by DispResNet.

Fig. 3 shows the disparity estimates of three different two-stage networks: DispNetC+DispNetS (akin to the proposal of [12]), DispNetC+DispResNet, and DispFulNet+DispResNet (our CRL), where DispNetC+DispNetS uses the model with separate training while DipsNetC+DispResNet uses the model after overall finetuning. Again, the three rows are segments taken from the FlyingThings3D, Middlebury and KITTI datasets, respectively. We see that, firstly, the proposed CRL provides sharpest disparity estimates among the three architectures, with the help of its first stage, DispFulNet. Furthermore, incorporating residual learning in the second stage produces high-quality disparities for ill-posed regions. Note the disparity estimates within the red boxes are progressively improved from DispNetC+DispNetS and DispNetC+DispResNet, to CRL.

4.3. Comparisons with Other Methods

In this experiment, we compare the proposed CRL to several state-of-the-art stereo matching algorithms. For a fair comparison, the Middlebury dataset is not adopted in this experiment as its amount of data is insufficient for finetuning our end-to-end network.

FlyingThings3D: Since our method only takes 0.47 second to process a stereo pair in the KITTI 2015 dataset, for a fair comparison, we hereby consider three efficient yet effective methods (with code publicly available), including SPS-St [27], MC-CNN-fst [29], and DispNetC [20]. We also employ the classic semi-global matching (SGM) algorithm [11] as the baseline. Note that to compare with MC-CNN-fst, we train its network for 14 epochs, with a dataset containing 17 million samples extracted from the FlyingThings3D.

Performance of the proposed CRL, along with those of the competing methods, are presented in Table. 4. Again, we see that our approach provides the best performance in terms of both evaluation metrics. In Fig. 4, we show some visual results of different approaches on the FlyingThings3D dataset, note that our CRL provides very sharp disparity estimates. Besides, our method is the only one that can generate the fine details within the red boxes.

KITTI 2015 dataset: Instead of using the training split mentioned in Section 4.1, we have also trained our network on all available training data of KITTI 2015 and submitted our results to its online leaderboard. Table 5 shows the leading submission results reported by the KITTI website, where only the three-pixel-error (3PE) values are available. In the table, “All” means all pixels are taken into account when computing 3PE, while “Noc” means only the non-occluded pixels are taken into account. The three columns “D1-bg,” “D1-fg” and “D1-all” means the 3PE of the background, the foreground and the all the estimates. As can be seen, our method *ranks first* in the online leaderboard. Particularly, our overall 3PE is 2.67%, while the second method, GC-NET [15], has a 3PE of 2.87%; however, our runtime is only about half of that of GC-NET. Visual results are not included here for conciseness, we recommend the readers go to the KITTI website [21] for more details.

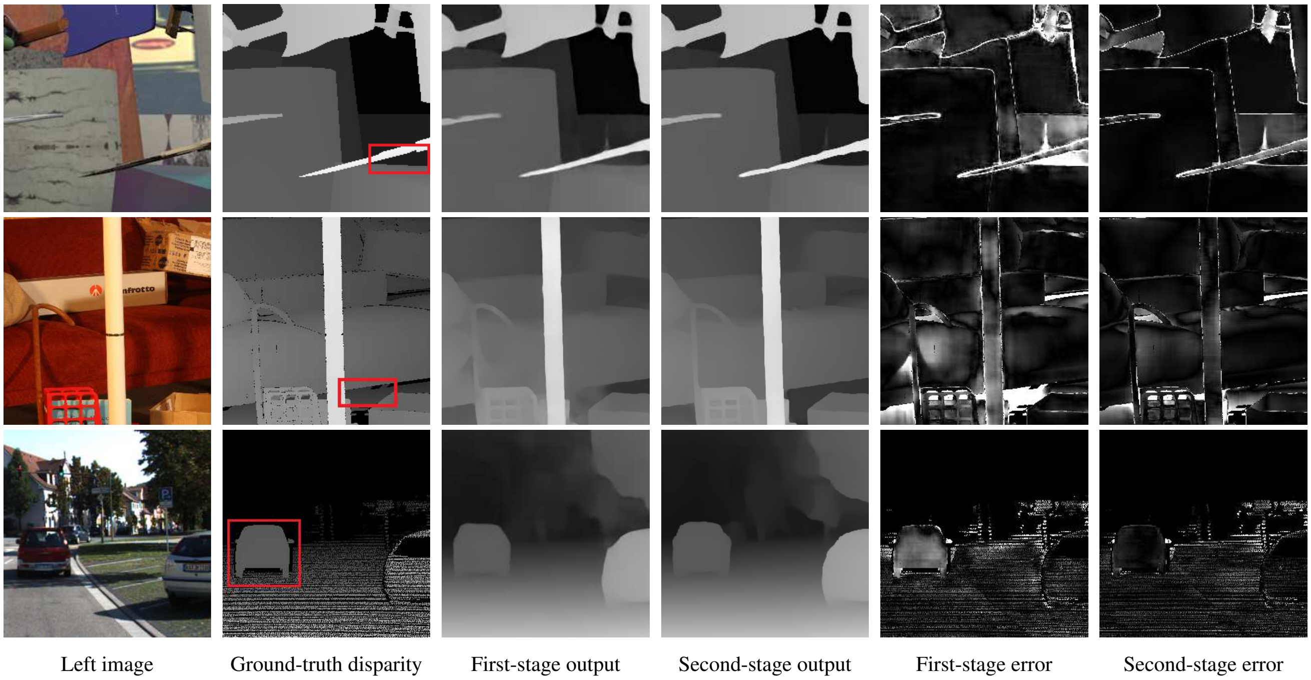
4.4. Discussions

Existing end-to-end CNNs for stereo matching, *e.g.*, [15, 20] and this work, all relies on a vast amount of training data with ground-truth. However, it is costly to collect depth data in the real physical world; while synthetic data, *e.g.*, the FlyingThings3D dataset, cannot fully reflects the properties of the real environment.

A potential solution to the above dilemma is to borrow the wisdom from traditional approaches and embed the left-right consistency check module into the CNNs. As mentioned in Section 2, it is explored by [7, 16] for monocular depth estimation, leading to unsupervised (or semi-supervised) method requiring (very) little amount of data with ground-truth. However, recent end-to-end CNN-based approaches already produces very accurate disparity estimates, in contrast to the case of monocular depth estimation. As a result, any new mechanisms (*e.g.*, left-right consistency check in this case) introduced to the networks need to be very reliable/robust, otherwise further improvements cannot be achieved. We leave this problem of designing robust left-right consistency check module for future investigation.

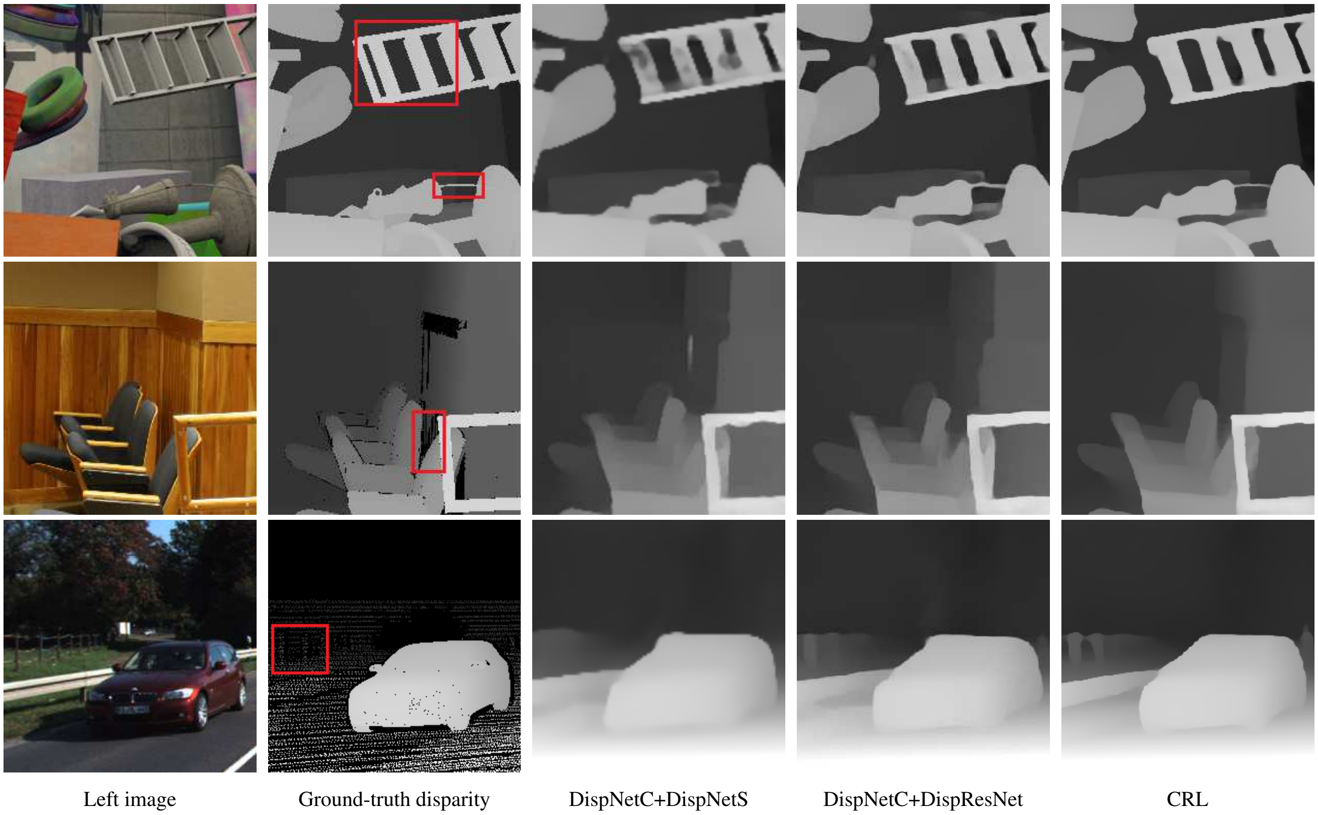
5. Conclusions

Recent works employing CNNs for stereo matching have achieved prominent performance. Nevertheless, estimating high-quality disparity for inherently ill-posed regions remains intractable. In this work, we propose a cascade CNN architecture with two stages: the first stage manages to produce an initial disparity image with fine details, while the second stage explicitly refines/rectifies the initial disparity with residual signals across multiple scales. We call our approach cascade residual learning. Our experiments show that, residual learning not only provides effective refinement but also benefits the optimization of the whole two-stage network. Our approach achieves state-of-the-art stereo matching performance, it ranks first in the KITTI



Left image Ground-truth disparity First-stage output Second-stage output First-stage error Second-stage error

Figure 2. Visual comparisons between the first-stage output by DispFulNet and the second-stage output by the whole CRL scheme (DispFulNet+DispResNet). Note that the regions within the red boxes are corrected by DispResNet.



Left image Ground-truth disparity DispNetC+DispNetS DispNetC+DispResNet CRL

Figure 3. Comparisons of three two-stage network architectures. Our proposed CRL deliveries sharpest and finest disparity images. Also note the regions bounded by the red boxes in different disparity images.

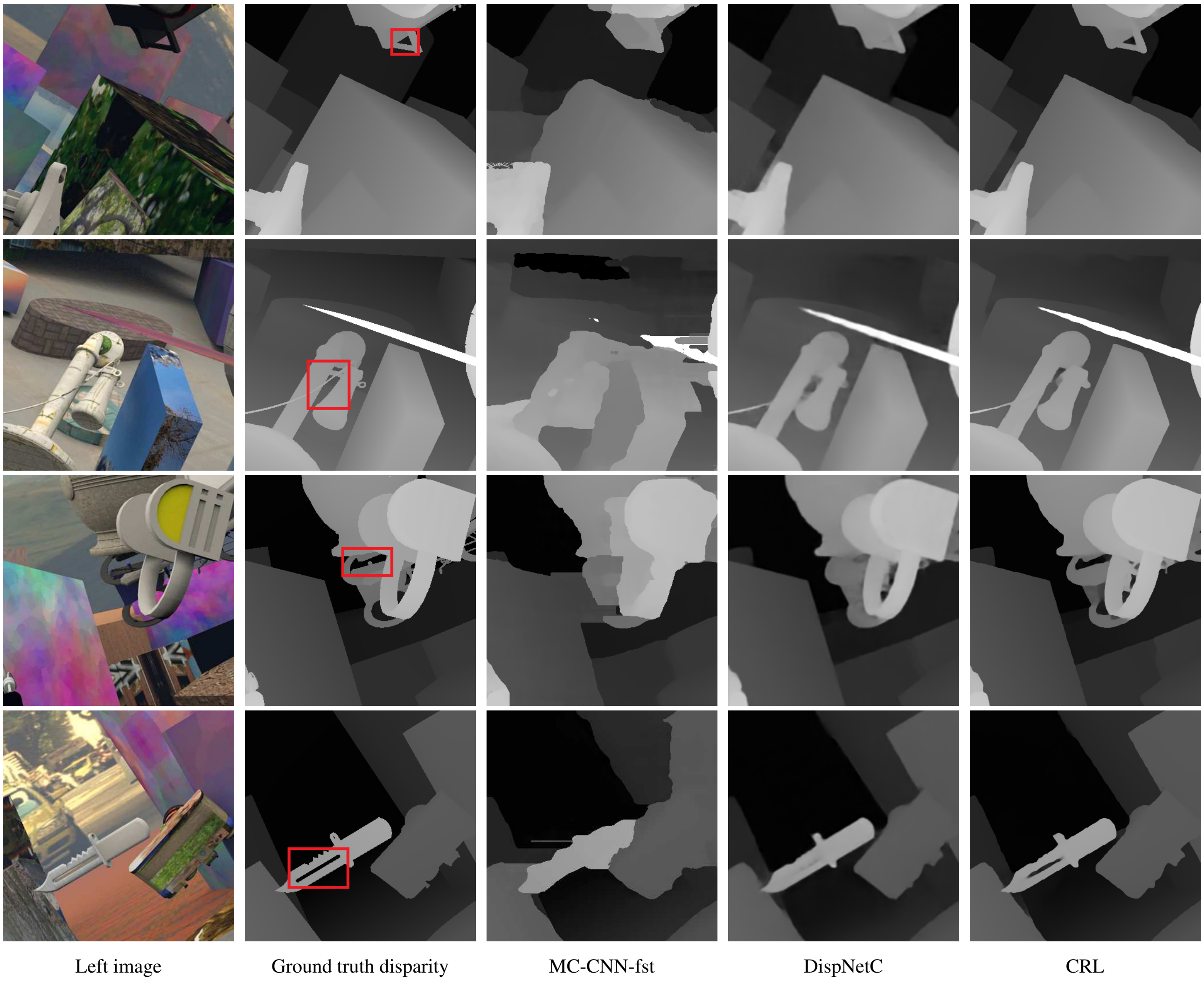


Figure 4. Visual results of the proposed CRL, accompanied with those of the competing methods, on the FlyingThings3D dataset. Our method is the only one that successfully estimates the details within the red boxes.

Methods	All			Noc			Runtime (sec)
	D1-bg	D1-fg	D1-all	D1-bg	D1-fg	D1-all	
CRL (Ours)	2.48	**3.59**	**2.67**	2.32	**3.12**	**2.45**	0.47
GC-NET [15]	**2.21**	6.16	2.87	**2.02**	5.58	2.61	0.9
DRR [6]	2.58	6.04	3.16	2.34	4.87	2.76	0.4
L-ResMatch [25]	2.72	6.95	3.42	2.35	5.74	2.91	48*
Displets v2 [8]	3.00	5.56	3.43	2.73	4.95	3.09	265*
D3DNet	2.88	6.60	3.50	2.71	6.08	3.26	**0.35**
SsSMNet	2.86	7.12	3.57	2.63	6.26	3.23	0.8

Table 5. Leading submissions of the KITTI 2015 stereo online leaderboard (as of August 2017). Three-pixel-error of our approach and the other state-of-the-art methods are tabulated, where our approach ranks first. The symbol "*" denotes runtime on CPU.

2015 stereo benchmark, exceeding the prior works by a noteworthy margin.

References

[1] S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(4):401–406,

1998. 2

[2] M. Bleyer, C. Rother, P. Kohli, D. Scharstein, and S. Sinha. Object stereojoint stereo matching and object segmentation. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3081–3088, 2011. 2

[3] T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 33(3):500–513, 2011. 2

[4] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4733–4742, 2016. 1

[5] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proc. of the IEEE International Conference on Computer Vision*, pages 2758–2766, 2015. 1, 2, 5

[6] S. Gidaris and N. Komodakis. Detect, replace, refine: Deep structured prediction for pixel wise labeling. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5248–5257, 2017. 2, 4, 8

[7] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 279–270, 2016. 2, 6

[8] F. Guney and A. Geiger. Displets: Resolving stereo ambiguities using object knowledge. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4165–4175, 2015. 2, 8

[9] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3279–3286, 2015. 2

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 1, 3, 5

[11] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008. 2, 6

[12] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2462–2470, 2017. 1, 2, 3, 5, 6

[13] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015. 3

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proc. of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014. 5

[15] 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. *arXiv preprint arXiv:1703.04309*, 2017. 2, 6, 8

[16] Y. Kuznietsov, J. Stückler, and B. Leibe. Semi-supervised deep learning for monocular depth map prediction. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6647–6655, 2017. 2, 6

[17] L. Ladickỳ, P. Sturgess, C. Russell, S. Sengupta, Y. Bastanlar, W. Clocksin, and P. H. Torr. Joint optimization for object class segmentation and dense stereo reconstruction. *International Journal of Computer Vision*, 100(2):122–133, 2012. 2

[18] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 1

[19] W. Luo, A. G. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5695–5703, 2016. 2

[20] 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 *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4040–4048, 2016. 2, 3, 4, 5, 6

[21] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 2, 4, 6

[22] P. Milanfar. A tour of modern image filtering: New insights and methods, both practical and theoretical. *IEEE Signal Processing Magazine*, 30(1):106–128, 2013. 2

[23] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition*, pages 31–42. Springer, 2014. 4

[24] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002. 1, 2

[25] A. Shaked and L. Wolf. Improved stereo matching with constant highway networks and reflective confidence learning. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4641–4650, 2017. 8

[26] J. Sun, N.-N. Zheng, and H.-Y. Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003. 2

[27] K. Yamaguchi, D. McAllester, and R. Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *European Conference on Computer Vision*, pages 756–771. Springer, 2014. 2, 4, 6

[28] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4353–4361, 2015. 2

[29] 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. 2, 4, 6