Unsupervised Learning of Stereo Matching*

Chao Zhou[1] Hong Zhang[1] Xiaoyong Shen[2] Jiaya Jia[1,2]
[1]The Chinese University of Hong Kong [2]Youtu Lab, Tencent
{zhouc, zhangh}@cse.cuhk.edu.hk dylanshen@tencent.com leojia9@gmail.com

Abstract

Convolutional neural networks showed the ability in stereo matching cost learning. Recent approaches learned parameters from public datasets that have ground truth disparity maps. Due to the difficulty of labeling ground truth depth, usable data for system training is rather limited, making it difficult to apply the system to real applications. In this paper, we present a framework for learning stereo matching costs without human supervision. Our method updates network parameters in an iterative manner. It starts with a randomly initialized network. Left-right check is adopted to guide the training. Suitable matching is then picked and used as training data in following iterations. Our system finally converges to a stable state and performs even comparably with other supervised methods.

1. Introduction

Stereo matching is a fundamental problem in computer vision for estimating depth from digital images. As shown in Figure 1, stereo matching takes two images of the same scene after rectification and outputs the disparity map by computing relative displacement of pixels between these two images. Finally, relative depth can be obtained according to the disparity map.

Along the pipeline of stereo matching, computing and aggregating the matching cost are the two key steps [10]. Several methods are developed to estimate distances varying from pixel space to feature space. Recently, convolutional neural networks (CNN) were exploited to learn the matching cost of two image patches for stereo vision [33, 8, 3, 18, 32]. These approaches treat the matching-cost problem as a binary classification one. Given two patches, the network predicts whether the two patches match or not. The probability is directly taken as the initial matching cost. Afterwards, cost aggregation updates local information according to image structure.

*This work is in part supported by a grant from the Research Grants Council of the Hong Kong SAR (project No. 413113).

Figure 1: Stereo matching approaches take left and right images as input as shown in (a) and (b), and output the disparity map (c).

Problems of Current CNNs Current CNN-based approaches improve performance in stereo matching in challenging public benchmark [6, 28, 23]. However, they do not produce similarly satisfying results for real-life photos. It is due to the limitation of currently available training data. First, most datasets are of specific scenes. For example, the KITTI dataset is for autonomous driving and all image pairs are taken with street views. Networks trained on them can hardly perform similarly well in general scenes.

Second, sizes of datasets with real photos are small. Large-scale depth data [20, 5] are mostly synthesized from 3D models. Images in these datasets are different from real-world images in appearance and structure. On the other hand, collecting high-accuracy depth data from stereo image pairs in various scenarios is very difficult, because of involvement of 3D sensors such as light detection and ranging (LIDAR). In short, the insufficient-data problem largely hinders this community from developing more advanced solutions or making better use of learning in stereo matching.

Our Contribution To address the data limitation problem, we in this paper propose an unsupervised learning scheme for stereo matching, which enables learning for diverse scenarios without labeled matching ground truth. As shown in Figure 3, our approach takes rectified left and right images without ground truth disparity as input, and predicts the disparity map.

The framework starts from a random disparity map and updates network parameters in an iterative manner. To better learn the disparity map, we develop a new network architecture as illustrated in Figure 2. The left and right images are fed into the cost-volume branch (CVB) to produce the cost-volume. To incorporate color information from the input images, we extract feature by the image-feature branch (IFB). Afterwards, each channel of the cost-volume is concatenated with the input image feature, processed by the joint filtering branch (JFB) for cost aggregation. We finally produce the disparity map using the Soft Argmax operator from the aggregated cost-volume.

Experiments show that our framework is suitable for matching data from different scenarios and achieves reasonable performance. The major contributions are threefold.

- We learn stereo matching without matching ground truth.
- We develop a CNN framework with branches for specific tasks in stereo matching learning.
- We conduct evaluation to show that our unsupervised learning method accomplishes comparable results with other strong-supervising schemes.

2. Related Work

We in this section review related stereo matching schemes and deep learning techniques.

Traditional Stereo Matching Early stereo matching methods [12, 13] refine initial matching costs computed with various metrics such as Euclidean distance of pixel values. Fitting hyper-parameters of graphical models [16, 26, 34] with ground truth data was also popular. Estimating confidence of computed matching costs is studied in recent work [7, 27]. These methods train a random forest classifier to either combine several confidence measures or integrate with a Markov random field.

CNN-based Matching Cost Learning Convolutional neural network (CNN) has also been widely applied to matching cost learning. Zbontar and LeCun [33] formulated matching as a binary classification problem to determine if two patches match using the network. The siamese architecture contains two branches with the same structure and sharing the same set of weights. Later methods followed this work improved network architecture. MatchNet [8] adopts AlexNet [14]. Pooling layers are used to reduce feature dimension and increase robustness against scale variation. Chen *et al*. [3] solved the scale problem where two siamese networks receiving the same patch pairs with different scales produce matching scores respectively. The final score is weighted average of the two.

To speed up training and testing, Luo *et al*. [18] simplified the metric network as an inner product layer. Zagoruyko and Komodakis [32] conducted study on the performance of different network architectures in matching cost computation. Dosovitskiy *et al*. [5] proposed an end-to-end optical flow estimation network by introducing the correlation layer that produces cost-volume given feature representation of each image patch. The performance of these methods is highly dependent of their training data.

Different from all above stereo matching methods, we explore the possibility of training matching cost networks without human supervision.

Unsupervised Deep Learning Unsupervised learning is popular in deep learning and was applied to video prediction [19], auto-encoder [11, 24, 25], visual representation [4, 29], to name a few. There are also nice work of unsupervised learning of edges [17] and optical flow [31]. Li *et al*. [17] exploited the relevance of motion boundaries and object edges. Alternating between motion estimation and edge detection forms the basic step of edge learning. Yu *et al*. [31] defined a loss function with data and smoothing terms analogous to objective functions in energy minimization schemes. Thus this method is up-bounded by the optimal solution of the loss function. In contrast, our method defines matching as a classification problem, which is solved better in occluded regions.

Unsupervised Optical Flow Optical flow estimation and stereo matching are two closely related problems. Recently, unsupervised deep learning [1] and [31] for optical flow estimation were proposed. With slight modification, these frameworks can be applied to stereo matching. Both methods use traditional optical flow energy functions as the loss function in their neural networks. The key difference between our method and these ones is that we can update our network in an iterative way by gradually adding useful training data. Our framework is therefore simple and powerful.

3. Unsupervised Stereo Matching Learning

We in this section detail our unsupervised stereo matching scheme and the corresponding convolutional neural networks.

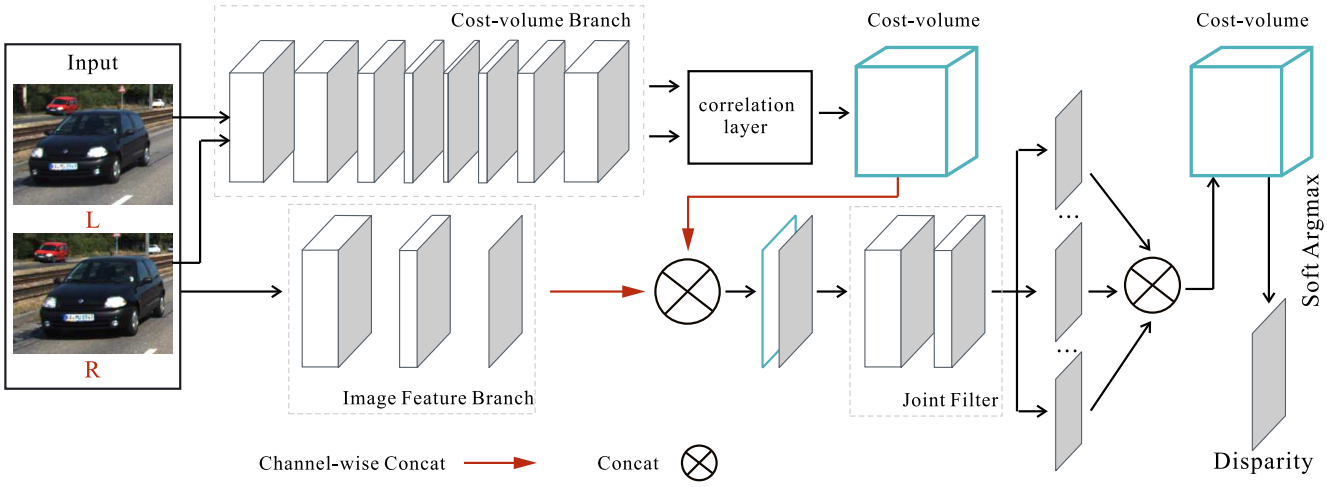


Figure 2: Our stereo matching learning network takes stereo images as input, and generates a disparity map. The architecture is with two branches where the first is for computing the cost-volume and the other is for jointly filtering the volume.

Algorithm 1 Iterative Unsupervised Learning

1: **for** $t = 0 \dots T$ **do**
2: Obtain matches $M^{L(t)}$ and $M^{R(t)}$ using parameter $\mathcal{E}^{t-1}$
3: Select confidential matches M^t from $M^{L(t)}$ and $M^{R(t)}$
4: Sample training data P^t from $\mathcal{I}$ using M^t
5: Train new parameter $\mathcal{E}^t$ using P^t
6: **end for**
7: **return** $\mathcal{E}^T$

3.1. Stereo Matching Learning Network

As shown in Figure 2, our stereo matching network is an end-to-end trainable framework taking stereo image pairs as input, and produce the disparity maps. Our network mainly includes cost-volume computation, cost-volume aggregation and disparity prediction. The detailed configuration is shown in Table 1. We explain them in what follows.

Cost-volume Computation As shown in Figure 2, we compute the cost-volume in the corresponding branch, which takes the left and right images as input, and generate the cost-volume. This branch employs the siamese architecture containing eight convolutional layers. Each layer is followed by batch normalization and ReLU. These layers produce feature maps for all patches of the two images. The maps are then fed into a correlation layer to compute the cost-volume. The correlation layer is the same as that of [21]. Detailed configuration is included in Table 1.

Cost-volume Aggregation Previous methods used edge-preserving filter to aggregate the cost-volume. We instead use an image-feature network to learn this process considering image structure. As shown in Figure 3, it extracts features from the two input images, similar to the scheme of Li *et al.* [15]. There are three convolutional layers followed by ReLU. They extract features (F_c) with size $1 \times h \times w$ from the input $h \times w$-pixel image.

After getting image features, we apply the joint filtering branch to incorporate the cost-volume with the color information from input. The feature (F_c) is concatenated with each channel of the cost-volume, and is further processed by three convolutional layers to produce the final cost-volume. This mimics the process of volume aggregation in traditional stereo matching methods. The learning scheme performs better because it automatically finds suitable parameters. We will discuss the performance details later.

Disparity Prediction With the filtered cost-volume, we produce the disparity map using the winner-take-all strategy. However, the Argmax operator is not derivable in back propagation. To address the issue, we use a soft Argmax operator [2], which returns the index of the maximum value in the cost-volume for each pixel.

With the above three components, our networks can learn stereo matching directly in an end-to-end manner. It is further enhanced by a unsupervised learning scheme in the following.

3.2. Unsupervised Learning Scheme

Let $\mathcal{I} = \{I_i^L, I_i^R\}_{i=1}^N$ be the dataset with N stereo image pairs. We obtain the initial match using randomly initialized network or DeepMatching [30] by ignoring vertical displacements. Both methods work. We will analyze them

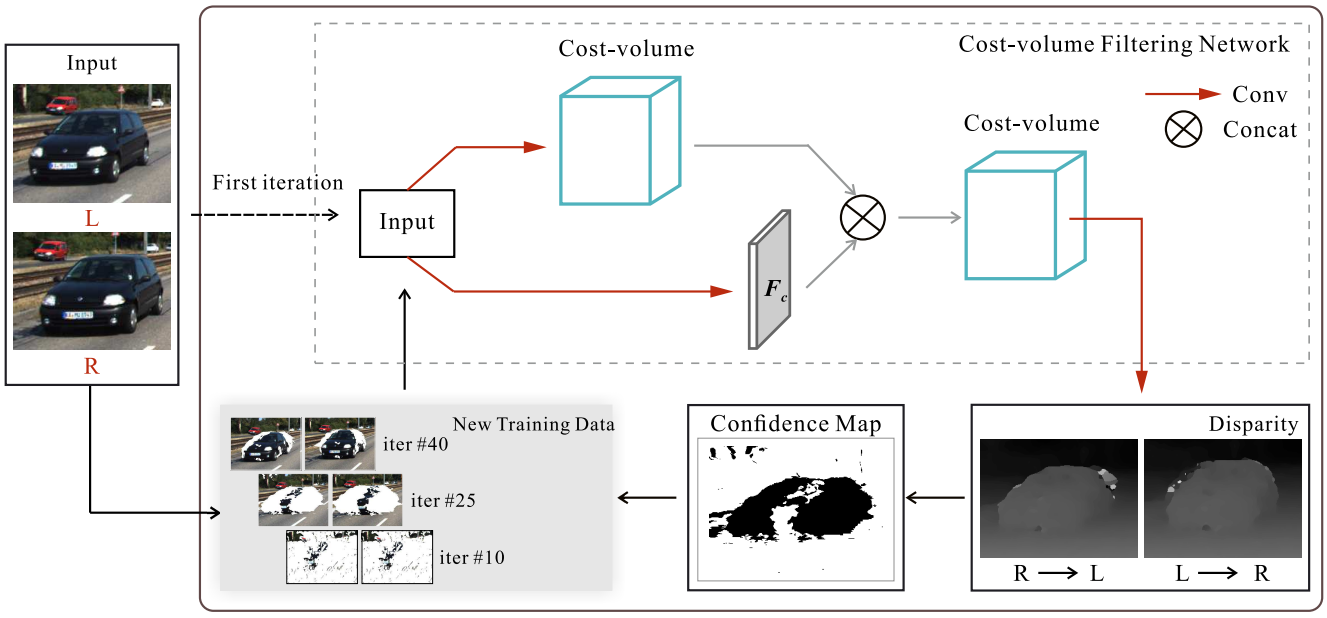


Figure 3: Our iterative unsupervised training framework consists of four parts: disparity prediction, confidence map estimation, training data selection and network training.

in the experiment section.

We train our network in an iterative way. At each iteration t, we compute matches from the left image to the right one, denoted as $M^{L(t)}$, and also from right to left, denoted as $M^{R(t)}$. We then develop a scheme to select confidential match M^t from $M^{L(t)}$ and $M^{R(t)}$. M^t are used to sample training data $P^t = \{I^L(\mathbf{p}), I^R(\mathbf{p})\}$ from $\mathcal{I}$, where $I^L(\mathbf{p})$ denotes the patch of I^L at position $\mathbf{p}$.

Afterwards, parameters $\mathcal{E}^t$ of our network are learned from P^t. The iterative process is illustrated in Figure 3 and Algorithm 1. The accuracy of the produced disparity maps and the number of confident matches increase in each iteration. The number of iterations T is around 50. We detail this process more in the following.

Confidential Match Selection The confidence map is computed using the information provided by the left and right disparity. Left-right consistency check is the simplest way to obtain this map. We warp the disparity of the left M^L using the disparity of right M^R by the forward warping function $W_{M^L}(M^R)$. By thresholding the difference between M^L and $W_{M^L}(M^R)$, we obtain the confidence map C. This process is expressed as

$$C_q = \begin{cases} 0, \|W_{M^L}(M^R)_q - M^L_q\| < t_c \\ 1, \|W_{M^L}(M^R)_q - M^L_q\| \geqslant t_c \end{cases}, \qquad (1)$$

where $W_M(I)$ means backward warping I using motion field M, and t_c is a threshold selected between 3 and 7 in our experiments.

This method is fast and effective. As shown in Figure 4, the confidence region grows with increasing iterations. We also statistically analyze the effectiveness. The accuracy of our final confidence map estimate on the KITTI 2015 dataset reaches 96%.

Training Data Selection Given a confident disparity $M^L(\mathbf{p})$ of the patch in position $\mathbf{p}$ in the left image I^L, we pick the corresponding patch in position $\mathbf{p} + M^L(\mathbf{p})$ in the right image I^R. To avoid flat patches, we manually remove them with low total variation values, computed as

$$tv(\mathbf{p}) = \sum_{q \in \Omega(\mathbf{p})} |(\partial I(\mathbf{p}))_q|, \qquad (2)$$

where $\Omega(\mathbf{p})$ is the rectangular region centered at $\mathbf{p}$. We pick patches with this value higher than the threshold t_d.

Post Processing Several modern stereo matching algorithms applied sophisticated post-processing, typically including cost aggregation, semi-global matching and slanted plane. As we already integrate the cost aggregation step in the network, we no longer need it. Our post processing contains only two steps to smooth the disparity maps produced by the network.

We first perform left-right consistency check. The inconsistent region, which is likely to be occlusion, is interpolated. Then we jointly filter the disparity using weighted median filter, where the radius of window is set to 7. Figure 5 shows the difference with and without post-processing.

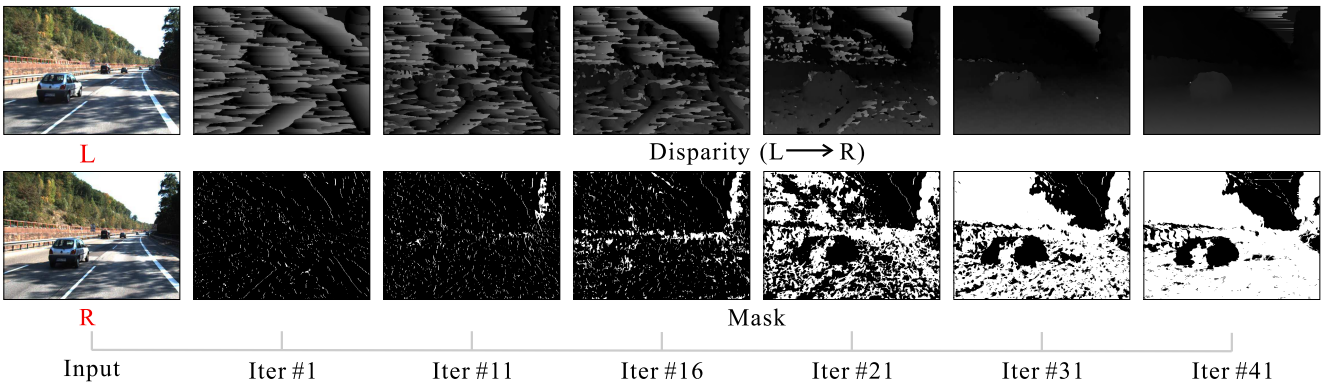


Figure 4: The area of the confidence map grows along with the iteration number, indicating that the network is well shaped after a few iterations. The white region in the confidence map means the confident region.

CVB1	CVB2	CVB3
conv layer (96, 7x7, 1x1), BatchNorm, ReLU	conv layer (96, 5x5, 1x1), BatchNorm, ReLU	conv layer (48, 5x5, 1x1), BatchNorm, ReLU
CVB4	CVB5	CVB6
conv layer (24, 3x3, 1x1), BatchNorm, ReLU	conv layer (12, 1x1, 1x1), BatchNorm, ReLU	conv layer (24, 3x3, 1x1), BatchNorm, ReLU
CVB7	CVB8	
conv layer (48, 5x5, 1x1), BatchNorm, ReLU	conv layer (96, 7x7, 1x1), BatchNorm	

IFB1	IFB2	IFB3
conv layer (96, 9x9, 1x1), ReLU	conv layer (48, 1x1, 1x1), ReLU	conv layer (1, 5x5, 1x1), ReLU
JF1	JF2	JF3
conv layer (64, 9x9, 1x1), ReLU	conv layer (32, 1x1, 1x1), ReLU	conv layer (1, 5x5, 1x1), ReLU

Table 1: Configuration of each component, cost-volume branch (CVB), image feature branch (IFB) and joint filtering branch (JF), of our network. Torch notations (channels, kernel, stride) are used to define the convolutional layers.

3.3. Analysis

Convergence Analysis We experimentally analyze convergence of our unsupervised learning scheme. Figure 4 shows update of the predicted disparity and confidence map in iterations. The disparity map gets smoother and is with more details in later iterations. The confident region also increases. For the KITTI 2015 dataset, the average percentage of confident regions in the confidence map reaches 75%. Most of them are rich in texture and edges, which means our selected confident patches are suitable for training. Generally 50 iterations are enough to let the maps stable.

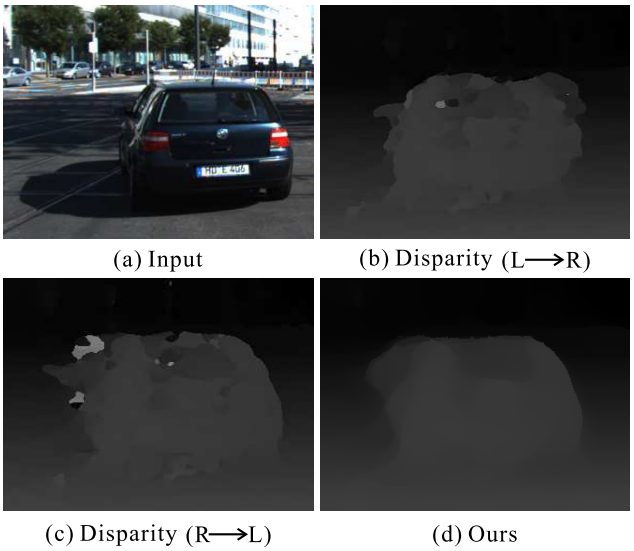


Figure 5: After processing, the network output results shown in (b) and (c), we obtain a better result (d).

Relations and differences to Previous methods Previous unsupervised methods [1] and [31] define their loss function following traditional optical-flow energy written as

$$E(u,v;I(x,y,t),I(x,y,t+\Delta t) = E_{photometric}(u,v;I(x,y,t),I(x,y,t+\Delta t)) + \lambda E_{smoothness}(u,v), \quad (3)$$

given two temporally consecutive images $I(x, y, t)$ and $I(x, y, t + \Delta t)$. Ahmadi *et al*. [1] took the energy as the loss function. Yu *et al*. [31] applied the similar idea and used a coarse-to-fine scheme during test.

Our method has several advantages compared with these methods. First, our method learns feature representation for patches and compare them in feature domain, instead of intensity domain. Thus our method does not require to satisfy the strong assumption of intensity consistency. In comparison, many previous methods do not work well when image pairs are taken under different lighting or camera setting.

Second, methods adopting traditional optical flow energy functions naturally inherit the drawback of traditional schemes, such as occlusion handling. The smooth term is needed to compromise computation on these regions. Our method does not find matches from the other image for the occluded patches. They are thus naturally removed in unsupervised learning.

Finally, if a method adopts the loss function similar to the traditional energy function, the performance of the neural networks is accordingly up-bounded considering traditional solvers to this function. Our method does not have this limitation, and therefore has the ability to produce comparable results with supervised learning methods.

We also make improvement on the architecture of the neural network. The key steps of stereo matching cost computation, cost aggregation and winner-take-all cascade form an end-to-end learnable framework in our system. The joint filtering branch is new with color information from the input images, which was shown to be useful in [10]. The joint filtering branch consists of convolutional layers and runs fast. Moreover, with the ability of learning, the filtering kernel is better than manually crafted ones. We show the comparison between our joint filtering network and other filters in the experiment section.

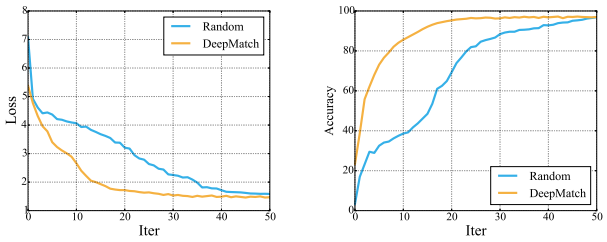
The above mentioned key differences boost the performance. Our scheme does not heavily correspond to traditional energy minimization schemes, and thus performs differently.

4. Experiments and Evaluations

4.1. Setting

We use the KITTI 2015 dataset [22], which includes 200 color image pairs to train and evaluate our method. In our experiments, we randomly pick 160 image pairs for training, and the other 40 images pairs for validation.

The training images are preprocessed by normalizing them to zero mean and standard deviation ones. In each unsupervised iteration, we first train the cost-volume branch by fixing other parts of the network. The branch is trained by minimizing cross-entropy loss similarity to that of [33] and [18]. We use the AdaGrad algorithm with learning rate $1e^{-2} * 0.8^{t-1}$, where t is the iteration number of our unsupervised method.



(a) Loss v.s. Iterations (b) Accuracy v.s. Iterations

Figure 6: The loss decreases while the validation accuracy increases in our unsupervised iterations.

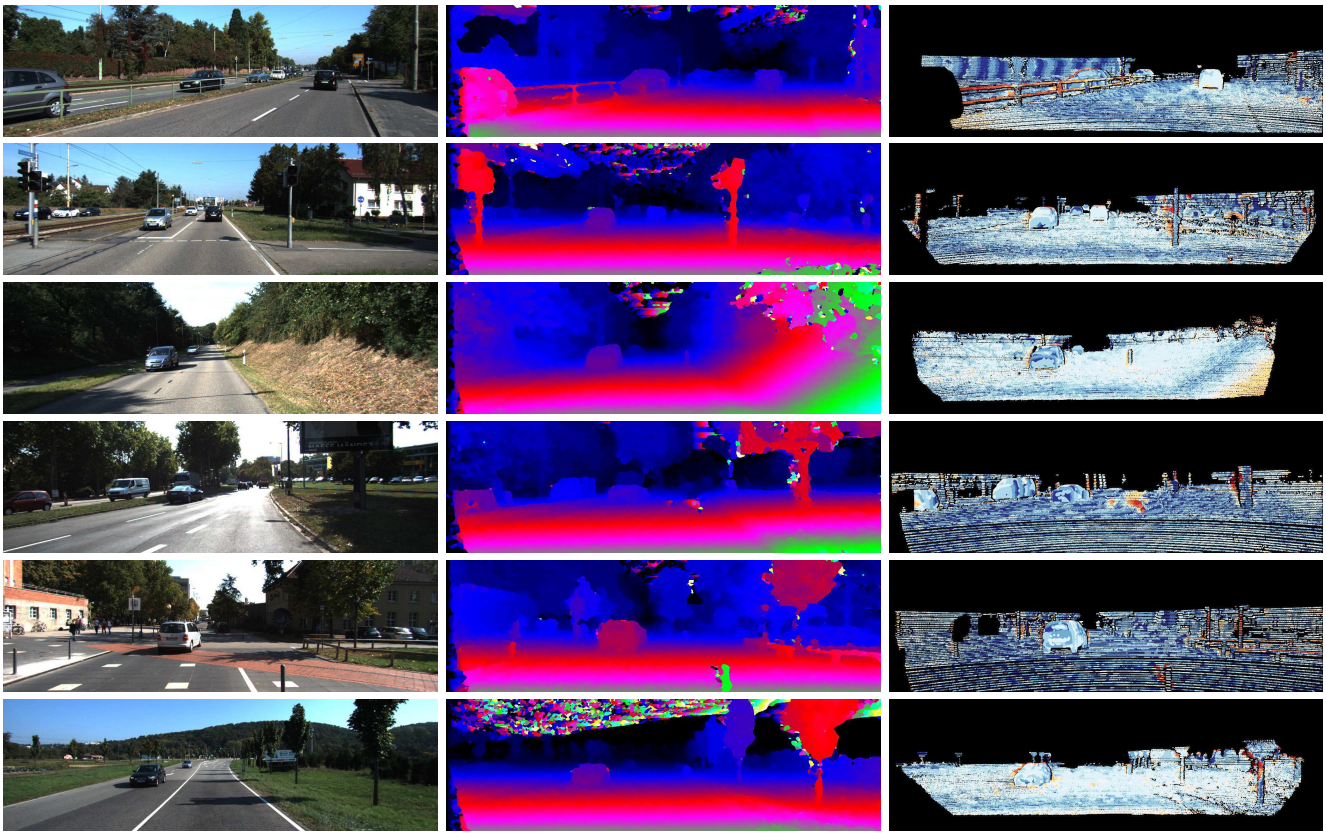
We train the cost-volume branch for 2,000 iterations. The batch size is fixed to 128. The image size is 1,242 $\times$ 375 with RGB channels. After the cross-entropy loss of the cost-volume branch gets steady, we train remaining parts in an end-to-end scheme. We set the learning rate as $1e^{-2}$. It takes around 50 unsupervised iterations with random initialization. For each iteration, we train the network using around 8,000 steps. The entire training process takes about 20 hours on a single NVIDIA TitanX card. The testing time is around 0.39 second for each image pair.

To evaluate results, we apply the bad pixel error, which calculates the percentage of disparity errors below a fixed threshold. We also evaluate the end-point-error (EPE) traditionally in optical flow estimation. Both the errors in non-occluded region (NOC) and all (ALL) pixels are reported. For existing methods to compare, we directly quote respectively reported results if available, and use our implementation otherwise.

4.2. Evaluations

Different Initialization As mentioned in earlier sections, either random initialization or other schemes can be used in our framework. For random initialization, we initialize the network weights using Xavier initialization. The network is then directly used for disparity prediction. As shown in Figure 4, although the network produces poor disparity in the first iteration, the shape is reasonable. These correctly predicted pixels are selected as training data to update the network parameters. With more iterations, the network gains stronger capability to predict correct disparity.

We also attempts to obtain initial matching using DeepMatching [30]. As shown in Figure 6, these two kinds of initialization result in similar performance. However, random initialization converges slower using 50 iterations, while DeepMatching initialization takes half of the iterations. Table lists the final accuracy.



(a) Input (b) Disparity (c) Error

Figure 7: Some of our results on KITTI 2015. All results are directly produced by our network without any post processing.

	>2 pixel	>3 pixel	>5 pixel
Random	9.63	7.81	6.01
DeepMatching [30]	9.50	7.69	5.67

Table 2: Comparisons of different initialization methods.

	>3 pixel	>5 pixel
Without Cost Aggregation	9.23	7.12
With Box Filter	7.33	5.42
With Guided Filter [9]	7.13	5.28
Ours	**6.81**	**5.01**

Table 3: We compare our cost aggregation with existing popular methods including box filter and guided filter. Our method outperforms others.

Effectiveness of Joint Filtering Branch We evaluate the effectiveness of the cost aggregation branch. First, we compare the performance with and without the cost aggregation branch. To show the advantage over traditional cost aggregation methods, we also compare our scheme with the widely used box filter and guided image filter [9]. For fair comparison, we train our network on the KITTI 2015 dataset in a supervised manner. The quantitative result is listed in Table 3 with the $>$3-pixel and $>$5-pixel error measure. As indicated by our experimental results, with the cost aggregation branch, both the $>$3-pixel and $>$5-pixel errors drop significantly. Our cost aggregation method outperforms box filter and guided image filter naturally.

Supervised Learning Although our framework is flexible where the network architecture can be replaced with other types for stereo matching, our proposed network outperforms existing ones. To show the effectiveness, we train our network in a supervised manner, and compare our result with others on the KITTI 2015 dataset. For fair comparison, we compare raw-network output without applying any smoothing or post processing for all methods. As shown in Table 4, our method achieves the best result regarding almost all metrics, while the computational time is comparable.

Approaches	> 2 pixels		> 3 pixels		> 4 pixels		> 5 pixels		End-Point		Runtime(s)
	NOC	All	NOC	All	NOC	All	NOC	All	NOC	All	
MC-CNN [33]	13.20	15.83	11.35	13.21	11.14	11.67	10.11	11.59	3.41 px	4.55 px	23
Deep Embed [3]	9.81	11.26	7.29	8.51	5.83	7.32	5.26	6.48	1.92 px	2.65 px	3.1
Content-CNN [18]	10.10	11.37	7.14	8.55	5.74	7.51	5.41	6.49	1.91 px	2.76 px	0.34
DispNet [21]	9.56	**10.74**	7.19	8.23	6.01	7.55	5.16	6.98	1.82 px	2.55 px	0.12
Ours	**9.23**	10.96	**6.81**	**7.29**	**5.55**	**7.28**	**5.01**	**6.22**	**1.57** px	**2.29** px	0.39

Table 4: Comparison of supervised methods on KITTI 2015. Our neural network architecture is effective to produce decent results.

Approaches	NOC		All	
	train	test	train	test
USCNN [1]	12.41	11.17	18.12	16.55
Yu *et al.* [31]	14.72	15.32	22.69	19.14
Ours	**8.35**	**8.61**	**9.41**	**9.91**

Table 5: Comparison of our method and existing unsupervised ones. Our method performs the best on KITTI 2015.

Unsupervised Learning Unsupervised optical flow and stereo matching methods are similar. So optical flow approaches can be applied to the stereo matching task with slight modification. Recent unsupervised optical flow methods using convolutional neural networks [1] and [31] define their loss function following the traditional optical flow energy (3), given two temporally consecutive images $I(x, y, t)$ and $I(x, y, t+\Delta t)$. These methods rely on a strong assumption that the two input frames are consistent in intensity. Further, it has been discussed extensively in the literature that a lower loss does not necessarily correspond to better quality of the results.

We compare our method with those of [1] and [31] on KITTI 2015 validation dateset and report the >3-pixel error in Table 5. Our method achieves the smallest error among all existing unsupervised methods for both training or testing. Comparing the results in Table 4, our unsupervised method is even comparable with existing supervised methods. We show some of our unsupervised learning results on KITTI 2015 in Figure 7. All the results are raw network output without any postprocessing for fair comparison.

5. Conclusion

Convolutional neural networks show its strong capability in various tasks, including stereo matching. However, due to the difficulty of ground truth data collection, supervised stereo matching methods can hardly produce satisfying results for real cases. Our main contribution is to propose a flexible unsupervised learning scheme suitable for various stereo matching network architectures. Our unsupervised method produces comparable results even with supervised methods. Moreover, we come up with a novel network structure that jointly filters the cost-volume. The network structure results in satisfying performance.

References

[1] A. Ahmadi and I. Patras. Unsupervised convolutional neural networks for motion estimation. In *ICIP*, 2016. 2, 5, 6, 8

[2] O. Chapelle and M. Wu. Gradient descent optimization of smoothed information retrieval metrics. *Inf. Retr.*, 13(3):216–235, 2010. 3

[3] Z. Chen, X. Sun, L. Wang, Y. Yu, and C. Huang. A deep visual correspondence embedding model for stereo matching costs. In *ICCV*, pages 972–980, 2015. 1, 2, 8

[4] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, pages 1422–1430, 2015. 2

[5] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, pages 2758–2766, 2015. 1, 2

[6] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, pages 3354–3361, 2012. 1

[7] R. Haeusler, R. Nair, and D. Kondermann. Ensemble learning for confidence measures in stereo vision. In *CVPR*, pages 305–312, 2013. 2

[8] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *CVPR*, pages 3279–3286, 2015. 1, 2

[9] K. He, J. Sun, and X. Tang. Guided image filtering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(6):1397–1409, 2013. 7

[10] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(2), 2013. 1, 6

[11] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013. 2

[12] D. Kong and H. Tao. A method for learning matching errors for stereo computation. In *BMVC*, pages 1–10, 2004. 2

[13] D. Kong and H. Tao. Stereo matching via learning multiple experts behaviors. In *BMVC*, pages 97–106, 2006. 2

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012. 2

[15] Y. Li, J.-B. Huang, A. Narendra, and M.-H. Yang. Deep joint image filtering. In *ECCV*, 2016. 3

[16] Y. Li and D. P. Huttenlocher. Learning for stereo vision using the structured support vector machine. In *CVPR*, 2008. 2

[17] Y. Li, M. Paluri, J. M. Rehg, and P. Dollár. Unsupervised learning of edges. *CoRR*, abs/1511.04166, 2015. 2

[18] W. Luo, A. G. Schwing, and R. Urtasun. Efficient deep learning for stereo matching. In *CVPR*, pages 5695–5703, 2016. 1, 2, 6, 8

[19] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *CoRR*, abs/1511.05440, 2015. 2

[20] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. *CoRR*, abs/1512.02134, 2015. 1

[21] N. Mayer, E. Ilg, P. Häusser, 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 *CVPR*, 2016. 3, 8

[22] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *CVPR*, pages 3061–3070, 2015. 6

[23] C. J. Pal, J. J. Weinman, L. C. Tran, and D. Scharstein. On learning conditional random fields for stereo - exploring model structures and approximate inference. *International Journal on Computer Vision*, 99(3):319–337, 2012. 1

[24] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. *CoRR*, abs/1604.07379, 2016. 2

[25] V. Patraucean, A. Handa, and R. Cipolla. Spatio-temporal video autoencoder with differentiable memory. *CoRR*, abs/1511.06309, 2015. 2

[26] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *CVPR*, 2007. 2

[27] A. Spyropoulos, N. Komodakis, and P. Mordohai. Learning to detect ground control points for improving the accuracy of stereo matching. In *CVPR*, pages 1621–1628, 2014. 2

[28] C. Strecha, W. von Hansen, L. J. V. Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *CVPR*, 2008. 1

[29] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, pages 2794–2802, 2015. 2

[30] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *ICCV*, pages 1385–1392, 2013. 3, 6, 7

[31] J. J. Yu, A. W. Harley, and K. G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. *CoRR*, abs/1608.05842, 2016. 2, 5, 6, 8

[32] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, pages 4353–4361, 2015. 1, 2

[33] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *CVPR*, pages 1592–1599, 2015. 1, 2, 6, 8

[34] L. Zhang and S. M. Seitz. Estimating optimal parameters for MRF stereo from a single image pair. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(2):331–342, 2007. 2