IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 31, 2018, accepted May 7, 2018, date of publication May 18, 2018, date of current version June 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2838442

Multiscale Feature Extractors for Stereo Matching Cost Computation

KYUNG-RAE KIM, YEONG JUN KOH, (Student Member, IEEE), AND CHANG-SU KIM, (Senior Member, IEEE)

School of Electrical Engineering, Korea University, Seoul 136-701, South Korea

Corresponding author: Chang-Su Kim (changsukim@korea.ac.kr)

This work was supported in part by the Cross-Ministry Giga KOREA Project Grant funded by the Korean Government (MSIT) (development of 4D reconstruction and dynamic deformable action model based hyper-realistic service technology) under Grant GK18P0200 and in part by the National Research Foundation of Korea Grant funded by the Korean Government (MSIP) under Grant NRF-2015R1A2A1A10055037 and Grant NRF-2018R1A2B3003896.

ABSTRACT We propose four efficient feature extractors based on convolutional neural networks for stereo matching cost computation. Two of them generate multiscale features with diverse receptive field sizes. These multiscale features are used to compute the corresponding multiscale matching costs. We then determine an optimal cost by combining the multiscale costs using edge information. On the other hand, the other two feature extractors produce uni-scale features by combining multiscale features directly through fully connected layers. Finally, after obtaining matching costs using one of the four extractors, we determine optimal disparities based on the cross-based cost aggregation and the semiglobal matching. Extensive experiments on the Middlebury stereo data sets demonstrate the effectiveness and efficiency of the proposed algorithm. Specifically, the proposed algorithm provides competitive matching performance with the state of the arts, while demanding lower computational complexity.

INDEX TERMS Stereo matching, matching cost computation, multiscale feature extraction, convolutional neural networks.

I. INTRODUCTION

Dense stereo matching is one of the most extensively studied topics in low-level computer vision. However, stereo matching is still an important topic since it poses challenging issues, which are not solved perfectly yet, such as occlusion, aperture problem, large displacement, disparity discontinuity, and difficulties on object boundaries, repetitive texture regions, and non-textured regions. In addition, stereo matching is essential in a wide variety of applications, including 3D reconstruction, robotics, and autonomous vehicles.

Scharstein and Szeliski [1] revealed that a stereo matching algorithm performs in four steps in general: matching cost computation, cost aggregation, disparity optimization, and disparity refinement. The matching cost computation, on which we focus in this work, is traditionally done by patch-based methods using the sum of absolute differences or normalized cross-correlation. Recently, it has been shown in [2]–[7] that the matching cost computation using convolutional neural networks (CNNs) yields more reliable costs and improves overall stereo matching performances on the Middlebury stereo datasets [8] and the KITTI datasets [9], [10]. These CNN-based methods can be classified into two types. The first type uses a CNN to extract a feature vector from each patch. Then, it compares the CNN features of two patches via the dot product to yield a matching cost. On the other hand, the second type replaces the dot product operation with fully-connected layers. Whereas the dot product is computationally simpler than the fully-connected layers, the second type yields more accurate stereo matching results in general. The first type is adopted in [2]–[4], while the second type in [2] and [5]–[7]. In this work, we consider the first type and develop CNN-based feature extractors.

In this paper, we propose four networks, which are used independently for matching cost computation. We first propose two CNN-based multiscale feature extractors, called network A and network B. Network A is composed of 25 convolutional layers, while network B consists of 19 convolutional layers. Both networks generate multiscale features, whose receptive field sizes vary from 7×7 to 39×39. Using the multiscale features, we compute the corresponding multiscale matching costs. These multiscale matching costs have a trade-off between reliability and

accuracy. Therefore, we determine an optimal matching cost by combining the multiscale costs based on edge information. Second, we propose two CNN-based uni-scale feature extractors, called network C and network D. Networks C and D merge multiscale features into uni-scale ones using fully-connected layers. In other words, networks C and D are end-to-end networks to extract uni-scale features. After obtaining matching costs by employing one of the four networks, we aggregate the costs using the cross-based cost aggregation. We then optimize disparities using the NTDE method [11], which was our preliminary work. In this work, we lower the complexity of [11] by obtaining a pre-estimated disparity map from the cost aggregation result. Experimental results on the Middlebury stereo datasets demonstrate that the proposed four networks extract reliable features for the matching cost computation, and that the proposed algorithm provides competitive performances with the state-of-the-art algorithms, while requiring lower computational complexity.

The rest of this paper is organized as follows: Section II reviews related work. Sections III and IV propose the four networks for the matching cost computation, and Section V develops the stereo matching algorithm. Section VI discusses experimental results. Section VII concludes this work.

II. RELATED WORK

A. STEREO MATCHING

In general, stereo matching is performed in four steps [1]: matching cost computation, cost aggregation, disparity optimization, and disparity refinement.

First, patch-based matching costs are computed, *e.g.* the sum of absolute differences (SAD) [12], the sum of squared differences (SSD) [13]–[15], the normalized cross-correlation (NCC) [16], or the sum of gradient magnitude differences (GRAD) [17]. These costs, however, may be ineffective in the cases of occlusion, lack of textural information, incorrect calibration, and noisy input. For more reliable matching, Hirschmüller [18] exploited the mutual information between image patches, which is robust against illumination variation. Also, several stereo matching algorithms [19]–[22] adopt the census transform [23] for the matching cost computation. The census transform represents an image patch with a binary descriptor, by comparing the intensities of the center pixel and the other pixels within the patch. The census transform is also combined with SAD [20] or GRAD [21], [22].

Second, the cost aggregation step sums the matching costs of neighboring pixels within a support window. Yoon and Kweon [24] computed an adaptive weight for each pixel in a support window based on the color similarity and the geometric distance. By extending this bilateral filter [24], Chen *et al.* [25] developed a trilateral filter for stereo matching. Also, Hosni *et al.* [26] adopted the guided filter [27] to preserve edge properties during the cost aggregation. Zhang *et al.* [28] constructed an adaptive support region for each pixel to constrain that pixels within the support region should have similar disparities. They adopted cross-based support regions.

Third, a disparity map is obtained by minimizing a global energy function, which consists of data and smoothness terms in general. Optimization techniques, such as dynamic programming [29], graph cuts [30] and belief propagation [31], are adopted to minimize energy functions. Hirschmüller [18] adopted the semiglobal matching (SGM) to minimize a 2D energy function efficiently by performing 1D minimization in several directions. Due to its low complexity and relatively high performance, SGM has been widely used in various stereo matching algorithms [2]–[6], [20], [32].

Also, slanted-plane models have been introduced to obtain robust disparity maps [17], [21], [22], [33]–[36]. These methods find optimal three-dimensional plane parameters for local disparities. Bleyer *et al.* [33] initialized planes randomly and updated them iteratively to lower matching costs. Assuming that a homogeneous region can be represented by a single plane, segmentation techniques have been adopted in the slanted-plane models in [17], [21], [22], [34]–[36]. Hong and Chen [34] computed segment-wise plane parameters based on weighted least-squares regression. To alleviate the sensitivity to outliers in the least-squares regression, Klaus *et al.* [17] solved the plane fitting problem using a decomposition method, which estimates the horizontal slant and the vertical slant separately. Yamaguchi *et al.* [21] formulated an energy function using a hybrid Markov random field (MRF), which uses continuous random variables for slanted 3D planes and discrete random variables for occlusion boundaries. Yamaguchi *et al.* [35] also used superpixels to determine planes. Zhang *et al.* [22] performed 2D triangulation using superpixels and found a plane for each triangle. Li *et al.* [36] constructed several superpixel structures and assigned plane labels to superpixels using an iterative α-expansion graph cut.

Fourth, most stereo algorithms perform disparity refinement as post-processing. The left-right consistency check [20], [36], which compares disparities of pixels in a left image with those of warped pixels in a right image, is commonly used. Pixels, which fail this consistency check, are considered to be occluded. Disparities of occluded pixels are then filled in by superposing neighboring values. Also, a median filter and a bilateral filter are adopted to refine disparity maps and remove outliers [2], [11].

B. MATCHING COST COMPUTATION USING CNNs

Recently, CNNs have been adopted with remarkable successes in various vision tasks, including image classification [37]–[39], object detection [40], object tracking [41], [42], image segmentation [43], [44], and edge detection [45]. Furthermore, CNNs have been also used to learn patch distance metrics for stereo matching [2]–[7].

Žbontar and LeCun [2] proposed two Siamese networks–the fast one and the accurate one–to compute matching costs between image patches. The accurate network

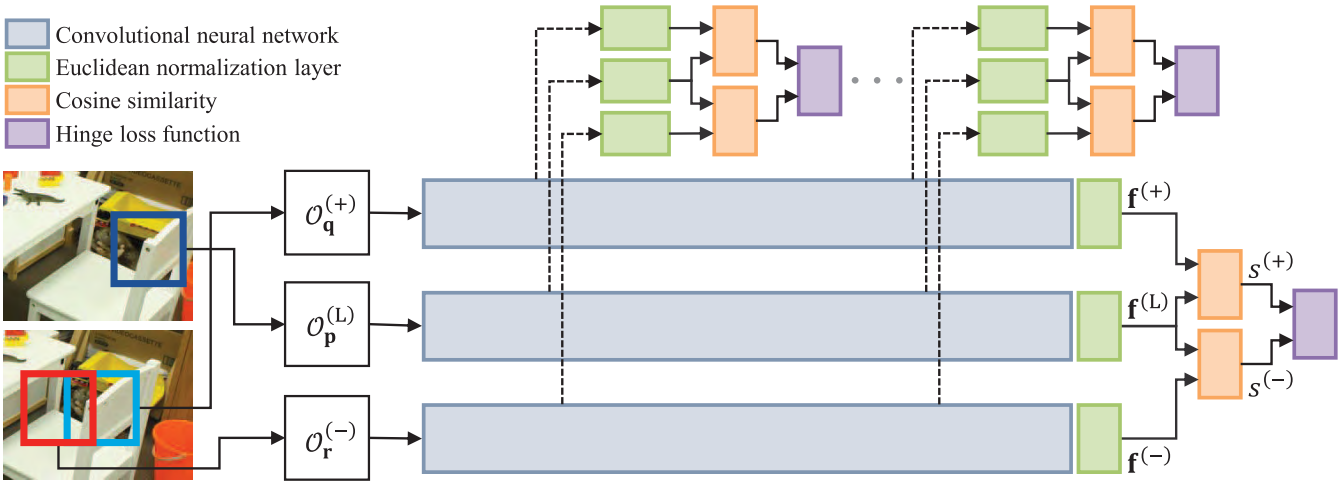


FIGURE 1. Training of the triplet network, which is composed of three identical CNNs with shared parameters. We employ the resultant CNN as the feature extractor. To achieve multiscale feature extraction, we consider losses at hidden layers as well as the last layer. We compute the losses based on the cosine similarities between pairs of matching features.

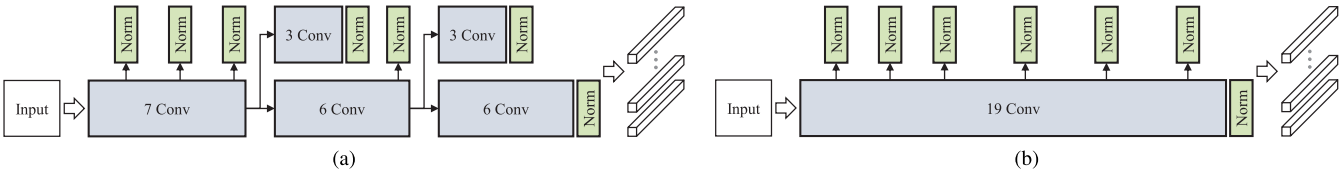


FIGURE 2. Two different network architectures to generate multiscale features. Except for the last layers, each convolutional layer contains a ReLU block. (a) Network A consists of 25 convolutional layers. (b) Network B has 19 convolutional layers.

outperforms the fast one, but its computational complexity is much higher due to several fully-connected layers at its end. Chen *et al.* [3] developed a multiscale deep embedding model to exploit two different scales of input pairs. Luo *et al.* [4] also trained a Siamese network based on the cross-entropy over all possible disparities. Seki and Pollefeys [5] adopted a confidence prediction network to improve the prediction accuracy in difficult regions due to, for example, noise and reflection. Park and Lee [6] incorporated a per-pixel pyramid pooling layer into the accurate network in [2] to handle a large image patch. Ye *et al.* [7] designed a multi-size pooling layer and included it in the accurate and fast networks in [2].

III. MULTISCALE FEATURE EXTRACTION NETWORKS

We develop two CNN-based feature extractors for the matching cost computation in stereo matching. Both extractors generate multiscale features, but they have different structures. To train each extractor, we use a triplet network that consists of three identical CNNs with shared parameters. The triplet network takes three input patches and is trained to compare the similarity between a reference patch and a positive patch with that between the reference patch and a negative patch. By connecting loss functions to hidden layers, the resultant CNN is capable of producing multiscale features with different receptive field sizes.

A. NETWORK ARCHITECTURE

Fig. 1 illustrates the triplet network, which we use to design the feature extractor. The triplet network includes three identical CNNs with shared parameters. After its training, we adopt the resultant CNN as the feature extractor. Given three input image patches, the triplet network generates their multiscale features, whose similarities are compared to evaluate loss functions at hidden layers and the last layer. By minimizing the loss functions, we optimize the shared parameters and construct the multiscale feature extractor.

Fig. 2 shows two feature extraction networks, in which each convolutional layer contains a rectified linear unit (ReLU) block. The set of all such convolutional layers in Fig. 2(a) and (b), respectively, corresponds to a 'convolutional neural network' block in Fig. 1.

1) NETWORK A

Fig. 2(a) shows network A with 25 convolutional layers, the kernel sizes of which are $3 \times 3 \times 1$ at the first layer and $3 \times 3 \times 112$ at the others without padding. The number of kernels is fixed to 112 at all convolutional layers. Network A consists of a main network and two sub-networks. In the main network, 19 convolutional layers are connected serially. Each sub-network, which has 3 convolutional layers, forks from 7th or 13th convolutional layer of the main network. The ReLU activation function follows each convolutional layer, except the three last layers at the ends of the main network and the sub-networks. We extract multiscale features from the outputs of 3rd, 5th, 7th, 13th, and the three last layers. Table 1 summarizes the CNN structure. Since the output sizes are bigger than 1×1 except for the output of the

TABLE 1. The structures of the two networks for multiscale feature extraction. Networks A and B use input patches of size 39 × 39 × 1.

Layer	Networks A and B	
	Output size	Receptive field
Input	$39 \times 39 \times 1$	-
3rd Conv	$33 \times 33 \times 112$	7×7
5th Conv	$29 \times 29 \times 112$	11×11
7th Conv	$25 \times 25 \times 112$	15×15
10th Conv	$19 \times 19 \times 112$	21×21
13th Conv	$13 \times 13 \times 112$	27×27
16th Conv	$7 \times 7 \times 112$	33×33
19th Conv	$1 \times 1 \times 112$	39×39

19th convolutional layer, we extract features from the centers of the outputs.

The receptive field of a neuron means the local region in an input patch affecting the output of the neuron. Note that, as a feature for the input patch, we use the output of the center neuron at a convolutional layer. Thus, the receptive field indicates the effective patch whose information is used for the matching cost computation. For example, if the center neuron at 5th convolutional layer is used as the feature, the effective patch size is 11×11 as specified in Table 1. In [4] and [6], it was demonstrated that a larger receptive field (or effective patch) provides better stereo matching performance in general. However, as a receptive field gets larger, the network simplifies the input information more. This is desirable for extracting image semantics, but results in a loss of details in the input. Therefore, the network with a large receptive field may degrade the matching performance for an image region with complex texture. To address this issue adaptively, we design network A to generate multiscale features with different receptive field sizes. Specifically, we extract 7 multiscale features from 3rd to last convolutional layers, as listed in Table 1. Then, we compute matching costs using the multiscale features and combine those costs based on edge information, as will be described in Section III-C and III-D.

2) NETWORK B

As shown in Fig. 2(b), network B extracts multiscale features similarly to network A, but has less convolutional layers without branching into sub-networks. In other words, network B has the same structure as the main network in network A, which has 19 convolutional layers. We extract multiscale features at 3rd, 5th, 7th, 10th, 13th, 16th, and the last convolutional layers. Note that the features from 10th and 16th layers substitute for those from the two last layers of the sub-networks of network A.

B. TRAINING PHASE

Let $I^{(L)}$ and $I^{(R)}$ denote a pair of left and right images for stereo matching. First, we select a 39×39 reference patch $\mathcal{O}_{\mathbf{p}}^{(L)}$ in the left image $I^{(L)}$, whose center pixel is $\mathbf{p}$. Next, we extract the positive patch $\mathcal{O}_{\mathbf{q}}^{(+)}$ in the right image $I^{(R)}$, which matches $\mathcal{O}_{\mathbf{p}}^{(L)}$ in $I^{(L)}$. Note that the difference vector $\mathbf{q} - \mathbf{p}$ is the ground-truth disparity. Moreover, we choose a negative patch $\mathcal{O}_{\mathbf{r}}^{(-)}$ randomly in $I^{(R)}$, which does not match $\mathcal{O}_{\mathbf{p}}^{(L)}$.

To train the triplet network, we feed these three patches $\mathcal{O}_{\mathbf{p}}^{(L)}$, $\mathcal{O}_{\mathbf{q}}^{(+)}$, and $\mathcal{O}_{\mathbf{r}}^{(-)}$ into the three CNNs, as illustrated in Fig. 1. Note in Table 1 that the output of the last convolutional layer (19th Conv in network A or B) has the size $1 \times 1 \times 112$, which can be regarded as a 112-dimensional feature vector. Let $\mathbf{f}^{(L)}$, $\mathbf{f}^{(+)}$, and $\mathbf{f}^{(-)}$ denote such feature vectors for $\mathcal{O}_{\mathbf{p}}^{(L)}$, $\mathcal{O}_{\mathbf{q}}^{(+)}$, and $\mathcal{O}_{\mathbf{r}}^{(-)}$, respectively. The Euclidean norms of these feature vectors are scaled to 1 by the normalization layers, respectively. We train the triplet network so that the reference feature vector is similar to the positive one but dissimilar from the negative one. To this end, we compute two cosine similarities: $s^{(+)}$ between $\mathbf{f}^{(L)}$ and $\mathbf{f}^{(+)}$, and $s^{(-)}$ between $\mathbf{f}^{(L)}$ and $\mathbf{f}^{(-)}$. In other words,

$$s^{(+)} = (\mathbf{f}^{(L)})^T \mathbf{f}^{(+)}, \tag{1}$$

$$s^{(-)} = (\mathbf{f}^{(L)})^T \mathbf{f}^{(-)}. \tag{2}$$

Then, we adopt the hinge loss function [46] to penalize the case $s^{(-)} > s^{(+)}$, which is given by

$$L(s^{(+)}, s^{(-)}) = \max\left\{0, m + s^{(-)} - s^{(+)}\right\} \tag{3}$$

where m is a margin.

The hinge loss function in (3) is also connected to the intermediate convolutional layers, as well as the last convolutional layer, using green blocks in Fig. 2(a) or (b). By training the triplet network to minimize the sum of the loss functions, we can obtain multiscale features with different receptive fields via a single training process. However, except for the last convolutional layer, the outputs of the convolutional layers are not vectors, but volumes, as specified in Table 1. To extract the feature vectors from the intermediate convolutional layers, we select the center vectors from the volumes. We then compute the cosine similarities and the hinge losses in the same way as the last convolutional layer. Finally, we train the triplet network using the stochastic gradient descent to minimize the sum of the multiscale losses.

C. MULTISCALE FEATURE DESCRIPTION

After training the triplet network, we use its parameters to construct the corresponding CNN for a full-size image. To generate an output feature map of the same size as an input image, we perform zero padding at the convolutional layers. Given an $M \times N \times 1$ image, the CNN feature extractor yields a feature map of size $M \times N \times 112$ at each convolutional layer. By using network A or B, we extract 7 feature maps from the image. Let F_n denote the output feature map, where n denotes the receptive field size. For the matching cost computation, we obtain two sets of multiscale feature maps, $\mathcal{F}^{(L)} = \{F_7^{(L)}, F_{11}^{(L)}, \ldots, F_{39}^{(L)}\}$ and $\mathcal{F}^{(R)} = \{F_7^{(R)}, F_{11}^{(R)}, \ldots, F_{39}^{(R)}\}$, for the left and right images, respectively.

D. MATCHING COST COMPUTATION

We compute matching costs, which are the base information for obtaining a disparity map in stereo matching. Let $\mathcal{D}_{\mathbf{p}}$ be the set of candidate disparities for pixel $\mathbf{p}$. The matching cost at $\mathbf{p}$ for a candidate disparity $d \in \mathcal{D}_{\mathbf{p}}$, which corresponds to the receptive field size n, is defined as

$$C_n(\mathbf{p}, \mathbf{d}) = \|F_n^{(L)}(\mathbf{p}) - F_n^{(R)}(\mathbf{p} - \mathbf{d})\|_2 \tag{4}$$

where $\mathbf{d} = (d, 0)$.

We combine multiscale matching costs from network A or B by employing edge information. To this end, we extract an edge map of the left image $I^{(L)}$ using the HED edge detector in [45]. In the edge map, each pixel represents the edge strength within [0, 1]. We consider the sum $E_n(\mathbf{p})$ of edge strengths in a square window, whose center is pixel $\mathbf{p}$ and side has size $n \in \mathcal{N} = \{7, 11, ..., 39\}$. Note that the window size is set to be one of the receptive field sizes. We compute the difference between the sum of edge values at size n and its previous size in the ascending order. If the difference is larger than a threshold $\rho_1 = 1$, we select the matching cost corresponding to the size n. For example, when the difference between $E_{11}(\mathbf{p})$ and $E_7(\mathbf{p})$ is larger than ρ_1, the matching cost $C(\mathbf{p}, \mathbf{d})$ at $\mathbf{p}$ is set to be $C_{11}(\mathbf{p}, \mathbf{d})$. When the difference is smaller than ρ_1, we compare the next difference between $E_{15}(\mathbf{p})$ and $E_{11}(\mathbf{p})$, and so on. Notice that, in the case of the smallest size 7, we check whether the sum $E_7(\mathbf{p})$ is larger than a threshold $\rho_2 = 35$ or not. Also, in the case that all tests for the 7 receptive field sizes fail, we average all matching costs, *i.e.*

$$C(\mathbf{p}, \mathbf{d}) = \frac{1}{N} \sum_{n \in \mathcal{N}} C_n(\mathbf{p}, \mathbf{d}) \tag{5}$$

where $N = 7$ is the number of receptive field sizes. Fig. 3 illustrates the combination process to select the matching costs of appropriate receptive field sizes. We see that, for pixels with high edge strengths, a small receptive field is selected in general. On the contrary, large receptive fields are chosen for pixels far from edges.

We represent the result of this selection process using a 3D weight map $W(\mathbf{p}, n)$ of size $H \times W \times N$, where H and W are the height and width of the left image. For each pixel $\mathbf{p}$, the weight for the selected receptive field size n is set to 1, while those for the other sizes are set to 0. When no single size is selected, weights are uniformly set to $\frac{1}{N}$ for all sizes. Then, in order to smooth the weights, we apply a Gaussian filter $G_\sigma(\mathbf{p})$. More specifically, the smoothed weight map $W_G(\mathbf{p}, n)$ is given by

$$W_G(\mathbf{p}, n) = \sum_{\mathbf{p}_\omega} G_\sigma(\mathbf{p}_\omega) W(\mathbf{p}_\omega, n) \tag{6}$$

where $\sigma = 20$ is the standard deviation for the Gaussian filter and $\mathbf{p}_\omega$ denotes a neighbor pixel of $\mathbf{p}$. We then normalize the weights by

$$\tilde{W}_G(\mathbf{p}, n) = \frac{W_G(\mathbf{p}, n)}{\sum_{n \in \mathcal{N}} W_G(\mathbf{p}, n)}. \tag{7}$$

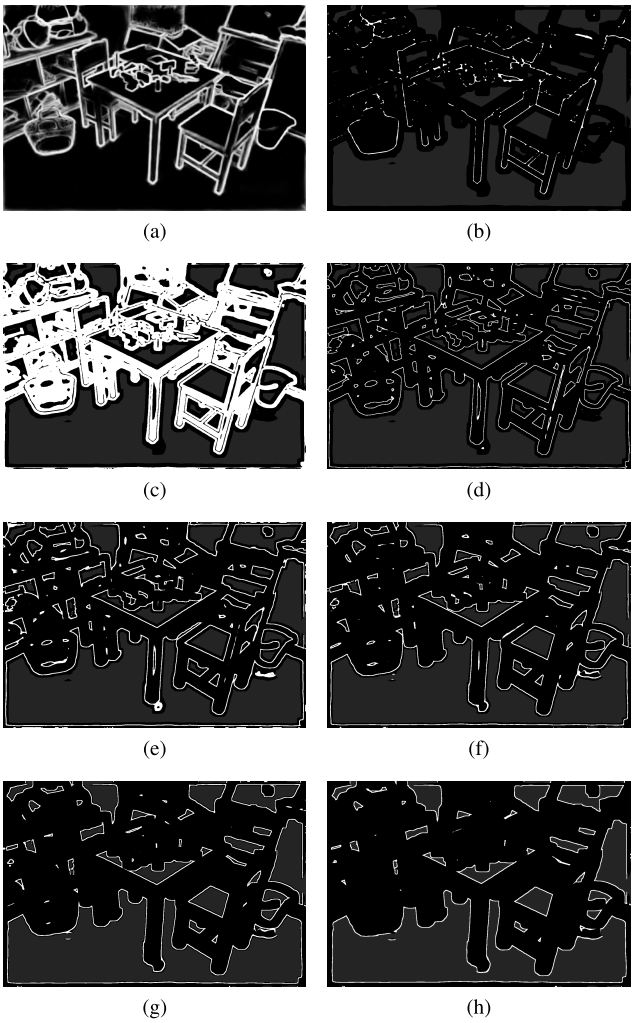


FIGURE 3. (a) Edge map for an input image. (b)∼(h) Selection results at each receptive field size from $n = 7$ to $n = 39$. White pixels represent the selected pixels in the corresponding receptive field. For dark gray pixels, no single receptive field is selected, and the averaging in (5) is performed instead.

Finally, the combined matching cost at $\mathbf{p}$ is defined as a weighted superposition of the multiscale matching costs, given by

$$C(\mathbf{p}, \mathbf{d}) = \sum_{n \in \mathcal{N}} \tilde{W}_G(\mathbf{p}, n) C_n(\mathbf{p}, \mathbf{d}). \tag{8}$$

IV. MULTISCALE FEATURE EXTRACTION AND COMBINATION NETWORKS

In the previous section, we propose two networks (*i.e.* network A and network B) for extracting multiscale features. For the matching cost computation, we calculate a matching cost in each scale and then combine the multiscale costs based on the edge information. On the other hand, in this section, we propose two end-to-end networks, called network C and network D, that not only generate multiscale features but also combine them into a single feature. More specifically, these networks extract multiscale outputs of

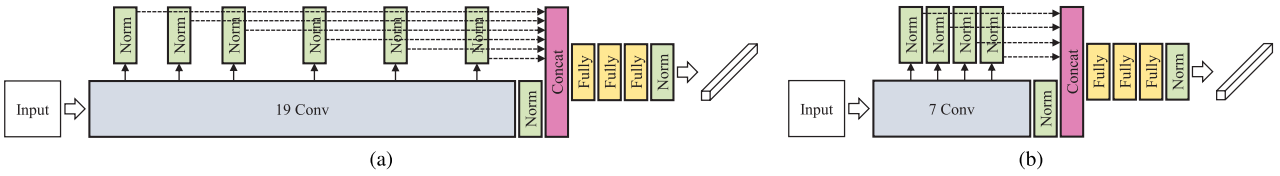


FIGURE 4. The network architectures for extracting and combining multiscale features. (a) Network C includes 3 fully-connected layers, which combine multiscale features and generates a single feature. (b) Network D consists of 7 convolutional layers and 3 fully-connected layers.

various receptive field sizes from hidden layers and then combine them using fully-connected layers.

A. NETWORK ARCHITECTURE

We also use the triplet network in Fig. 1 to design the feature extraction and combination networks. However, note that the loss function is connected to the last layer only to generate a single feature. Figs. 4(a) and (b) show the structures of network C and network D, respectively. Whereas networks A and B use the outputs of hidden layers as multiscale features, networks C and D combine the outputs of hidden layers through fully-connected layers.

1) NETWORK C

In Fig. 4(a), network C consists of two parts. The first part has the same structure as the whole network B. In addition, the second part has 3 fully-connected layers, which have 1024 nodes at the first two layers, respectively, and 112 nodes at the last layer. The ReLU activation follows each convolutional or fully-connected layer except for the last layers of both parts. The input of the second part is obtained by concatenating the output responses of the first part. Since the output responses have different sizes as listed in Table 2, we only extract their center vectors and concatenate them along the channel dimension. The second part uses the fully-connected layers to combine the concatenated features into a single feature, which becomes the output of network C.

TABLE 2. The structures of the two networks for multiscale feature extraction and combination. Networks C uses input patches of size 39 × 39 × 1, while network D uses those of size 15 × 15 × 1.

Layer	Network C		Network D	
	Output size	Receptive field	Output size	Receptive field
Input	39 × 39 × 1	-	15 × 15 × 1	-
3rd Conv	33 × 33 × 112	7 × 7	9 × 9 × 112	7 × 7
4th Conv	-	-	7 × 7 × 112	9 × 9
5th Conv	29 × 29 × 112	11 × 11	5 × 5 × 112	11 × 11
6th Conv	-	-	3 × 3 × 112	13 × 13
7th Conv	25 × 25 × 112	15 × 15	1 × 1 × 112	15 × 15
10th Conv	19 × 19 × 112	21 × 21	-	-
13th Conv	13 × 13 × 112	27 × 27	-	-
16th Conv	7 × 7 × 112	33 × 33	-	-
19th Conv	1 × 1 × 112	39 × 39	-	-

2) NETWORK D

In Fig. 4(b), network D has a similar structure to network C, but has less convolutional layers. More specifically, the first part is composed of 7 convolutional layers, whose kernel sizes are 3 × 3 × 1 at the first layer and 3 × 3 × 112 at the others. The input of the second part is obtained by concatenating the output responses of 3rd to 7th convolutional layers. The second part has 3 fully-connected layers as in network C. Table 2 summarizes the output sizes and the receptive field sizes of network D.

B. TRAINING PHASE

For network C, patches $\mathcal{O}_{\mathbf{p}}^{(L)}$, $\mathcal{O}_{\mathbf{q}}^{(+)}$, and $\mathcal{O}_{\mathbf{r}}^{(-)}$ have the size 39×39, as in networks A and B. In contrast, for the shallower network D, the patch size is 15×15, since the largest receptive field size of the features is 15×15. To train the triplet network for network C or D, the same process in Section III-B is adopted except that the single loss function is connected to the last layer only.

C. FEATURE DESCRIPTION AND MATCHING COST COMPUTATION

Given an input image, we generate a single feature map of the same size by employing the parameters of the triplet network. To this end, we perform zero padding at the convolutional layers. Let $F_s^{(L)}$ and $F_s^{(R)}$ denote the single feature maps for the left and right images, respectively. Then, we compute the matching cost for a candidate disparity $\mathbf{d} = (d, 0)$ at pixel $\mathbf{p}$ by

$$C(\mathbf{p}, \mathbf{d}) = \|F_s^{(L)}(\mathbf{p}) - F_s^{(R)}(\mathbf{p} - \mathbf{d})\|_2. \tag{9}$$

V. STEREO MATCHING

This section proposes a stereo matching algorithm, which is based on the matching cost computation in Sections III and IV. In addition, the proposed algorithm performs cost aggregation and disparity optimization. After aggregating matching costs using the cross-based cost aggregation [28], we optimize the disparity map using the semiglobal matching with adaptive smoothness constraints in non-textured regions and on depth edges [11].

A. CROSS-BASED COST AGGREGATION

Most stereo matching methods include the cost aggregation step. This step aggregates matching costs of pixels in a support region, in which the pixels tend to have similar intensities. In this work, we aggregate the matching costs, $C(\mathbf{p}, \mathbf{d})$ in (8) or (9), by employing the cross-based cost aggregation (CBCA) technique [28]. This technique has low computational complexity but provides relatively high performance.

First, a cross-based local support region is constructed for each pixel $\mathbf{p} = (p_x, p_y)$. The region is formed from an upright

cross shape, which consists of left, right, up, and bottom arms. Starting from the left adjacent pixel $\mathbf{p}_1 = (p_x - 1, p_y)$ of $\mathbf{p}$, pixel $\mathbf{p}_l = (p_x - l, p_y)$ is included in the left arm of $\mathbf{p}$, if the following two rules are satisfied:

$$|I(\mathbf{p}_l) - I(\mathbf{p})| < \tau \tag{10}$$

$$\|\mathbf{p}_l - \mathbf{p}\|_2 < \eta \tag{11}$$

where $I(\mathbf{p})$ denotes the pixel intensity at $\mathbf{p}$, $\tau = 0.04$, and $\eta = 11$. The inclusion terminates when either condition is violated. The resultant left arm is denoted by $\mathcal{L}^{\mathbf{p}} = \{\mathbf{p}_1, ..., \mathbf{p}_L\}$. Similarly, the right arm $\mathcal{R}^{\mathbf{p}}$, the up arm $\mathcal{U}^{\mathbf{p}}$, and the bottom arm $\mathcal{B}^{\mathbf{p}}$ are formed. Then, to extend the upright cross to the support region, the left and right arms at each pixel $\mathbf{q} \in \{\mathcal{U}^{\mathbf{p}}, \mathcal{B}^{\mathbf{p}}\}$ are further considered. In other words, the cross-based support region $\mathcal{S}^{\mathbf{p}}$ of $\mathbf{p}$ is given by

$$\mathcal{S}^{\mathbf{p}} = \{\mathcal{L}^{\mathbf{p}}, \mathcal{R}^{\mathbf{p}}, \mathcal{U}^{\mathbf{p}}, \mathcal{B}^{\mathbf{p}}, \mathcal{U}^{\mathbf{q}}, \mathcal{B}^{\mathbf{q}} : \mathbf{q} \in \{\mathcal{U}^{\mathbf{p}}, \mathcal{B}^{\mathbf{p}}\}\}. \tag{12}$$

Fig. 5 shows an example of the support region.

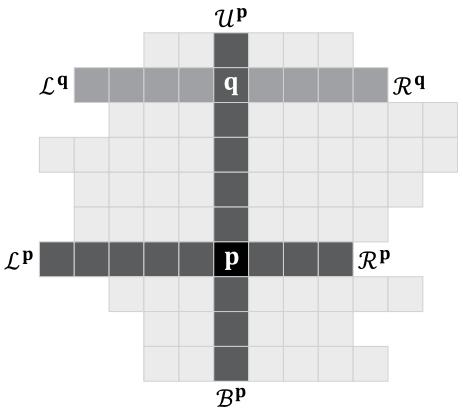


FIGURE 5. The support region of pixel $\mathbf{p}$ for the cross-based cost aggregation (CBCA) [28]. It has the left arm $\mathcal{L}^{\mathbf{p}}$, the right arm $\mathcal{R}^{\mathbf{p}}$, the up arm $\mathcal{U}^{\mathbf{p}}$, and the bottom arm $\mathcal{B}^{\mathbf{p}}$. Also, the left and right arms of each pixel $\mathbf{q}$ in $\mathcal{U}^{\mathbf{p}} \cup \mathcal{B}^{\mathbf{p}}$ are included in the support region.

Next, we aggregate matching costs by averaging the costs in the support region. In other words, the aggregated cost at $\mathbf{p}$ is given by

$$C_{\text{ag}}(\mathbf{p}, \mathbf{d}) = \frac{1}{|\mathcal{S}^{\mathbf{p}}|} \sum_{\mathbf{q} \in \mathcal{S}^{\mathbf{p}}} C(\mathbf{q}, \mathbf{d}) \tag{13}$$

where $|\mathcal{S}^{\mathbf{p}}|$ is the number of pixels in the support region $\mathcal{S}^{\mathbf{p}}$.

B. DISPARITY OPTIMIZATION

After aggregating matching costs, we optimize a disparity map. Instead of applying a global optimization method to maximize the matching performance at the cost of high complexity, we perform semiglobal optimization, which is computationally simpler but provides comparable performance. We adopt the semiglobal matching (SGM) [18], while enforcing adaptive smoothness constraints in non-textured regions and on depth edges [11].

1) SEMIGLOBAL MATCHING

To obtain a dense disparity map D, an energy function $E(D)$ is defined, which is composed of data and smoothness terms in general. In particular, in [18], the energy function is defined as

$$\begin{aligned} E(D) = \sum_{\mathbf{p}} (& C_{\text{ag}}(\mathbf{p}, D_{\mathbf{p}}) \\ & + \sum_{\mathbf{q} \in \mathcal{N}_{\mathbf{p}}} P_1 \cdot T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1] \\ & + \sum_{\mathbf{q} \in \mathcal{N}_{\mathbf{p}}} P_2 \cdot T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]) \end{aligned} \tag{14}$$

where P_1 and P_2 are smoothness penalties, and $P_1 < P_2$. Also, $T[\cdot]$ is the indicator function, $D_{\mathbf{p}}$ is the disparity at $\mathbf{p}$, and $\mathcal{N}_{\mathbf{p}}$ is the set of adjacent pixels to $\mathbf{p}$. The first term is the data cost. The second term imposes a small penalty P_1 at pixel $\mathbf{p}$ whose disparity differs by one from the disparity of an adjacent pixel $\mathbf{q}$. Similarly, the third term imposes a larger penalty P_2 at $\mathbf{p}$ whose disparity differs by more than one from an adjacent disparity.

By applying the SGM technique [18], which decreases a global energy based on the path-wise optimization, we attempt to minimize the energy function $E(D)$ efficiently. For the energy function $E(D)$ in (14), the path-wise matching cost in a direction $\mathbf{r}$ is formulated recursively by

$$\begin{aligned} L_{\mathbf{r}}(\mathbf{p}, \mathbf{d}) = \; & C_{\text{ag}}(\mathbf{p}, \mathbf{d}) \\ & + \min\{L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, \mathbf{d}), \min_{\mathbf{i}} L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, \mathbf{i}) + P_2 \\ & \qquad L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, \mathbf{d}_{-1}) + P_1, L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, \mathbf{d}_{+1}) + P_1\} \\ & - \min_{\mathbf{k}} L_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, \mathbf{k}). \end{aligned} \tag{15}$$

where $\mathbf{d}_a = (d + a, 0)$, and $L_{\mathbf{r}}(\mathbf{p}, \mathbf{d})$ for $\mathbf{p}$ outside the image boundary is set to be infinity. The last subtracting term is for preventing $L_{\mathbf{r}}(\mathbf{p}, d)$ from becoming too large. Unlike [18], we use only four directions $\mathbf{r} = (1, 0)$, $(-1, 0)$, $(0, 1)$, and $(0, -1)$. Two of them are horizontal, while the other two are vertical. We also set the penalties P_1 and P_2 according to the intensity differences between pixels $\mathbf{p}$ and $\mathbf{p} - \mathbf{r}$ in the left image and pixels $\mathbf{p} - \mathbf{d}$ and $\mathbf{p} - \mathbf{d} - \mathbf{r}$ in the right image, as described in [2] and [20].

2) ADAPTIVE SMOOTHNESS CONSTRAINT IN NON-TEXTURED REGIONS

In general, non-textured regions, which are flat in pixel intensities, also have homogeneous disparities. However, the matching cost computation, including the proposed one, in non-textured regions is unreliable in general, and the disparity estimation in such regions is one of the major challenges in dense stereo matching. Hence, to improve the matching performance, we detect non-textured regions and apply an adaptive smoothness constraint in those regions.

To identify non-textured regions, we extract the gradient maps ∇I_x and ∇I_y in the x and y directions from the input image I, which are normalized by the mean subtraction and the standard deviation division. Then, we obtain a set

(a) (b)

FIGURE 6. (a) An input image and (b) its gradient map, where green pixels represent the non-textured region $\mathcal{P}_{\mathrm{NT}}$.

$\mathcal{P}_{\mathrm{g}} = \{\mathbf{p} : |\nabla I_x(\mathbf{p})| \geq \tau_{\mathrm{g}} \text{ or } |\nabla I_y(\mathbf{p})| \geq \tau_{\mathrm{g}}\}$ with a threshold $\tau_{\mathrm{g}} = 0.1$. We find a window, which does not include any pixel in $\mathcal{P}_{\mathrm{g}}$, and declare that all pixels in the window belong to the non-textured region. In other words, the non-textured region $\mathcal{P}_{\mathrm{NT}}$ is defined as

$$\mathcal{P}_{\mathrm{NT}} = \left\{ \mathbf{p} : {}^{\exists} W_{\mathbf{p}} \text{ such that } \sum_{\mathbf{q} \in W_{\mathbf{p}}} T[\mathbf{q} \in \mathcal{P}_{\mathrm{g}}] = 0 \right\} \quad (16)$$

where $W_{\mathbf{p}}$ is a 31×31 window including pixel $\mathbf{p}$. Also, to reduce false detections, the pixels within ten pixel distances from $\mathbf{q} \in \mathcal{P}_{\mathrm{g}}$ are excluded from the non-textured region, as illustrated in Fig. 6.

As mentioned above, in the non-textured region $\mathcal{P}_{\mathrm{NT}}$, disparities tend to be homogeneous or continuous. Hence, in $\mathcal{P}_{\mathrm{NT}}$, we set a large penalty $\hat{P}_2$, instead of P_2 in (14), for a large disparity difference between adjacent pixels by

$$\hat{P}_2(\mathbf{p}) = \begin{cases} P_2 & \text{if } \mathbf{p} \notin \mathcal{P}_{\mathrm{NT}}, \\ \sigma_{\mathrm{NT}} \cdot P_2 & \text{if } \mathbf{p} \in \mathcal{P}_{\mathrm{NT}}, \end{cases} \quad (17)$$

where σ_{NT} is an amplification factor greater than 1, which is fixed to $\sigma_{\mathrm{NT}} = 2$ in this work.

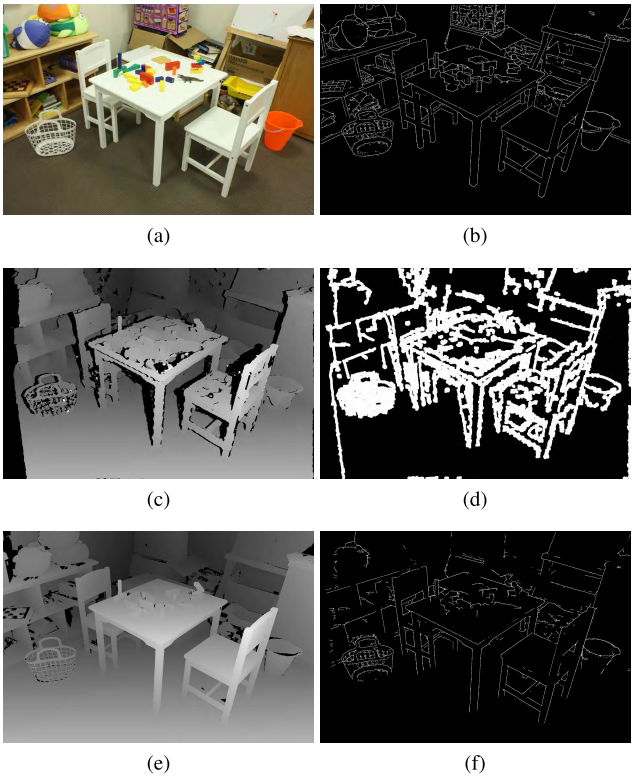
3) ADAPTIVE SMOOTHNESS CONSTRAINT ON DEPTH EDGES

We develop another adaptive smoothness constraint, which employs a depth edge map. We exploit the tendency that depth discontinuity occurs at intensity edges in an input image. However, intensity edges do not always imply depth discontinuity. Therefore, we detect depth edges selectively from the set of color edges, similarly to [47], but using a simpler scheme. Notice that a pre-estimated disparity map using the matching costs in Section V-A can inform of depth discontinuity roughly. The pre-estimated disparity map is defined as

$$D^{\mathrm{pre}}(\mathbf{p}) = \arg\min_{\mathbf{d}} C_{\mathrm{ag}}(\mathbf{p}, \mathbf{d}). \quad (18)$$

We use this pre-estimated disparity map to detect depth edges.

We obtain an edge map E_{I} for the input image. We also extract an edge map E_{P} from the pre-estimated disparity map, which is dilated to broaden the edges. For computing both edge maps, we use the sketch tokens in [48]. We obtain two sets of edge pixels $\mathcal{P}_{\mathrm{I}} = \{\mathbf{p} \mid E_{\mathrm{I}}(\mathbf{p}) \geq \epsilon\}$ and $\mathcal{P}_{\mathrm{P}} = \{\mathbf{p} \mid E_{\mathrm{P}}(\mathbf{p}) \geq \epsilon\}$ with a threshold $\epsilon = 0.5$. The set of depth edge pixels $\mathcal{P}_{\mathrm{DE}}$ is then defined as the intersection of $\mathcal{P}_{\mathrm{I}}$ and $\mathcal{P}_{\mathrm{P}}$, $\mathcal{P}_{\mathrm{DE}} = \mathcal{P}_{\mathrm{I}} \cap \mathcal{P}_{\mathrm{P}}$. Fig. 7 shows step-by-step results of detecting the depth edges. We see in Fig. 7(f) that the depth edges faithfully indicate large differences between neighboring disparities.



(a) (b) (c) (d) (e) (f)

FIGURE 7. The extraction of depth edges: (a) input image I, (b) its edge pixels $\mathcal{P}_{\mathrm{I}}$, (c) pre-estimated disparity image D^{pre}, (d) its edge pixels $\mathcal{P}_{\mathrm{P}}$, (e) ground-truth depth map, and (f) depth edge pixels $\mathcal{P}_{\mathrm{DE}}$.

For depth edge pixels in $\mathcal{P}_{\mathrm{DE}}$, the probability of a large difference between adjacent disparities is very high. Therefore, contrary to ordinary pixels, for those pixels in $\mathcal{P}_{\mathrm{DE}}$, we set a small penalty for a large difference of adjacent disparities and a large penalty for a small difference. To this end, by adding a new smoothness term, we modify the energy function in (14) to

$$\begin{aligned} \hat{E}(D) = \sum_{\mathbf{p}} (& C_{\mathrm{ag}}(\mathbf{p}, D_{\mathbf{p}}) \\ & + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_0^{\mathrm{E}}(\mathbf{p}) \cdot T[D_{\mathbf{p}} = D_{\mathbf{q}}] \\ & + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_1^{\mathrm{E}}(\mathbf{p}) \cdot T[|D_{\mathbf{p}} - D_{\mathbf{q}}| = 1] \\ & + \sum_{\mathbf{q} \in N_{\mathbf{p}}} P_2^{\mathrm{E}}(\mathbf{p}) \cdot T[|D_{\mathbf{p}} - D_{\mathbf{q}}| > 1]) \end{aligned} \quad (19)$$

and determine the three adaptive penalties by

$$P_0^{\mathrm{E}}(\mathbf{p}) = \begin{cases} 0 & \text{if } \mathbf{p} \notin \mathcal{P}_{\mathrm{DE}}, \\ \hat{P}_2(\mathbf{p}) & \text{if } \mathbf{p} \in \mathcal{P}_{\mathrm{DE}}, \end{cases} \quad (20)$$

TABLE 3. Step-by-step comparison of the proposed algorithm with the conventional MC-CNN algorithm [2] on the 2014 Middlebury stereo training set.

Network	Baseline		Baseline + PP		CBCA		Optimization	
	2.0 Err	Avg Err	2.0 Err	Avg Err	2.0 Err	Avg Err	2.0 Err	Avg Err
MC-CNN-fst [2]	25.78%	14.95	21.03%	10.30	19.92%	9.99	11.99%	3.81
MC-CNN-acrt [2]	23.96%	15.13	18.86%	9.54	15.70%	8.16	10.21%	3.52
Network A	18.58%	7.37	14.15%	4.56	**12.07%**	**3.50**	**9.93%**	**2.08**
Network B	18.69%	7.45	14.22%	4.60	12.10%	**3.50**	9.97%	2.10
Network C	**16.49%**	**5.93**	**13.59%**	**4.22**	12.52%	3.57	11.19%	2.35
Network D	19.49%	8.26	15.09%	5.44	12.23%	3.63	10.04%	2.18

$$P_1^{\mathrm{E}}(\mathbf{p}) = \begin{cases} P_1 & \text{if } \mathbf{p} \notin \mathcal{P}_{\mathrm{DE}}, \\ \hat{P}_2(\mathbf{p}) & \text{if } \mathbf{p} \in \mathcal{P}_{\mathrm{DE}}, \end{cases} \tag{21}$$

$$P_2^{\mathrm{E}}(\mathbf{p}) = \begin{cases} \hat{P}_2(\mathbf{p}) & \text{if } \mathbf{p} \notin \mathcal{P}_{\mathrm{DE}}, \\ P_1 & \text{if } \mathbf{p} \in \mathcal{P}_{\mathrm{DE}}. \end{cases} \tag{22}$$

Note that $\hat{P}_2(\mathbf{p})$ defined in (17) is larger than P_1. Therefore, neighboring disparity differences of 0 or 1 are strongly penalized for depth edge pixels in $\mathcal{P}_{\mathrm{DE}}$, whereas differences larger than 1 are strongly penalized for ordinary pixels in $\mathcal{P}_{\mathrm{DE}}^{\mathsf{c}}$.

Similarly to (15), we minimize $\hat{E}(D)$ by formulating the path-wise matching costs recursively as

$$\begin{aligned} \hat{L}_{\mathbf{r}}(\mathbf{p}, \mathbf{d}) = C_{\mathrm{ag}}(\mathbf{p}, \mathbf{d}) - \min_{\mathbf{k}} \hat{L}_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, \mathbf{k}) \\ + \min\{\hat{L}_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, \mathbf{d}) + P_0^{\mathrm{E}}(\mathbf{p}), \\ \hat{L}_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, \mathbf{d}_{-1}) + P_1^{\mathrm{E}}(\mathbf{p}), \\ \hat{L}_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, \mathbf{d}_{+1}) + P_1^{\mathrm{E}}(\mathbf{p}), \\ \min_{\mathbf{i}} \hat{L}_{\mathbf{r}}(\mathbf{p} - \mathbf{r}, \mathbf{i}) + P_2^{\mathrm{E}}(\mathbf{p})\}. \end{aligned} \tag{23}$$

Then, the dense disparity map D^* is optimized by summing the path-wise matching costs in all four directions and minimizing the sum,

$$D_{\mathbf{p}}^* = \arg\min_{\mathbf{d}} \sum_{\mathbf{r}} \hat{L}_{\mathbf{r}}(\mathbf{p}, \mathbf{d}). \tag{24}$$

4) POST-PROCESSING

We refine this disparity map in a post-processing step. First, for the left-right consistency check [49], the left disparity map D^{L} is computed by considering the left image as a reference image, and the right disparity map D^{R} is computed similarly in a symmetrical manner. Let $\mathbf{p}_d$ denote the corresponding pixel in the right image that matches pixel $\mathbf{p}$ in the left image. If $\left| D_{\mathbf{p}}^{\mathrm{L}} + D_{\mathbf{p}_d}^{\mathrm{R}} \right| > 1$, the disparity of $\mathbf{p}$ is regarded as erroneous and interpolated using those of neighboring pixels. Then, the subpixel estimation [50] is performed through the quadratic curve fitting of neighboring costs. Finally, a 5×5 median filter and a bilateral filter are applied subsequently.

VI. EXPERIMENTAL RESULTS

A. EXPERIMENTAL SETTING

We train each network in Fig. 2 or Fig. 4 in three steps. Let us describe the training procedure for network A, B, or D. First, we train the front part of the network from the first to the seventh convolutional layers with three losses for network A or B, while with five losses for network D. Before the training, we initialize the parameters of the first five convolutional layers using those of the MC-CNN-acrt network in [2]. Second, we train the rest of the network while fixing the front part in the first step. Third, we train the entire network jointly. In the case of network C, the trained network B is used as the front part. After fixing the front part, the fully connected layers are trained. Finally, the entire network is trained together.

We perform each training step via the stochastic gradient descent with a momentum of 0.9 and a batch size of 64 examples for 14 epochs. The learning rate is 0.003 for the first eleven epochs and 0.0003 for the last three epochs. To train each network, we use the triplet network structure as illustrated in Fig. 1. The training is performed in a single NVIDIA GeForce GTX 1080 Ti GPU. In order to train the networks and evaluate the stereo matching results, we use the Middlebury datasets [8]. The Middlebury stereo benchmark is composed of 2001, 2003, 2005, 2006, and 2014 datasets, which have full, half, and quarter resolutions. We use the half resolution. In these datasets, 64 image pairs and their ground truth disparity vectors are available. Among them, we select 15 image pairs as a test set and use the other image pairs to train the four networks. Specifically, 26M patch pairs are used for training.

The proposed algorithm has five parameters, which are fixed in all experiments. Specifically, $\tau = 0.04$ in (10), $\eta = 11$ in (11), $P_1 = 1$ and $P_2 = 45$ in (14), and $\sigma_{\mathrm{NT}} = 2$ in (17). To assess stereo matching results, we adopt the average disparity error (Avg Err) and the percentage of disparity errors (2.0 Err). In 2.0 Err, an estimated disparity is declared as erroneous, if the distance between the estimated disparity and its ground truth is larger than 2 pixels in the full resolution.

B. COMPARATIVE PERFORMANCE EVALUATION

Table 3 compares the four proposed networks with the conventional stereo matching algorithm MC-CNN [2] on the 2014 Middlebury dataset. MC-CNN has two versions: MC-CNN-fst for fast processing and MC-CNN-acrt for accurate estimation. To obtain the MC-CNN results, we use the provided source code[1] and train the two versions using the

[1] https://github.com/jzbontar/mc-cnn

same set of image pairs and ground truth that we employ to train the proposed four networks.

We evaluate the stereo matching performance after each step: pure matching cost (PMC) computation, CBCA, optimization, and post-processing. To this end, we perform stereo matching using four methods:

- Baseline: PMC + winner-takes-all (WTA),
- Baseline + PP: PMC + WTA + post-processing,
- CBCA: PMC + CBCA + WTA + post-processing,
- Optimization: PMC + CBCA + optimization + post-processing.

Note that WTA selects the disparity, corresponding to the minimum matching cost, for each pixel. For the optimization step, we adopt NTDE [11] in Section V-B. For MC-CNN, we use its optimization method based on SGM in [2].

In Table 3, the proposed networks A, B, and D outperform the conventional MC-CNN for all four steps. Also, network C provides better performances than MC-CNN except for the 'Optimization' step. Note that, for straightforward stereo matching without sophisticated cost aggregation and disparity optimization, it is beneficial to have a large receptive field. In networks A, B, and C, the maximum receptive field size is 39×39. On the other hand, the receptive field sizes of MC-CNN-fst and MC-CNN-acrt are 9×9 and 11×11, respectively. Therefore, for the 'Baseline' and 'Baseline + PP' steps, networks A, B, and C outperform MC-CNN significantly. Even though the receptive size of network D is 15×15 only, it provides competitive results to the other proposed networks, by employing the fully-connected layers effectively. For the 'CBCA' and 'Optimization' steps, the performance gaps between the proposed algorithm and the conventional MC-CNN decrease, but network A still achieves the best performances for the 'CBCA' and 'Optimization' steps.

Network A outperforms network B in all cases, even though both networks provide multiscale features of the same receptive field sizes. This is because network A contains the additional convolutional layers in the sub-networks. Thus, the features extracted from network A are more effective. Also, the competitive performances of networks C and D indicate that the fully-connected layers successively replace the matching cost combination technique based on edge information. Note that network C surpasses network D for the 'Baseline' and 'Baseline + PP' steps, since it has the larger receptive field size of 39×39.

Table 4 compares the matching cost computation performances of the proposed algorithm with those of the conventional CNN-based algorithms. To this end, the 'Baseline' method is used to compute disparities. We also compare the computational times. In the 'Ours' column, we report the running times that we measured in the same environment. In the other column, we list the time measurements reported in [7]. In Table 4, the algorithms are divided into two categories. The algorithms in the upper part focus on accuracy rather than on computational complexity, and thus they use fully-connected layers in the matching cost computation. In contrast, the algorithms in the lower part extract features from the networks and then compute matching costs simply via the inner product. So they are faster in general. Note that Ye *et al.* (accurate) [7] yields the smallest error. However, the proposed four networks provide smaller errors than Ye *et al.* (fast) and MC-CNN-fst [2] in the fast category. The running times of the proposed networks are longer than MC-CNN-fst [2], since the proposed networks need additional computations for combining multiscale information. However, despite these computations, the proposed networks are faster, as well as more accurate, than MC-CNN-acrt. Ye *et al.* [7] improves the performance of MC-CNN-acrt, but the proposed networks are more accurate than their fast version.

TABLE 4. Matching cost computation performances of the proposed algorithm and the conventional CNN-based algorithms. In this test, the 'Baseline' method is used to compute disparities.

	Baseline		
	2.0 Err	Times (s)	
		Ours	[7]
MC-CNN-acrt [2]	23.96%	**106.9**	**41.5**
LW-CNN [6]	11.85%	-	229.8
Ye *et al.* (accurate) [7]	**11.60%**	-	43.7
MC-CNN-fst [2]	25.78%	**0.43**	**0.53**
Ye *et al.* (fast) [7]	22.09%	-	1.4
Network A	18.58%	13.29	-
Network B	18.69%	10.05	-
Network C	**16.49%**	11.34	-
Network D	19.49%	4.66	-

C. MULTISCALE FEATURES

Next, Table 5 lists the stereo matching performances using multiscale features of network A. Network A generates seven multiscale features of different receptive field sizes. CostAvg denotes the case of averaging the seven matching costs without adaptive weights, while the proposed network A uses the adaptive weighting scheme in Section III-D. We observe that the matching performance generally improves as the

TABLE 5. Stereo matching performances using multiscale features of network A.

Receptive field	Baseline + PP		Optimization	
	2.0 Err	Avg Err	2.0 Err	Avg Err
7×7	22.87%	9.89	11.73%	2.88
11×11	17.82%	7.25	10.52%	2.40
15×15	15.87%	6.13	10.20%	2.21
21×21	14.72%	5.10	10.60%	2.23
27×27	14.17%	4.62	10.81%	2.25
33×33	**14.12%**	4.49	11.23%	2.36
39×39	14.13%	**4.43**	11.49%	2.42
CostAvg	14.40%	5.36	10.37%	2.24
Network A	14.15%	4.56	**9.93%**	**2.08**

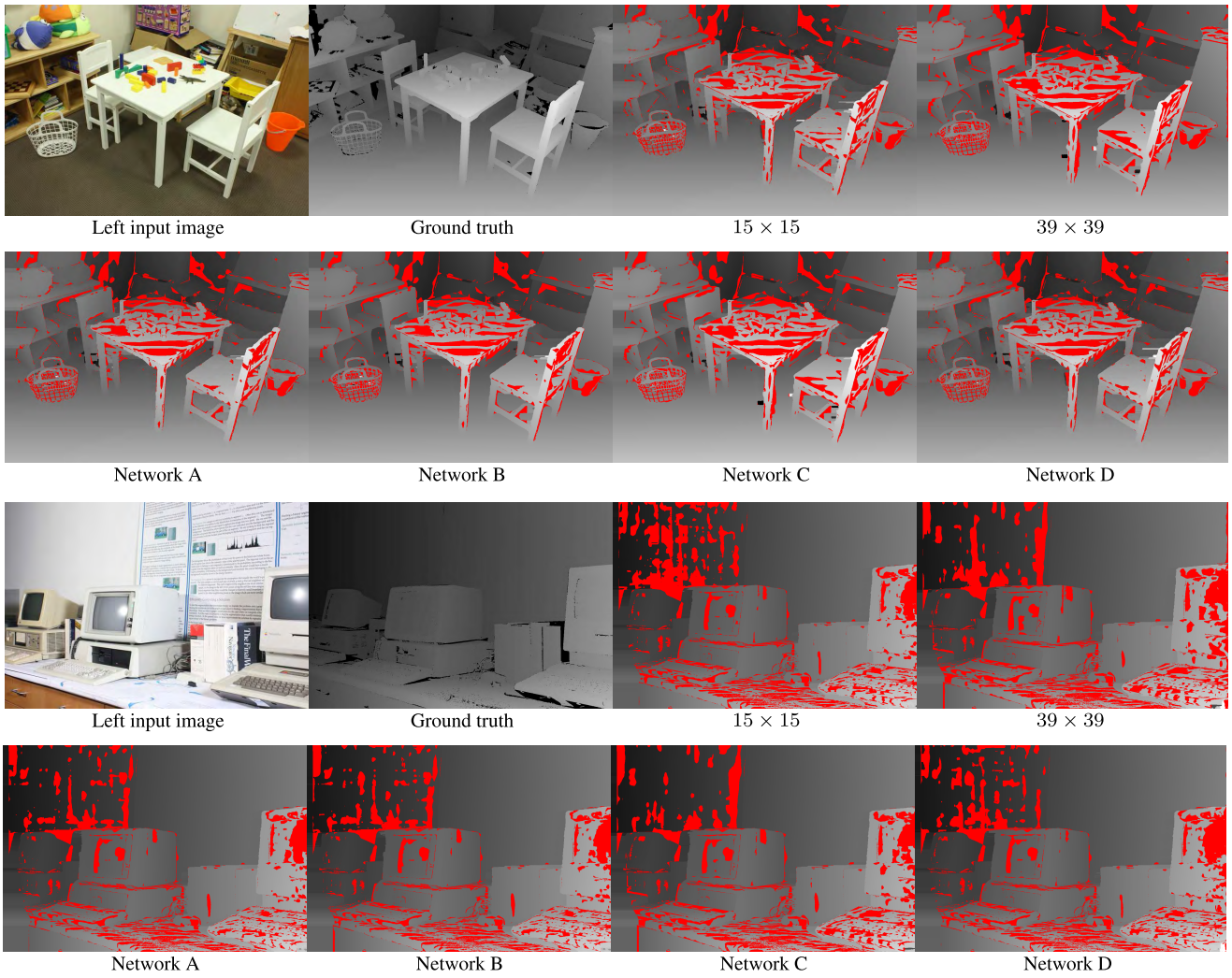


FIGURE 8. Examples of estimated disparities on the Middlebury stereo datasets. In this test, the 'Optimization' method is used, and 15 × 15 and 39 × 39 features are extracted from network A. Red pixels depict disparity errors, which are larger than 2 pixels. The other pixels depict estimated disparities in gray levels.

receptive field gets larger, but it saturates at about 27 × 27. The combination of multiscale features is not effective in the simple 'Baseline + PP' method, but it improves the performance significantly in the 'Optimization' method.

In the 'Baseline + PP' method, a larger receptive field is more effective on average. However, a small receptive field is also useful in some cases. Specifically, features of small receptive fields provide good matching performance on regions with complicated texture, whereas those of large receptive fields are effective on flat regions. Thus, multiscale features have their own pros and cons. To exploit their pros, we combine the matching costs, computed from multi-scale features, adaptively using edge information. Thus, network A achieves the best performance in the 'Optimization' method in Table 5.

D. QUALITATIVE RESULTS

Fig. 8 illustrates estimated disparities on the 'Playtable' and 'Vintage' image pairs. For both pairs, the 39 × 39 receptive field yields more reliable matching results than the 15 × 15 receptive field on textureless regions. Also, notice that all proposed networks estimate disparities more accurately than single features, by combining multiscale features.

E. EVALUATION ON MIDDLEBURY STEREO BENCHMARK

Table 6 shows the evaluation results on the Middlebury stereo training dense set benchmark.[2] We compare the proposed networks A, B, C, and D with the fourteen published methods that yield the lowest 2.0 Err results. It is worthy to point out that networks A, B and D are much faster than the top ten compared methods. For example, network A outperforms MC-CNN-acrt [2] and is about 4 times faster. These results indicate that the proposed algorithm provides competitive stereo matching performance at a relatively low computational cost.

[2]http://vision.middlebury.edu/stereo/eval3/

TABLE 6. Comparison of the proposed algorithms with the fourteen conventional methods on the Middlebury stereo training dense set benchmark.

Methods	2.0 Err	Avg Err	Time (s)
LocalExp [51]	**6.52%**	1.69	846
3DMST [52]	7.08%	1.89	167
FEN-D2DRR [7]	7.89%	**1.26**	73.3
PMSC [36]	8.20%	2.12	579
LW-CNN [6]	8.31%	2.41	224
MeshStereoExt [22]	9.32%	2.32	133
Network A	9.93%	2.08	21.3
NTDE [11]	9.94%	2.19	128
Network B	9.97%	2.10	16.3
Network D	10.0%	2.18	10.9
MC-CNN-acrt [2]	10.1%	3.81	106
MC-CNN+TDSR [53]	10.2%	2.28	575
SNP-RSM [54]	10.8%	2.44	246
MC-CNN+RBS [55]	10.8%	2.60	157
JMR [56]	10.8%	2.30	4.28
Network C	11.2%	2.35	20.3
MC-CNN-fst [2]	11.7%	3.87	**1.26**
MDP [57]	13.7%	4.75	56.9

VII. CONCLUSIONS

In this work, we proposed four CNN-based feature extractors (networks A, B, C, and D) for stereo matching cost computation. For training each network, we adopted a triplet structure, which comprises three identical CNNs. To extract multiscale features, we added the hinge marginal loss function to hidden convolutional layers, as well as at the end of the network. To find optimal matching costs, we combined multiscale matching costs, which are extracted by network A or B using edge information. On the other hand, we used uni-scale features from network C or D to compute matching costs. After obtaining the matching costs, we carried out the cross-based cost aggregation [28] and optimized disparities using the NTDE method [11].

Experimental results demonstrated that the proposed algorithm provides competitive stereo matching performance with the state-of-the-art algorithms, while requiring lower computational complexity. Among the four proposed networks, network A yields the best performance for the 'Optimization' step, while network C performs the best for the 'Baseline' step. On the other hand, considering the trade-off between performance and complexity, network D is the most efficient architecture for stereo matching with a negligible loss in the matching accuracy.

REFERENCES

[1] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vis.*, vol. 47, nos. 1–3, pp. 7–42, Apr. 2002.

[2] J. Žbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 2287–2318, Jan. 2016.

[3] Z. Chen, X. Sun, L. Wang, Y. Yu, and C. Huang, "A deep visual correspondence embedding model for stereo matching costs," in *Proc. ICCV*, Dec. 2015, pp. 972–980.

[4] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proc. CVPR*, Jun. 2016, pp. 5696–5703.

[5] A. Seki and M. Pollefeys, "Patch based confidence prediction for dense disparity map," in *Proc. BMVC*, 2016, pp. 1–13.

[6] H. Park and K. M. Lee, "Look wider to match image patches with convolutional neural networks," *IEEE Signal Process. Lett.*, vol. 24, no. 12, pp. 1788–1792, Dec. 2017.

[7] X. Ye, J. Li, H. Wang, H. Huang, and X. Zhang, "Efficient stereo matching leveraging deep local and context information," *IEEE Access*, vol. 5, pp. 18745–18755, 2017.

[8] 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 *Proc. GCPR*, 2014, pp. 31–42.

[9] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. CVPR*, Jun. 2012, pp. 3354–3361.

[10] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proc. CVPR*, Jun. 2015, pp. 3061–3070.

[11] K.-R. Kim and C.-S. Kim, "Adaptive smoothness constraints for efficient stereo matching using texture and edge information," in *Proc. ICIP*, Sep. 2016, pp. 3429–3433.

[12] S. Birchfield and C. Tomasi, "A pixel dissimilarity measure that is insensitive to image sampling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 4, pp. 401–406, Apr. 1998.

[13] T. Kanade and M. Okutomi, "A stereo matching algorithm with an adaptive window: Theory and experiment," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 9, pp. 920–932, Sep. 1994.

[14] A. Fusiello, V. Roberto, and E. Trucco, "Efficient stereo with multiple windowing," in *Proc. CVPR*, Jun. 1997, pp. 858–863.

[15] D. Kong and H. Tao, "A method for learning matching errors in stereo computation," in *Proc. BMVC*, 2004, pp. 1–10.

[16] D. Kong and H. Tao, "Stereo matching via learning multiple experts behaviors," in *Proc. BMVC*, 2006, pp. 1–10.

[17] A. Klaus, M. Sormann, and K. Karner, "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure," in *Proc. IEEE ICPR*, Aug. 2006, pp. 15–18.

[18] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 2, pp. 328–341, Feb. 2008.

[19] H. Hirschmüller and D. Scharstein, "Evaluation of stereo matching costs on images with radiometric differences," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 9, pp. 1582–1599, Sep. 2009.

[20] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, and X. Zhang, "On building an accurate stereo matching system on graphics hardware," in *Proc. IEEE ICCV Workshops*, Nov. 2011, pp. 467–474.

[21] K. Yamaguchi, T. Hazan, D. McAllester, and R. Urtasun, "Continuous Markov random fields for robust stereo estimation," in *Proc. ECCV*, Oct. 2012, pp. 45–58.

[22] C. Zhang, Z. Li, Y. Cheng, R. Cai, H. Chao, and Y. Rui, "MeshStereo: A global stereo model with mesh alignment regularization for view interpolation," in *Proc. ICCV*, Dec. 2015, pp. 2057–2065.

[23] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Proc. ECCV*, 1994, pp. 151–158.

[24] K.-J. Yoon and I. S. Kweon, "Adaptive support-weight approach for correspondence search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 4, pp. 650–656, Apr. 2006.

[25] D. Chen, M. Ardabilian, and L. Chen, "A fast trilateral filter-based adaptive support weight method for stereo matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 5, pp. 730–743, May 2015.

[26] 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.*, vol. 35, no. 2, pp. 504–511, Feb. 2013.

[27] K. He, J. Sun, and X. Tang, "Guided image filtering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 6, pp. 1397–1409, Jun. 2013.

[28] K. Zhang, J. Lu, and G. Lafruit, "Cross-based local stereo matching using orthogonal integral images," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 7, pp. 1073–1079, Jul. 2009.

[29] S. Birchfield and C. Tomasi, "Depth discontinuities by pixel-to-pixel stereo," *Int. J. Comput. Vis.*, vol. 35, no. 3, pp. 269–293, 1999.

[30] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1222–1239, Nov. 2001.

[31] J. Sun, N.-N. Zheng, and H.-Y. Shum, "Stereo matching using belief propagation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 7, pp. 787–800, Jul. 2003.
[32] S. N. Sinha, D. Scharstein, and R. Szeliski, "Efficient high-resolution stereo matching using local plane sweeps," in *Proc. CVPR*, Jun. 2014, pp. 1582–1589.
[33] M. Bleyer, C. Rhemann, and C. Rother, "Patchmatch stereo-stereo matching with slanted support windows," in *Proc. BMVC*, 2011, pp. 1–11.
[34] L. Hong and G. Chen, "Segment-based stereo matching using graph cuts," in *Proc. CVPR*, Jun. 2004, pp. 74–81.
[35] K. Yamaguchi, D. McAllester, and R. Urtasun, "Efficient joint segmentation, occlusion labeling, stereo and flow estimation," in *Proc. ECCV*, Sep. 2014, pp. 756–771.
[36] L. Li, S. Zhang, X. Yu, and L. Zhang, "PMSC: Patchmatch-based superpixel cut for accurate stereo matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 3, pp. 679–692, Mar. 2016.
[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recogntion," in *Proc. ICLR*, 2015, pp. 1–12.
[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Jun. 2016, pp. 770–778.
[40] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, Jun. 2014, pp. 580–587.
[41] H.-U. Kim and C.-S. Kim, "CDT: Cooperative detection and tracking for tracing multiple objects in video sequences," in *Proc. ECCV*, 2016, pp. 851–867.
[42] Y. J. Koh and C.-S. Kim, "CDTS: Collaborative detection, tracking, and segmentation for online multiple object segmentation in videos," in *Proc. ICCV*, Oct. 2017, pp. 3621–3629.
[43] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. CVPR*, Jun. 2015, pp. 3431–3440.
[44] Y. J. Koh and C.-S. Kim, "Primary object segmentation in videos based on region augmentation and reduction," in *Proc. CVPR*, Jul. 2017, pp. 7417–7425.
[45] S. Xie and Z. Tu, "Holistically-nested edge detection," in *Proc. ICCV*, 2015, pp. 1395–1403.
[46] J. Wang *et al.*, "Learning fine-grained image similarity with deep ranking," in *Proc. CVPR*, Jun. 2014, pp. 1386–1393.
[47] D. Chen, M. Ardabilian, and L. Chen, "Depth edge based trilateral filter method for stereo matching," in *Proc. ICIP*, Sep. 2015, pp. 2280–2284.
[48] J. J. Lim, C. L. Zitnick, and P. Dollár, "Sketch tokens: A learned mid-level representation for contour and object detection," in *Proc. CVPR*, Jun. 2013, pp. 3158–3165.
[49] S. D. Cochran and G. Medioni, "3-D surface description from binocular stereo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 10, pp. 981–994, Oct. 1992.
[50] Q. Yang, L. Wang, R. Yang, H. Stewénius, and D. Nistér, "Stereo matching with color-weighted correlation, hierarchical belief propagation and occlusion handling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 3, pp. 492–504, Mar. 2009.
[51] T. Taniai, Y. Matsushita, Y. Sato, and T. Naemura, "Continuous 3D label stereo matching using local expansion moves," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published, doi: 10.1109/TPAMI.2017.2766072.
[52] L. Li, X. Yu, S. Zhang, X. Zhao, and L. Zhang, "3D cost aggregation with multiple minimum spanning trees for stereo matching," *Appl. Opt.*, vol. 56, no. 12, pp. 3411–3420, 2017.
[53] S. Drouyer, S. Beucher, M. Bilodeau, M. Moreaud, and L. Sorbier, "Sparse stereo disparity map densification using hierarchical image segmentation," in *Proc. Int. Symp. Math. Morphol.*, 2017, pp. 172–184.
[54] S. Zhang, W. Xie, G. Zhang, H. Bao, and M. Kaess, "Robust stereo matching with surface normal prediction," in *Proc. ICRA*, May/Jun. 2017, pp. 2540–2547.
[55] J. Barron and B. Poole, "The fast bilateral solver," in *Proc. ECCV*, 2016, pp. 617–632.
[56] P. Knöbelreiter, C. Reinbacher, A. Shekhovtsov, and T. Pock, "End-to-end training of hybrid CNN-CRF models for stereo," in *Proc. CVPR*, 2017, pp. 2339–2348.
[57] A. Li, D. Chen, Y. Liu, and Z. Yuan, "Coordinating multiple disparity proposals for stereo computation," in *Proc. CVPR*, Jun. 2016, pp. 4022–4030.

KYUNG-RAE KIM received the B.S. degree in electrical engineering from Korea University, Seoul, South Korea, in 2014, where he is currently pursuing the Ph.D. degree in electrical engineering. His research interests include computer vision and machine learning and especially in the problems of stereo matching.

YEONG JUN KOH (S'13) received the B.S. and Ph.D. degrees in electrical engineering from Korea University, Seoul, South Korea, in 2011 and 2018, respectively. His research interests include computer vision and machine learning and especially in the problems of video object discovery and segmentation.

CHANG-SU KIM (S'95–M'01–SM'05) received the Ph.D. degree in electrical engineering from Seoul National University (SNU). From 2000 to 2001, he was a Visiting Scholar with the Signal and Image Processing Institute, University of Southern California at Los Angeles. From 2001 to 2003, he coordinated the 3-D Data Compression Group, National Research Laboratory for 3-D Visual Information Processing, SNU. From 2003 to 2005, he was an Assistant Professor with the Department of Information Engineering, The Chinese University of Hong Kong. In 2005, he joined the School of Electrical Engineering, Korea University, where he is currently a Professor. He has authored over 250 technical papers in international journals and conferences. His current research interests include image processing and computer vision. He is a member of the Multimedia Systems and Application Technical Committee of the IEEE Circuits and Systems Society. He was an APSIPA Distinguished Lecturer from 2017 to 2018. He received the Distinguished Dissertation Award in 2000 for his Ph.D. degree, the IEEK/IEEE Joint Award for Young IT Engineer of the Year in 2009, and the Best Paper Award for the *Journal of Visual Communication and Image Representation* in 2014. He served as an Editorial Board Member for the *Journal of Visual Communication and Image Representation* and an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING. He is a Senior Area Editor of the *Journal of Visual Communication and Image Representation* and an Associate Editor of the IEEE TRANSACTIONS ON MULTIMEDIA.