Look Wider to Match Image Patches with Convolutional Neural Networks

Haesol Park, and Kyoung Mu Lee

***Abstract*—When a human matches two images, the viewer has a natural tendency to view the wide area around the target pixel to obtain clues of right correspondence. However, designing a matching cost function that works on a large window in the same way is difficult. The cost function is typically not intelligent enough to discard the information irrelevant to the target pixel, resulting in undesirable artifacts. In this paper, we propose a novel convolutional neural network (CNN) module to learn a stereo matching cost with a large-sized window. Unlike conventional pooling layers with strides, the proposed per-pixel pyramid-pooling layer can cover a large area without a loss of resolution and detail. Therefore, the learned matching cost function can successfully utilize the information from a large area without introducing the fattening effect. The proposed method is robust despite the presence of weak textures, depth discontinuity, illumination, and exposure difference. The proposed method achieves near-peak performance on the Middlebury benchmark.**

***Index Terms*—stereo matching,pooling,CNN**

I. Introduction

Most stereo matching methods first compute the matching cost of each pixel with a certain disparity, before optimizing the whole cost volume either globally or locally by using specific prior knowledge [1]. For decades, many researchers have focused on the second step, designing a good prior function and optimizing it [2], [3], [4], [5], [6]. Few studies have been conducted on designing or selecting a better matching cost function.

One of the most widely used matching cost functions is a pixel-wise matching cost function, such as the one used in [7]. Along with sophisticated prior models, it sometimes produces good results, especially in preserving the detailed structures near the disparity discontinuities. However, the function fails when the image contains weakly-textured areas or repetitive textures. In such cases, a window-based matching cost, such as **CENSUS** or **SAD** [8], produces a more reliable and distinctive measurement. The critical shortcoming of window-based matching cost functions is their unreliability around disparity discontinuities. Figure 1 visually illustrates the characteristics of different matching cost measures.

One method to handle this trade-off is to make the window-based versatile to its input patterns [10], [11], [12]. The key idea is making the shape of the matching template adaptive so that it can discard the information from the pixels that are irrelevant to the target pixel. However, knowing the background pixels before the actual matching is difficult, making it a ‘chicken-and-egg’ problem. Therefore, the use of a CNN [13], [14] is appropriate, as it automatically learns the proper shape of the templates for each input pattern.

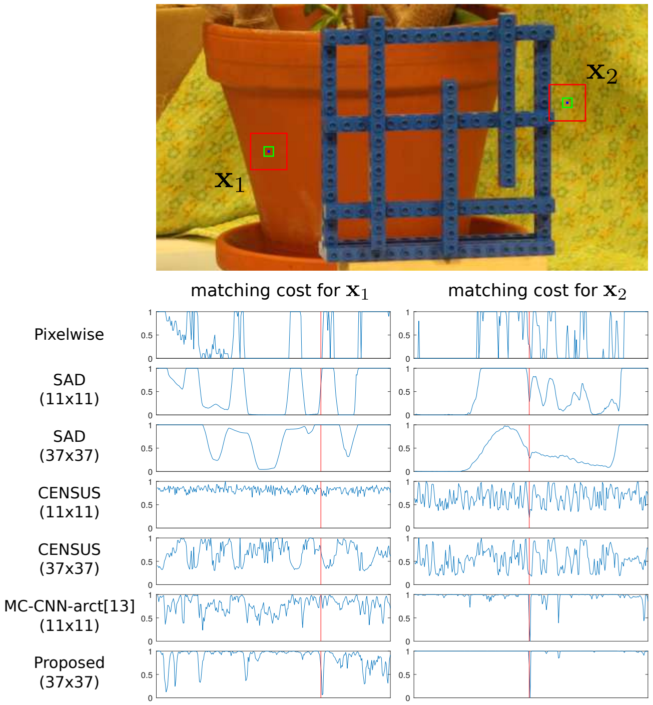

Fig. 1. The top image shows the reference image with two interested points, x_1 and x_2. The pixel positions are marked as blue dots, whereas the red and green boxes represent 37×37 and 11×11 windows centered on them, respectively. At the bottom, the matching costs for each pixel are visualized as a normalized function of disparity for different matching cost functions. The positions of true disparities are marked as red vertical lines. The pixel-wise cost shows the lowest values at the true disparity, but it also gives zero costs for other disparities. The SAD and CENSUS matching cost functions [9] become less ambiguous as the matching window becomes larger. However, these functions are affected by pixels irrelevant to the target pixel ($\mathbf{x}_2$). Even the matching cost learned by using the baseline convolutional neural network (CNN) architecture fails when the surface has a nearly flat texture ($\mathbf{x}_1$). On the other hand, the proposed CNN architecture works well both on weakly textured regions and disparity discontinuities.

The existing methods, however, are based on conventional CNN architectures resembling the **AlexNet** [15] or **VGG** [16] network, which are optimized for image classification task and not for image matching. The architectures comprise several convolution layers, each followed by a rectified linear unit (ReLU) [15], and pooling layers with strides. One of the limitations of using these architectures for matching is the difficulty of enlarging the size of the patches that are to be compared. The effective size of the patch is directly related to

H. Park and K. M. Lee are with Automation and Systems Research Institute, Seoul National University, Seoul 151-744, Korea

the spatial extent of the receptive field of CNN, which can be increased by (1) including a few strided pooling/convolution layers, (2) using larger convolution kernels at each layer, or (3) increasing the number of layers. However, use of strided pooling/convolution layers makes the results downsampled, losing fine details. Although the resolution can be recovered by applying fractional-strided convolution [17], reconstructing small or thin structures is still difficult if once they are lost after downsampling. Increasing the size of the kernels is also problematic, as the number of feature maps required to represent the larger patterns increases significantly. Furthermore, a previous study [18] reported that the repetitive usage of small convolutions does not always result in a large receptive field.

This paper contributes to the literature by proposing a novel CNN module to learn a better matching cost function. The module is an innovative pooling scheme that enables a CNN to view a larger area without losing the fine details and without increasing the computational complexity during test times. The experiments show that the use of the proposed module improves the performance of the baseline network, showing competitive results on the Middlebury [1], [19] benchmark.

II. Related Works

Given the introduction of high-resolution stereo datasets with the ground-truth disparity maps [20], [19], [21], many attempts have been made to learn a matching cost function using machine learning algorithms [13], [14], [22]. The most impressive results are obtained by using CNN [13], [14]. The architecture proposed in [13] takes a small 11×11 window and processes it without the use of pooling. The computed cost volume is noisy due to the limited size of the window. Thus, it is post-processed by using the cross-based cost aggregation (CBCA) [23], semi-global matching (SGM) [3], and additional refinement procedures. On the other hand, the method in [14] uses multiple pooling layers and spatial-pyramid-pooling (SPP) [24] to process larger patches. However, the results show a fattening effect owing to the loss of information introduced by pooling.

The main contribution of this paper is in proposing a novel pooling scheme that can handle information from a large receptive field without losing the fine details. Recently, several attempts have been made to accomplish the same goal in the context of semantic segmentation [25], [26], [27]. These methods combine the feature maps from the high-level layers with those from the lower layers, with the aim of correctly aligning the object-level information along the pixel-level details. While this approach can successfully align the boundaries of the big objects, its inherent limitation is its inability to recover small objects in the final output once they are lost during the abstraction due to multiple uses of pooling. In the same context, the *FlowNet* [28] architecture can upsample the coarse-level flow to the original scale by using lower-level feature maps. However, it fails to recover the extreme flow elements that are hidden due to the low resolution of high-level feature maps.

The architecture most closely related to the current work has been proposed in [24]. Unlike the other approaches, the SPP network excludes pooling layers between convolutional layers. Instead, it first computes highly-nonlinear feature maps by cascading convolutional layers several times and then generates high-level and mid-level information by pooling them at different scales. By keeping the original feature maps along with feature maps pooled at multiple scales, the SPP network can combine the features from multiple levels without losing fine details. Although the previously mentioned stereo method in [14] uses SPP, it also employs conventional pooling layers between convolutional layers, thus losing the detailed information.

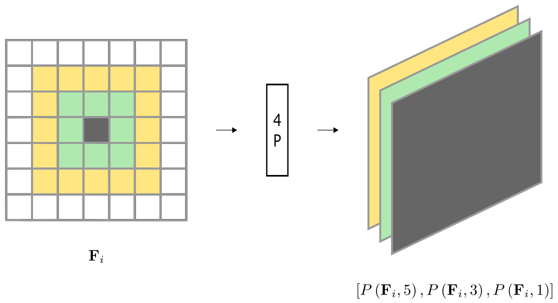

Fig. 2. The **4P** module with pooling size vector $\mathbf{s} = [5, 3, 1]$ is visualized. This figure shows its action for one channel of the feature maps for brevity; it does the same job for all channels.

III. Architecture of the Neural Network

The proposed architecture takes two input patches and produces the corresponding matching cost. In the following subsections, the newly proposed module is first introduced. Then the detailed architecture of the entire network is presented.

A. *Per-pixel Pyramid Pooling (**4P**)*

The use of pooling layers in CNN has been considered desirable because of its accuracy and efficiency in image classification tasks. While the use of max-pooling layers has been reported to provide an additional invariance in spatial transformation, the most important gain comes from the downsampling of feature maps. By performing pooling with a stride that is larger than one, the output feature maps after the pooling are scaled down. The final scale of the CNN output is decreased exponentially in terms of the number of pooling layers. Given that no parameters related to a pooling operation exist, this method is an effective way to widen the receptive field area of a CNN without increasing the number of parameters. The drawback of strided pooling is that the network loses fine details in the original feature maps as the pooling is applied. Thus, a trade-off exists in seeing a larger area and preserving the small details.

Inspired by the idea discussed in [24], we propose a novel pooling scheme to overcome this trade-off. Instead of using a small pooling window with a stride, a large pooling window is used to achieve the desired size of the receptive field. The use of one large pooling window can lead to the loss of finer details. Thus, multiple poolings with varying window sizes are performed, and the outputs are concatenated to

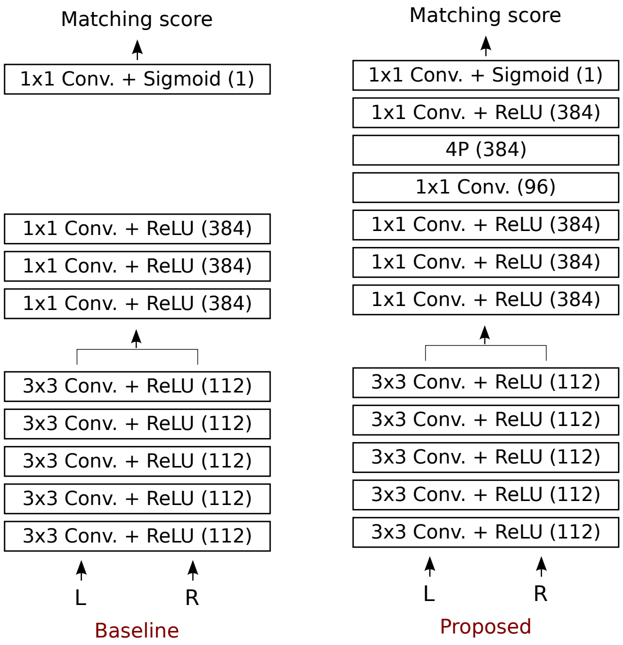

Fig. 3. The network structures are visualized for the baseline network, '**MC-CNN-acrt**' [13], and the proposed network. The parenthesized numbers at each layer represent the number of feature maps after the corresponding operations. Note that this figure is drawn in terms of the fully convolutional network.

create new feature maps. The resulting feature maps contain the information from coarse-to-fine scales. The multi-scale pooling operation is performed for every pixel without strides.

We call this whole procedure, "per-pixel pyramid pooling" (**4P**), which is formally defined as follows:

$$P^{4P}(\mathbf{F}, \mathbf{s}) = [P(\mathbf{F}, s_1), \cdots, P(\mathbf{F}, s_M)], \quad (1)$$

where $\mathbf{s}$ is a vector having M number of elements, and $P(\mathbf{F}, s_i)$ is the pooling operation with size s_i and stride one. The structure of this module is illustrated in Figure 2.

B. Proposed model

To validate the effect of the proposed module, we trained and tested CNNs with and without the **4P** module. The baseline architecture is selected as the '**MC-CNN-acrt**' [13]. The **4P** module in the proposed architecture is constructed by using the size vector $\mathbf{s} = [27, 9, 3, 1]$. The structures of two CNNs are visualized in Figure 3.

IV. Implementation Details

For a fair comparison, we followed the details in [13] to train the proposed architecture with a few exceptions mentioned below. First, the size of the training patch became 37×37. Furthermore, we only fine-tuned the parameters of the last three 1×1 convolution layers of the proposed architecture in Figure 3. The parameters of the earlier layers are borrowed from the pre-trained '**MC-CNN-acrt**' [13] network. In our experiments, this resulted in a better performance than the end-to-end training of the network with random initializations. Moreover, training a few convolution layers with pre-trained features is easier, making it converge faster. We have run a total of four epochs of training, where the last two epochs were run with a decreased learning rate from 0.003 to 0.0003.

TABLE I
THE QUANTITATIVE RESULTS ON THE 'TRAINING DENSE' SET OF THE MIDDLEBURY BENCHMARK [1] ARE SHOWN. THE ERROR REPRESENTS THE PERCENTAGE OF BAD PIXELS WITH A DISPARITY THRESHOLD 2.0, AND THE SAME WEIGHTING SCHEME IS APPLIED AS IN [1] WHEN COMPUTING THE AVERAGE.

Methods		**avg. error**
WTA	**MC-CNN-acrt [13]**	22.91
	proposed	11.75
after post-processing	**MC-CNN-acrt [13]**	10.26
	proposed (w/ parameters in [13])	9.72
	proposed (w/ parameter tuning)	8.45

We also used the same post-processing pipeline as in [13] during the test phase. The post-processing pipeline includes the use of the CBCA [23] and SGM [3], and the disparity maps are refined to have continuous values and undergo median filtering and bilateral filtering.

V. Experiments

To verify the effect of the proposed **4P** module, we have compared the results of the baseline and proposed network. The performance is measured using the 'training dense' set of the Middlebury benchmark [1]. The quantitative results are briefly summarized in Table I using the average errors. All experiments are performed by using the Intel core i7 4790K CPU and a single Nvidia Geforce GTX Titan X GPU.

The proposed method outperforms the baseline architecture regardless of the use of post-processing. The benefit of using the **4P** module is clear when the disparity maps are obtained by using the pixel-wise winner-takes-it-all (WTA) rule without any post-processing. Given that the images in the dataset contain many weakly-textured areas, the small-sized 11×11 window cannot distinguish the true matches from false ones without the aid of post-processing. On the other hand, the proposed architecture effectively sees the larger window, 37×37, by inserting the **4P** module before the final decision layer.

It is less straightforward to understand why the proposed architecture still outperforms the baseline even after post-processing. In that sense, it is worth to mention that the best parameter setting for post-processing of the proposed method largely differ from that of the baseline.[1] The most notable changes from the original parameter setting is that we use much less number of CBCA [23], and it means that multiple uses of CBCA [23] become redundant in the proposed architecture. From this fact, we can interpret the role of **4P** module as adaptive local feature aggregation. Compared to the hand-designed algorithm such as CBCA [23], the influence of neighboring pixels to a certain pixel is automatically learned

[1] Following the conventions in [13], the best parameter setting is as follows: `cbca_num_iterations_1 = 0`, `cbca_num_iterations_2 = 1`, `sgm_P1 = 1.3`, `sgm_P2 = 17.0`, `sgm_Q1 = 3.6`, `sgm_Q2 = 36.0`, and `sgm_V = 1.4`.

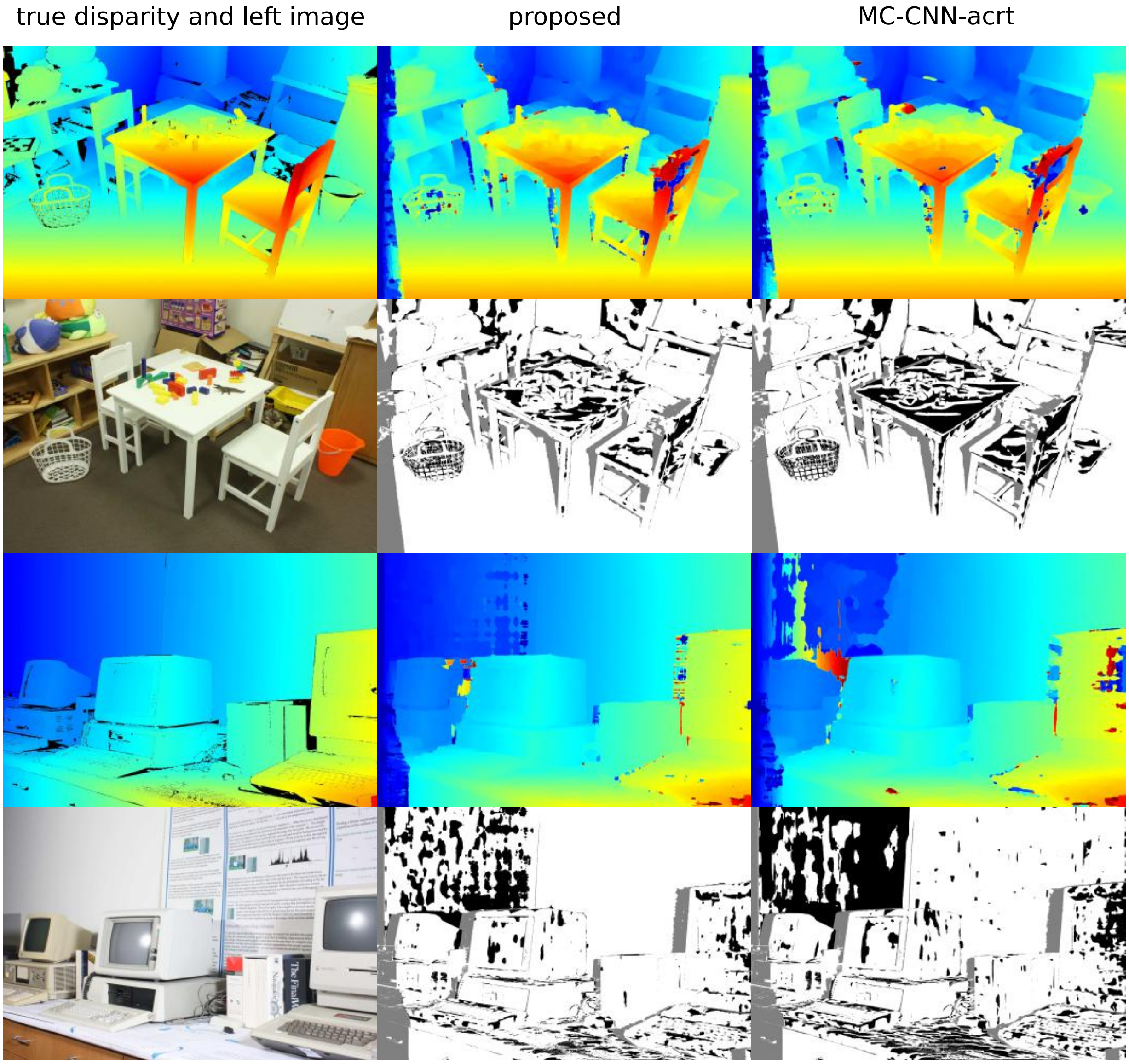

Fig. 4. The results for **PlaytableP** and **Vintage** are visualized. For each datum, the upper row shows the disparity map and the bottom row shows the corresponding error maps. While the '**MC-CNN-acrt**' [13] shows errors around the weakly-textured areas, such as the surfaces of the chair and the table in **PlaytableP** or the white wall in **Vintage**, the proposed method shows more reliable results.

and it can be jointly trained with the cost function itself. Furthermore, the information exchange among pixels is done in feature space which contains richer contextual information than the final cost volume space.

Note that the improvement over the baseline clearly results neither from the use of extra layers nor from the use of more parameters, as the authors of [13] already have shown that the additional use of fully-connected (FC) layers is less significant. Using two additional FC layers leads to an improvement of approximately 1.90%, whereas using the **4P** module results in a 21.42% improvement in terms of average error.

The main contribution of the proposed method lies in learning a less ambiguous matching cost function by inspecting a larger area. Figure 4 shows that the proposed network actually works better around the weakly-textured area than the '**MC-CNN-acrt**' [13]. The quantitative and qualitative results of each dataset, including the ones in the 'test dense' set, are available at the Middlebury benchmark [1] website.

VI. Conclusions

Viewing a large area to estimate the dense pixel correspondence is necessary to fully utilize the texture information to achieve less ambiguous and more accurate matching. A conventional matching cost function fails because neighboring pixels on the same surface as the target pixel are unknown. In this paper, a novel CNN module is proposed to make the CNN structure handle a large image patch without losing the small details, which enables it to learn an intelligent matching cost function for large-sized windows. The learned cost function can discriminate the false matches for weakly-textured areas or repeating textures, and can also conserve the disparity discontinuities well/. The learned cost function achieves competitive performance on the Middlebury benchmark.

REFERENCES

[1] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *IJCV*, vol. 47, no. 1-3, pp. 7–42, 2002.

[2] V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions using graph cuts," in *ICCV*, vol. 2. IEEE, 2001, pp. 508–515.

[3] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information," *PAMI*, vol. 30, no. 2, pp. 328–341, 2008.

[4] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon, "Global stereo reconstruction under second-order smoothness priors," *PAMI*, vol. 31, no. 12, pp. 2115–2128, 2009.

[5] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," in *CVPR*. IEEE, 2011, pp. 3017–3024.

[6] Q. Yang, "A non-local cost aggregation method for stereo matching," in *CVPR*. IEEE, 2012, pp. 1402–1409.

[7] S. Birchfield and C. Tomasi, "Depth discontinuities by pixel-to-pixel stereo," *International Journal of Computer Vision*, vol. 35, no. 3, pp. 269–293, 1999.

[8] H. Hirschmuller and D. Scharstein, "Evaluation of stereo matching costs on images with radiometric differences," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 9, pp. 1582–1599, 2009.

[9] H. Hirschmüller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *CVPR*. IEEE, 2007, pp. 1–8.

[10] K. Wang, "Adaptive stereo matching algorithm based on edge detection," in *ICIP*, vol. 2. IEEE, 2004, pp. 1345–1348.

[11] K.-J. Yoon and I. S. Kweon, "Adaptive support-weight approach for correspondence search," *PAMI*, vol. 28, no. 4, pp. 650–656, 2006.

[12] F. Tombari, S. Mattoccia, L. D. Stefano, and E. Addimanda, "Classification and evaluation of cost aggregation methods for stereo correspondence," in *CVPR*. IEEE, 2008, pp. 1–8.

[13] J. Žbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2287–2318, 2016.

[14] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *CVPR*, June 2015, pp. 4353–4361.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[17] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[18] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Object detectors emerge in deep scene cnns," *arXiv preprint arXiv:1412.6856*, 2014.

[19] 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 *Pattern Recognition*. Springer, 2014, pp. 31–42.

[20] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *CVPR*, 2012.

[21] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *CVPR*, 2015.

[22] L. Ladickỳ, C. Häne, and M. Pollefeys, "Learning the matching function," *arXiv preprint arXiv:1502.00652*, 2015.

[23] K. Zhang, J. Lu, and G. Lafruit, "Cross-based local stereo matching using orthogonal integral images," *Circuits and Systems for Video Technology*, vol. 19, no. 7, pp. 1073–1079, 2009.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *ECCV*. Springer, 2014, pp. 346–361.

[25] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015, pp. 3431–3440.

[26] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *CVPR*, June 2015, pp. 447–456.

[27] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," *arXiv preprint arXiv:1505.04366*, 2015.

[28] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," *arXiv preprint arXiv:1504.06852*, 2015.