Article

Efficient Depth Map Creation with a Lightweight Deep Neural Network

Join Kang ¹ and Seong-Won Lee ^{2,*}¹ Department of Applied Artificial Intelligence, Kwangwoon University, Seoul 01897, Korea; kjikji956@kw.ac.kr² School of Computer and Information Engineering, Kwangwoon University, Seoul 01897, Korea

* Correspondence: swlee@kw.ac.kr; Tel.: +82-940-5471

Abstract: Finding depth information with stereo matching using a deep learning algorithm for embedded systems has recently gained significant attention owing to emerging high-performance mobile graphics processing units (GPUs). Several researchers have proposed feasible small-scale CNNs that can run on a local GPU, but they still suffer from low accuracy and/or high computational requirements. In the method proposed in this study, pooling layers with padding and an asymmetric convolution filter are used to reduce computational costs and simultaneously maintain the accuracy of disparity. The patch size and number of layers are adjusted by analyzing the feature and activation maps. The proposed method forms a small-scale network algorithm suitable for a vision system at the edge and still exhibits high-disparity accuracy and low computational loads as compared to existing stereo-matching networks.

Keywords: AI; embedded computer vision; neural networks; algorithm; stereo matching; feature visualization

Citation: Kang, J.; Lee, S.-W. Efficient Depth Map Creation with a Lightweight Deep Neural Network. *Electronics* **2021**, *10*, 479. <https://doi.org/10.3390/electronics10040479>

Academic Editor: Athanasios Voulodimos

Received: 24 December 2020

Accepted: 11 February 2021

Published: 18 February 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Artificial intelligence (AI) technologies are designed to collect data with useful features by analyzing data containing a wide range of information and solving problems using correlation among the features. Such methods have mainly been performed on cloud computing resources owing to extensive computational complexity while processing data. Meanwhile, AI applications for edge devices have recently gained significant attention owing to the feasibility of low-cost computation with the advancement of high-performance processors, such as graphics processing units (GPUs). Among them, AI-based embedded computer vision requiring real-time local signal processing is located at the core of next-generation technologies such as autonomous driving. In particular, various studies are being conducted on the implementation of AI-based stereo matching [1], which identifies the distances to the surrounding objects at relatively low system cost [2,3]. The three categories for stereo matching are global, local, and semiglobal matching, in which disparity maps are created by determining the corresponding matching points between two images obtained using a stereo camera [4]. The detection of matching points and generation of a disparity map becomes more accurate with the advancement of deep learning technologies.

The networks are constructed based on the recently proposed convolutional neural network (CNN) algorithm, which produces adequate results owing to its high accuracy [5]; however, such networks often do not fit into an embedded edge system because the deep learning network structure requires high computational resources for data analysis and feature extraction. Consequently, the slow computing speed owing to the lack of computing resources in the edge system makes the real-time operation of stereo matching impractical. Several researchers have proposed a feasible CNN-based small-scale network for a local GPU, which extracts features from both the left and right image patches [6,7]. They used a small Siamese network followed by an inner-product layer to yield results using relatively small networks that are comparable to those obtained from large networks.

However, Zbontar's MC-CNN-fast (Multi-Contrast CNN fast) [6] uses small (3×3) patches to determine the disparity, which leads to less accurate results and increased total execution time. Patrick et al.'s method [8] improves the accuracy of disparity at the expense of additional computational resources. Luo et al. [4] employed convolution networks without padding to reduce the computational cost; consequently, there exists a recurring distortion of the extracted features because the data in the central region of the patch are heavily overweighted.

The method proposed in this study uses padding and pooling to reduce data while preventing the distortion of features and an asymmetric convolution filter to reduce computational costs and simultaneously maintain the accuracy of disparity. In addition, by adjusting the patch size and number of layers through an analysis of the feature and activation maps, a small-scale network algorithm suitable for an embedded vision network, which detects objects more accurately using a small number of parameters, is proposed.

The remainder of this paper is organized as follows. In Section 2, previously designed stereo-matching networks are introduced. In Section 3, the proposed network structure and feature analysis are described for the proposed algorithm. In Section 4, the proposed algorithm is compared with existing algorithms. Finally, Section 5 presents the conclusions.

2. Related Work

The network that obtains the disparity map analyzes the similarity scores of the left and right images. Reducing the sizes of the images used as input data in the network can also reduce the number of layers required in the entire network. However, reducing the input size modifies the image data and causes feature distortion, which increases the errors in the disparity map. Consequently, we use patches cut into suitable sizes to determine the association of the features. Stereovision matching using CNN-based structural networks allows for the adjustment of input data with image patches [8] and performs matching process operations [4] using small left and right image patches to minimize the size of the input data to the extent possible.

The fully connected layer (i.e., the dense layer) serves to classify features extracted from the end of a CNN-based network. However, these layers use relatively more parameters than are required in the computation of the convolution layer and possibly cause a rearrangement of the disparity information present in the feature map. Therefore, recent studies use a small network algorithm in the form of a Siamese network and the dot-product layer at the end of the two networks to obtain matching scores according to the cosine similarity and extract the disparity map [4,6,7].

MC-CNN-fast [6] reduces the left and right input image patches to 9×9 and configures the Siamese network using 3×3 filters. They have the advantage of faster training than conventional networks, with a small number of parameters and shorter operations. However, small input patches can easily miss the significant boundary features of the entire input images in the process of extracting features regardless of the structure of the network [9]. Therefore, MC-CNN-fast may not be able to maintain sufficient accuracy of the disparity map.

Luo et al. [4] also uses the form of a small Siamese network to extract a $(1 \times 1 \times 64)$ -dimensional feature map from the left network and a $(|y_i| \times 64)$ -dimensional feature map from the right network. Therefore, Luo et al.'s network obtains similarity scores by calculating the dot product for both the feature maps and calculating SoftMax for image pixels to obtain a disparity map. Because of the small size of the network with small input patches and the simplicity of the cosine similarity, Luo et al.'s network is suitable for embedded systems. Because the network repeatedly uses convolution layers without padding to extract feature maps, the features of the edges converge to the center of the feature map as the operation progresses. Although overlapping the convolution layer without padding reduces computation and retains the details by not using pooling layers, it also causes geometric distortion for the edge of the feature map, which may increase disparity errors. In addition, without padding, the size of the feature map decreases slowly through the network layers, and the total number of computations is relatively large.

Patrick et al. [8] used pooling and deconvolution layers to minimize feature distortion while maintaining the accuracy of the disparity map. By adopting pooling layers, Patrick et al.'s network can utilize wide receptive fields on the feature maps to improve the matching performance. The deconvolution layer used in the last stage of the network to calculate the correlation increases the accuracy of disparity. However, with the addition of the deconvolution layer, the computations increase, and the total runtime and network size increase compared to Luo et al.'s network.

AnyNet [10] is an end-to-end architecture based on U-net [11] that extracts the disparity map by concatenating the feature map of the early training to the feature later in the training. AnyNet extracts the disparity map relatively quickly with networks using fewer parameters. For the other networks mentioned above, we find the convergence point of stereo images using Siamese structure, while AnyNet extracts features using the characteristics of U-net. It tends to be relatively less accurate to extract features due to fewer filters and fewer parameters than other networks, while its complex structures and data processing take more time.

Asymmetric convolution performs tasks with fewer parameters than conventional square kernel convolutional layers. Furthermore, training with horizontal and vertical kernels can prevent loss of unique features in deep networks [12]. For Inception-v3 [13], the convolution of (7×7) is decomposed into asymmetric filters, such as (1×7) , which can reasonably train a parameter-reduced network.

3. Proposed Method

The proposed method uses the inner product with the Siamese network, similar to a structure that performs well in small systems, such as Luo et al.'s study, to find and calculate the convergence points of the left and right images and to extract the disparity map. However, to address the problem of distortion of the peripherals, we used the padding and pooling layers to prevent overlapping of the features to the center, and we also proposed a network that uses an asymmetric filter to further reduce the amount of calculation while increasing the accuracy of disparity using horizontal information.

Small Network Algorithm

Figure 1 presents the network architectures proposed in this paper. First, we adjust the input data according to the size of the left and right patches. The left side is equal in width and length, but the right side is designed to extend the street to facilitate finding matching points. The initial padding layer creates the feature map size $(39, 37, 3)$, extracts the shape using a $(9, 9)$ filter size of the convolution layer with the same size padding, and subsamples it in the pooling window size $(1, 5)$. Thereafter, the convolution layer is placed using asymmetrical filters of size $(1, 7)$ and pooling layers of size $(1, 5)$. With asymmetrical filters, the total number of parameters can be reduced and features that contain horizontal information can be extracted.

In this work, we perform supervised learning based on CNN architecture. CNN uses filters to extract characteristics for images that came into the input for each layer in the process of training in a way that is mainly used for sequential data such as images and image data. Using 2D or higher filters allows training without losing the special characteristics, so that the special characteristics of data such as the disparity map are suitable for important processes. Therefore, in this paper, we use a CNN-based network and use a method to extract cosine similarity by training networks of the same structure to analyze the differences in convergence points using left and right images.

Figure 1. Network structure for extracting similarity score: (a) network structure composed of padding, pooling, and convolution layers; (b) size of each layer (height, width, channel).

Stereoscopic images are obtained simultaneously from two cameras placed horizontally at fixed intervals. It uses the horizontal distance between a specific point of the right image to a matching point of the left image to determine the depth. Repeating the search for the matching point along the image horizontally includes lots of redundant computation due to the overlying pixels in the patches to search. Therefore, the proposed method extends the right patch horizontally and allows it to be computed only once.

In stereo-matching-based methods, it is important to find the location of horizontal convergence points in the left and right images. We apply asymmetric filters to use only significant vertical features and to preserve most horizontal information of images, so that they can maintain the horizontal accuracy of the disparity map.

We use padding layers to allow feature maps to be extracted in an appropriate form to calculate the matching points of the input image data. Feature maps extracted from the convolution layer maintain information about the original features through down sampling using the average pooling layer and by expanding the receptive fields. The right image patch contains information about horizontal functions. Therefore, using the cost aggregation calculated from the inner product layer helps extract the similarity more clearly while calculating matching scores.

Figure 2 presents a visualization of the feature map extracted by the proposed method. In this study, a filter of size (9×9) was used to calculate the convolution later; the figure contains a part of the feature map of the network.

Figure 2. Visualization of the feature map extracted by the proposed method.

The network generates as many filters as the number of channels, which are the hyperparameters specified by the user. It uses a filter to extract the feature map into the

next layer. When we used small filters such as (3×3) and extracted feature maps from patches measuring (37×37) based on patches on the left side of the Siamese network, they were insufficient to extract the features that each filter needed from the network. Employing a network that computes the similarity score rather than small-sized filters to extract features for the local area, and using relatively large-sized filters (9×9) and (5×5) , it is possible to increase the similarity score using a large number of pixels to recognize objects. In addition, subsampling through the pooling layer in the middle of the solution layer prevents data from overlapping.

Figure 3 presents the visualized data of the feature map of both networks for calculating the similarity score. We adjust the image data in the input to the left and right image patch sizes. The use of asymmetric filters of sizes (1×5) and (1×7) in the network increases the effect of the reactive fields, and if the width is greater than the height, additional features can be extracted, as compared with the use of symmetric filters [14]. After the network operation, through the dot product layer, we calculate the dot product of the $(1 \times 201 \times 64)$ -dimensional feature map of the right network using the right image patch and the $(1 \times 1 \times 64)$ -dimensional feature map with the small left image patch input of the network on the left. Because the $(1 \times 201 \times 64)$ -dimensional feature map of the right image patch network contains the horizontal information about the features, it is possible to extract a better feature map using the asymmetric filter than when using the symmetric filter. Finally, after using $(5, 5)$ solution layers, the matching point is determined through the dot product operation and the disparity map is extracted [4,15].

Figure 3. Visualized data of the feature map of both networks for calculating similarity score.

The small network architecture proposed in this paper facilitates training using a pooling layer and asymmetric filters around the structure of the stereo pair and image patches of different sizes. Each of the patches acting on the input images has the same height. However, the small patches on the left are extracted in a square shape based on the center of the input image to minimize the amount of computation, whereas the patches on the right are extracted in a rectangular shape with a long width relative to the center to ensure that the matching points can be identified. The network is a structure in which a single network forms a stereo pair by sharing parameters with each other. It has a total of six convolution layers, three pooling layers, and added padding layers to prevent the distortion of the features.

4. Experiment

The performance of the proposed network was measured with a 3.5 GHz AMD Ryzen Threadripper 2950X 16-Core Processor computer. We used 4 NVIDIA® GeForce® RTX 2080 Ti GPUs to obtain comparison results with other high computational studies using high performance GPUs. We implemented Python TensorFlow in Ubuntu 18.04 OS and used

CUDA version 10.1 for GPU computation. The size of the input images is 1242×375 RGB. The runtime measures the time at which the input image enters the network and the disparity output. The runtime is measured using the Python method while the experiment was conducted.

Table 1 presents the results of N-pixel errors extracted through the KITTI 2015 validation set for the several Siamese architectures. The number in parentheses indicates the size of the input patch.

Table 1. Comparison of N-pixel errors, total parameters, and runtime(s) on the KITTI 2015 validation set.

		Proposed	MC-CNN-Fast [6]	Patrick's [8]	Luo's (37) [4]	Luo's (29)	Luo's (19)	AnyNet [10]
Network Parameters		382,016	263,104	436,288	694,112	562,432	297,216	43,269
Pixel Error (%)	3	6.44	14.40	8.14	7.07	7.15	10.01	9.01
	4	5.12	13.59	7.37	5.94	6.01	9.35	8.35
	5	4.44	12.72	6.23	5.29	5.34	8.56	7.57
Runtime(s)		0.11	0.13	0.15	0.17	0.16	0.14	0.20

N-pixel error is a calculation of the proportion of bad pixels averaged over ground-truth pixels, which we consider to be correctly estimated if the discrepancy or flow endpoint error is less than N-pixel. We estimate the error rate if we deviate from the N-pixel anomaly.

By comparing Luo et al.'s efficient deep learning for stereo matching with the networks proposed in this paper, we can confirm that our three-pixel error decreased by 0.8%, although the size of the image patch was reduced by half from Luo et al.'s (37) [4] parameters based on the experimental results of the network.

Luo et al. use a network of modified forms according to the size of the input patch. The numbers in parentheses indicate the size of the patch that Luo et al. use as input and constructs the network using (3×3) and (5×5) sizes. We implement comparison experiments with several networks in Luo et al. Even with a difference of approximately 100,000 parameters from Luo et al.'s (19) [4] study, evidently our N-pixel error is approximately 4%.

For MC-CNN-fast, the number of parameters is clearly small, but the N-pixel error is 7.96% different from ours when (3×3) filters are used. The networks used by Patrick et al. use multiple (3×3) convolutions. To compare with this task, we proceed with the experiment after some adjustments to the redundant convolutional layers. Patrick's method has 0.8% more parameters and 1.7% higher three-pixel error compared to the proposed method.

AnyNet has fewer parameters compared to other schemes. However, the process of extracting and concatenating features, the process of the sub-network using the synthesized feature map, and the process of extracting fine features are repeated to extract results. Thus, the computational latency increases, which mainly involves matrix operations, then the network is found to be slow compared to other networks with simple structures.

The proposed method gives the most accurate disparity results while having the fastest runtime compared to other networks, as shown in Table 1. For comparison to Luo et al.'s networks, which have a structure similar to the one proposed and show poorer accuracy with fewer parameter for fast runtime, the proposed method has $|0.11 - 0.14| / 0.14 = 21.4\%$ faster runtime and $|6.44 - 10.01| / 10.01 = 35.6\%$ better accuracy compared to the fastest Luo et al.'s network (19×19) . On the other hand, the proposed method has $|0.11 - 0.17| / 0.17 = 35.2\%$ faster runtime while showing $|6.44 - 7.07| / 7.07 = 8.9\%$ better accuracy compared to the most accurate Luo et al.'s network (37×37) . The proposed method also has more than $|0.11 - 0.13| / 0.13 = 15.3\%$ faster runtime compared to the other networks with different structures, while providing more than $|6.44 - 8.14| / 8.14 = 20.8\%$ accurate disparity results as three-pixel errors, as shown in Table 1. The proposed method removes nonsignificant operations of the network and turns out the reduced total runtime while maintaining the pixel errors by adopting asymmetric filters and subsampling layers.

The runtime results in the table indicate that the training operations are fast owing to the nature of the small networks through the runtime section. However, the runtime of the

proposed network is found to be smaller than that in all other networks. In the case of the method presented in this paper, the pooling layer reduces the height of the entire image by one-third, and the time required to extract features is relatively less as compared with other current methods because the convolution operation in the solution layer becomes significantly smaller despite the relatively large filter size.

Figure 4 presents a photograph of the testing disparity map using the training network from the KITTI 2015 dataset. By comparing the map obtained with the disparity map that detected cars on the road extracted by other networks, our disparity map evidently had a clearer edge and segmentation as compared with the map extracted using Luo et al.'s network, and the noise was relatively low. Noticeably, information on objects far from the core is output relatively accurately.

Figure 4. KITTI 2015 test set: (left) original images, (center) depth map images obtained using proposed algorithm, (right) depth map images obtained using Luo et al.'s algorithm.

Figure 5 shows the original images and depth maps from the KITTY 2015 test set, which are extracted by the proposed method and Luo's (37×37) method. For the comparison purpose, we draw boxes where noticeable differences exist. Compared to the proposed method, there are more noises at the boundary of objects in the result images generated by Luo's network (37×37).

Figure 5. Noise comparison in the disparity map: (left) original images, (center) depth map images obtained using the proposed algorithm, (right) depth map images obtained using Luo et al.'s algorithm with (37×37) configuration.

5. Conclusions

Edge computing with artificial intelligence technology makes embedded vision systems of high-speed depth creation using an advanced mobile GPU feasible. The stereo-matching method using lightweight deep learning, as proposed in this paper, attempts to reduce the distortion at the edge of the image by using padding and down sampling, which are basically used in image processing and exploits implicit depth information from the image features. In addition, fewer parameters used in the network training were successfully performed as a parallel operation based on a GPU. As a simple result of the experiment part, the proposed technique facilitates good disparity matching with $|6.44 - 7.07| / 7.07 = 8.9\%$ lower three-pixel error while spending $|0.11 - 0.17| / 0.17 = 35.2\%$ less runtime, compared to the Luo's work with (37×37) input sizes. Compared to MC-CNN-fast, the proposed technique reduces three-pixel error by $|6.44 - 14.40| / 14.40 = 55.2\%$ with $|0.11 - 0.13| / 0.13 = 15.3\%$ less runtime. We also reduce three-pixel error by $|6.44 - 9.01| / 9.01 = 28.5\%$ and runtime by $|0.11 - 0.20| / 0.20 = 45\%$ compared to AnyNet. In future work, we plan to conduct research to further improve the matching performance by including the motion information obtained during real-time operations in the computation. We also plan to further study how matrix operations of the network can be optimized on mobile GPUs to further reduce the runtime and memory of the proposed method.

Author Contributions: Conceptualization, J.K. and S.-W.L.; methodology, J.K. and S.-W.L.; software, J.K.; validation, J.K.; formal analysis, J.K. and S.-W.L.; investigation, J.K. and S.-W.L.; resources, J.K. and S.-W.L.; data curation, J.K. and S.-W.L.; writing—original draft preparation, J.K. and S.-W.L.; writing—review and editing, S.-W.L.; visualization, J.K.; supervision, S.-W.L.; project administration, S.-W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by the MOTIE and KSRC support program for the development of the future semiconductor device (10080649), by the Global Human Resources Program in Energy Technology of the KETEP granted financial resource from the MOTIE, Republic of Korea (2019401000050), and by the Research Grant of Kwangwoon University in 2021.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Song, T.; Kim, Y.; Oh, C.; Sohn, K. Deep Network for Simultaneous Stereo Matching and Dehazing. In Proceedings of the British Machine Vision Conference (BMVC), Newcastle, UK, 3–6 September 2018.
2. Liu, J.; Ji, S.; Zhang, C.; Qin, Z. Evaluation of deep learning based stereo matching methods: From ground to aerial images. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2018**, *42*, 593–597. [[CrossRef](#)]
3. Hamzah, R.A.; Hamid, A.M.A.; Salim, S.I.M. The Solution of Stereo Correspondence Problem Using Block Matching Algorithm in Stereo Vision Mobile Robot. In Proceedings of the 2010 Second International Conference on Computer Research and Development, Kuala Lumpur, Malaysia, 7–10 May 2010; pp. 733–737.
4. Luo, W.; Schwing, A.G.; Urtasun, R. Efficient deep learning for stereo matching. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 5695–5703.
5. Xu, B.; Zhao, S.; Sui, X.; Hua, C. High-speed Stereo Matching Algorithm for Ultra-high Resolution Binocular Image. In Proceedings of the 2018 IEEE International Conference on Automation, Electronics and Electrical Engineering (AUTEEE), Shenyang, China, 16–18 November 2018; pp. 87–90.
6. Zbontar, J.; LeCun, Y. Stereo matching by training a convolutional neural network to compare image patches. *J. Mach. Learn. Res.* **2016**, *17*, 1–32.
7. Zbontar, J.; LeCun, Y. Computing the Stereo Matching Cost with a Convolutional Neural Network. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1592–1599.
8. Patrick, B.; Mazomenos, E.; Stoyanov, D. Widening Siamese architectures for stereo matching. *Pattern Recognit. Lett.* **2019**, *120*, 75–81.
9. Hamwood, J.; Alonso-Caneiro, D.; Read, S.A.; Vincent, S.J.; Collins, M.J. Effect of patch size and network architecture on a convolutional neural network approach for automatic segmentation of OCT retinal layers. *Biomed. Opt. Express* **2018**, *9*, 3049–3066. [[CrossRef](#)] [[PubMed](#)]

10. Wang, Y.; Lai, Z.; Huang, G.; Wang, B.; Maaten, L.; Campbel, M.; Weinberger, K. Anytime Stereo Image Depth Estimation on Mobile Devices. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019.
11. Ronneberger, O.; Fischer, P.; Brox, T. U-net: Convolutional networks for biomedical image segmentation. In Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention, Munich, Germany, 5–9 October 2015; pp. 234–241.
12. Ding, X.; Guo, Y.; Ding, G.; Han, J. ACNet: Strengthening the Kernel Skeletons for Powerful CNN via Asymmetric Convolution Blocks. In Proceedings of the International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019.
13. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, J. Rethinking the inception architecture for computer vision. In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016.
14. Wang, S.; Dong, J. Asymmetric filtering-based dense convolutional neural network for person re-identification combined with Joint Bayesian and re-ranking. *J. Vis. Commun. Image Represent.* **2018**, *57*, 262–271. [[CrossRef](#)]
15. Chen, Z.; Sun, X.; Wang, L.; Yu, Y.; Huang, C. A Deep Visual Correspondence Embedding Model for Stereo Matching Costs. In Proceedings of the 2015 IEEE International Conference on Computer Vision ICCV, Santiago, Chile, 7–13 December 2015; pp. 972–980.