Regular Article

Multitasking Correlation Network for Depth Information Reconstruction

Van Quang Nguyen[1], **Cao Duy Hoang**[2], **Hong Phuc Nguyen**[3]

[1] University of Science and Technology, The University of Da Nang, Da Nang, Vietnam
[2] Vietnamese-German University, Thu Dau Mot City, Vietnam
[3] Eastern International University, Thu Dau Mot City, Vietnam

Correspondence: Hong Phuc Nguyen, phuc.nguyenhong@eiu.edu.vn

Communication: received 29 August 2021, revised 1 November 2022, accepted 13 February 2023
Online publication: 4 April 2023, Digital Object Identifier: 10.21553/rev-jec.293
The associate editor coordinating the review of this article and recommending it for publication was Prof. Vo Nguyen Quoc Bao.

Abstract– **In this paper, we propose a novel multi-tasking network for stereo matching. The proposed network is trained to approximate similarity functions in statistics and linear algebra such as correlation coefficient, distance correlation and cosine similarity. By doing this, the proposed method decreases the amount of time needed to calculate the disparity map by using CNN's ability to calculate multiple pairs of image patches at the same time. We then compare the execution time and overall accuracy between the traditional method using functions and our method. The results show the model's ability to mimic the traditional method's performance while taking considerably less time to perform the task.**

Keywords– **Machine learning, Deep Learning, Stereo Matching.**

1 Introduction

Reconstructing the scene in 3D is key in many applications, such as robotics and self-driving cars. In order to create a depth perception image for the systems, sensors such as Lidars are used to gauge the depth of an object using pulsed lasers. This solution, however, is not cost-effective and does not perform well in harsh conditions. Utilizing cameras to produce depth perception through stereo vision is an attractive solution, as it is typically way more cost-efficient. But, despite decades of research, estimating depth from stereo matching algorithms is still an open problem, and improving its effectiveness is a challenge that needs to be solved in the following years, due to the increasing needs for 3D visualization in the industry. One of those challenges is decreasing matching cost calculation time. Typically, a matching cost is computed at each pair of pixels from left and right images in a certain disparity range, which could cost a tremendous amount of time for high-resolution images. One way to circumvent this problem is to apply Deep learning (specifically - CNN) to simulate the computational process. Since its introduction, deep learning techniques have solved many problems with great efficiency thanks to its ability to simulate statistical or approximate problems. In recent years, the concept of deep learning being used to solve mathematical problems like function integration, or solving differential equations has been shown to be possible and more effective than other mathematical frameworks such as Matlab [1]. With that idea in mind, we propose a method of imitating the calculation of the correlation matrix using Convolutional Neural Networks, and after that, use it to calculate the disparity value of pairs of image batches at a time.

This paper will be organized as follows: In "Related Work" section, we introduce the concept of Deep Learning and examples of its ability to simulate mathematical equations. We also talk about the functions we will simulate, which are cosine similarity, correlation coefficient, and distance correlation in the same section. In the "Methodology" section are the explanations of the CNN architectures that were used to simulate the equations, mainly, the single-task approach and the multitask approach. The process of training and testing both the single task and the multitask approach are noted in section 4 ("Experiments results"). And finally, we explain how we can use the nature of CNNs to calculate multiple pairs of image patches at the same time, which will cut the time to fractions of a second, and show the disparity mapping results along with the time duration between using the traditional function and using our CNN approach. For convenience's sake and for ease of comparison, post-processing methods were not used and the model's disparity maps were directly compared against each other.

2 Related Works

2.1 Deep Learning

Deep learning is a part of a broader family of machine learning methods based on artificial neural networks with representation learning. The concept of using nodes in a neural network similar to communication in biological systems have been studied as early as 1873 [2] from basic forms of neural nets such as "Perceptrons" to the Deep learning methods as we know now. From computer vision's applications such as Image Classification and Object defection using CNNs [3] to Recurrent Neural Networks being used in Natural Language Processing task [4], deep learning has proved its ability to solve a plethora of practical problems in different fields.

Recently, Researchers at Facebook conducted a research to determine Deep learning's ability to solve more elaborate mathematical problems, such as symbolic integration and solving differential equations [1]. This research paper showed the ability of a deep learning method to solve mathematical problems with increase efficiency, which then inspires our proposed method's idea - using Deep learning technique (specifically Convoluitonal Neural Network) to imitate the functions we are going to introduce below.

2.2 Stereo Matching

Stereo correspondence is a fundamental problem of pairing each of pixels in a left and right picture pair together, and it is crucial for computer vision to recover depth math of a scene from left and right stereo images [5, 6]. Various stereo correspondence algorithms have been used in various fields such as medical, satellite-based earth observation, space exploration, autonomous robots, and security systems. However, solving the stereo correspondence problem is still a challenging task in certain areas due to texture-less regions [7], occlusions, illumination variations, changes int the weather such as snow, sun, and rain [8], and cameras with different focal lengths [9].

2.3 Different Correlation Functions

We are interested in computing a disparity image given a stereo image pair. Throughout this paper we assume that the image pairs are rectified. In order to create a disparity map to solve stereo matching problem, we have to build a function to calculate the disparity value, which represents the deviation of each pixel in the right image correspond to the left image.

$$d_p = \arg\min_{d \in D}(C(p,q)), \tag{1}$$

$$\text{where } C(p,q) = (L(q) - R(p))^2, \tag{2}$$

$$\text{and } q = (x_p - d, y_p). \tag{3}$$

In the equations above, the equation (1) is used to find the disparity value by taking the index of the minimum value of the equation (2), which is the Euclidean distance between p^{th} pixel of the left image and q^{th} pixel of the right image. Thus, to find the correspond q pixel to p pixel, we have to make sure that the variables are aligned with (3), which is the p and q pixels are in the same row coordinates $(y_q = y_p)$ and the column coordinate is equal to $(x_p - d)$ where $d \in [0, D]$. After running the equation (2), we will have an array of disparity value with the length of the disparity range, which can be used to determine the space between the left pixel and the right pixel that is similar to the left the most. In this case, the more similar a pixel is, the lower the value.

Below we introduce our general matching cost function, which is used for calculating the disparity of the image pairs, which is used to create the data set for our propose method.

$$d_p = \arg\max_{d \in D}(C(p,q)),$$

$$\text{where } C(p,q) = f(W_p, W_q),$$

$$\text{and } q = (x_p - d, y_p).$$

From the set of equations above, we can see the similarity between it and the function we talked about previously. Similar to the equation (1) and (3) we showed, the general function we used to calculate disparity value start with the equation (1) that runs through all the images in the disparity range (3) to find the left and right image patches that has the most correlations to each other. The main difference in this case is out function finds the maximum value instead of the minimum value. Due to the correlation equations we used, the higher the value the more similar the image patches are. The second difference between the first three equations and our three function is that the equation to calculate Euclidean distance in (2) is replaced by a function $f(W_p, W_q)$, which represents the correlation functions that are used to measure similarity between image patches.

The first correlation function we used is cosine similarity, which is calculated by using the product of the two vectors divided by the product of each vector's length.

$$f(W_p, W_q) = \cos\varphi = \frac{W_p \cdot W_q}{|W_p| \cdot |W_q|}.$$

Cosine similarity determines the similarity by calculating the cosine of the angle between these two vectors, the more these two vectors are correlate, the smaller the angle, and in return, the higher the value (which peaks at one when the angle is 0°).

The second correlation function we used is correlation coefficient, which is calculated by the product of the two centered vectors divided by the product of each centered vector's length, which is similar to cosine similarity but invariant to shifting.

$$f(W_p, W_q) = r(W_p, W_q) = \frac{(W_p - \overline{W_p}) \cdot (W_q - \overline{W_q})}{|(W_p - \overline{W_p})| \cdot |(W_q - \overline{W_q})|}$$

Correlation coefficient, different from cosine similarity, measures the similarity between data set by calculating the ratio between the co-variance of two variables and the product of their standard deviations.

In statistics, both cosine similarity and correlation coefficient are used to determined the similarities of vectors, which is useful in many application, including finding similar image pairs to determined the disparity range. Both functions are powerful and being used in a lot of areas in deep learning. An example of which is their application in calculating similarity between word vectors in NLP [10].

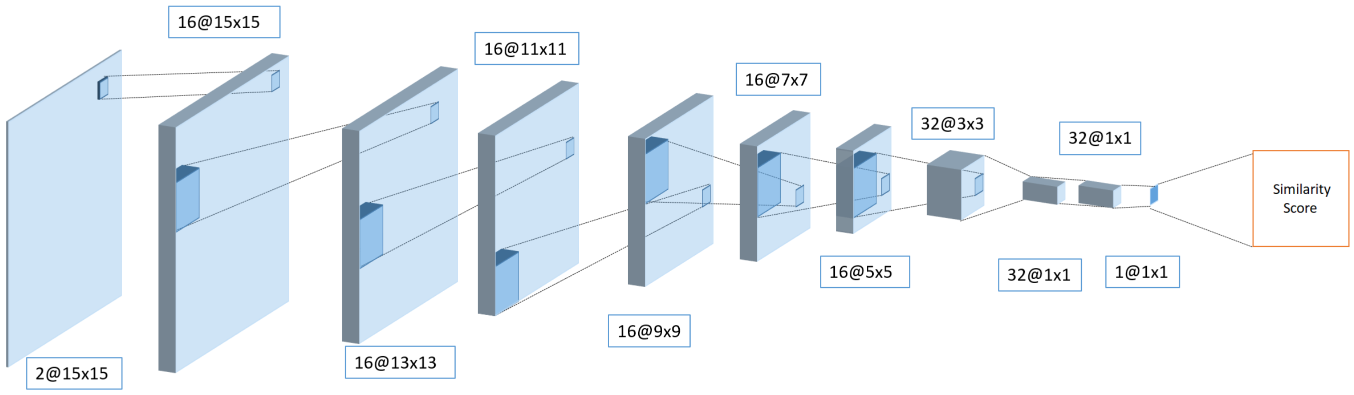


Figure 1. Illustration of the CNN's architecture in single-task.

The third and final correlation function we used is distance correlations, which is the co-variance of the double centered distances of two vectors divided by the dot product of the variance of each of the two double centered distances of vectors, or in other words, distance correlations of the two vectors divided by the dot product of the distance variance of the two vectors.

$$f(W_p, W_q) = dCor(W_p, W_q) = \frac{dCov(W_p, W_q)}{\sqrt{dVar(W_p)dVar(W_q)}}$$

Although cosine similarity and correlation coefficient are both effective in calculating the similarity between data, they cannot detect non-linear correlation between data-set. Distance correlation has demonstrated the ability to circumvent this problem, by being able to detect both linear and non-linear relations between data. This function has been found to have "higher statistical power", and be able to locate smaller sets of variables that provide equivalent statistical information [11].

Because of the effectiveness of these equations in calculating similarities between data in certain criteria, we chose these three algorithm to calculate the disparity map for our training set.

3 Methodology

For accurate matching of image pairs, we consider different texture and disparity information and develop a method of adaptively fusing the information needed for matching results of different matching windows by training a CNN model.

3.1 The Single-Task CNN Approach

To illustrate how CNNs work with stereo matching problems, we start with a single-task network. In this approach, we start off with a CNN with dynamic input size to imitate the functionality of the functions above. Assuming that the image pairs have been rectified and calibrated and the pair of images have the same size, we target the model to output a tensor with a dimension of $1 \times 1 \times 1$ ($H \times W \times D$), which corresponds to the output of cost function. In this case, we used the CNN network to take in a concatenation of the left and right image patches that has the same size, pass it through the network's convolution layers and normalize the output on top to predict the matching cost.

The purpose of the 1×1 size kernel at the first layer is to simulate the co-variance calculation between each pixels of the image patches that exist in cosine similarity function and other functions. After passing through the first layer, the CNN then down-sizes the output using 3×3 kernels. The architecture consists of 10 convolution layers From L1 to L10 with batch normalization in between. The first convolution layer consist of 16 kernels with an output size of $1 \times 1 \times 2$. From the L2 to L8, each layer has 16 kernels each with an output size of $3 \times 3 \times 16$, the last two layers will have a kernel size of 1×1, which then pass through a squeeze function in order to remove redundant dimension of the output (from $[1 \times 1 \times 1]$ to $[1 \times 1]$). The purpose of using batch normalization right after each layers is to scale the output of that layer, specifically, standardizing the activation of each input variable per mini-batch, such as the activation of a node from the previous layer from our own data, which will reduce the time of the training process and stabilizing our model. At the last layer, we use 1×1 kernel in the convolution layer since the disparity difference feature is one dimensional, we also compress the patch texture feature as a one-dimension feature in the last convolution layer. Recall that standardization refers to the re-scaling of data to have a mean of zero and a standard deviation of one. Our output is the disparity map with the matching cost computed throughout the layers. As a result we are able to calculate the correlations between the different disparities of each images by using CNNs to learn the functions.

3.2 The Multitask CNN Approach

After testing for a single task result using the architecture, we propose a multitask deep learning method for stereo matching problem, which uses the same architecture with a modified output layer to calculate multiple scores at the same times, hence multitask.

In multitask CNN approach, from L1 to L8, the architecture has the same layers as before. In the final layer, The multi-task learning classifier has the same architecture as a single-task classifier except that the

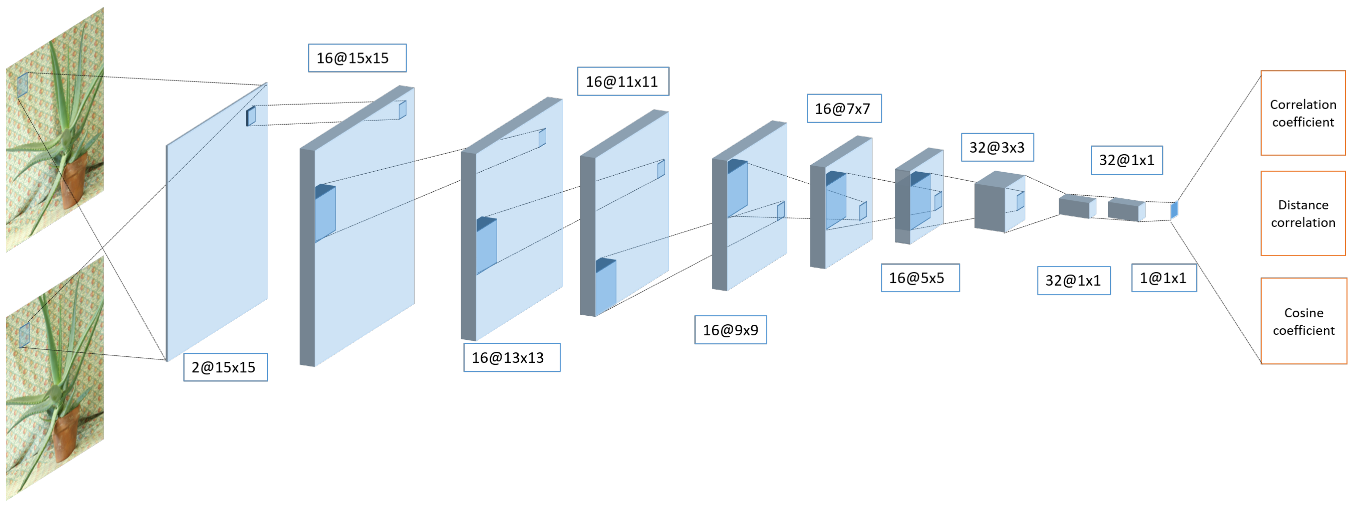


Figure 2. Illustration of the CNN's architecture in multi-task.

output layer is proportional to the number of tasks. This way, the information derived from the convolution layers can be used to calculate multiple results without needing to create a separate model.

A feature map created by a convolution layer can be represented with a three-dimensional tensor of order $D \times W \times H \times N$, where W and H are the width and height while N is the number of channels and D is the batch size. A pooling operation with a size of $K \times K$ and a certain number of stride applied to a tensor will results in another tensor. Using this property, by pairing a left image patch with multiple right image patches to create an input of $D \times W \times H \times 2$, where D is the number of pairs of image patches, and $W \times H \times 2$ represents the width, height, and the two image patches paired together, we are able to calculate the disparity map for multiple right images patches with one left patch, which reduces the time for the model to find the right disparity value.

4 Experiments Results

We assess our proposed method on the training data from Middlebury 2006 [12]. The Middlebury dataset focuses on the training and validation for stereo matching, with high resolution images and accurate disparity mapping. We implemented the networks using Keras with TensorFlow back-end. The experiments were conducted on Google Colab and used two CNNs models we mentioned above. First, we calculate the matching cost using out functions we mentioned at the beginning of our paper to have a base for evaluating the our CNN model along with the time for the functions to run in order to compare the run-time to out CNN approach. We did not apply any post-processing to the Disparity map for the ease of comparing the methods' results together.

The data we used were from the Stereo Data from Middlebury data set, with each image patch pairs having a size of 15×15. We used the traditional method to calculate the matching cost for each image pair, the

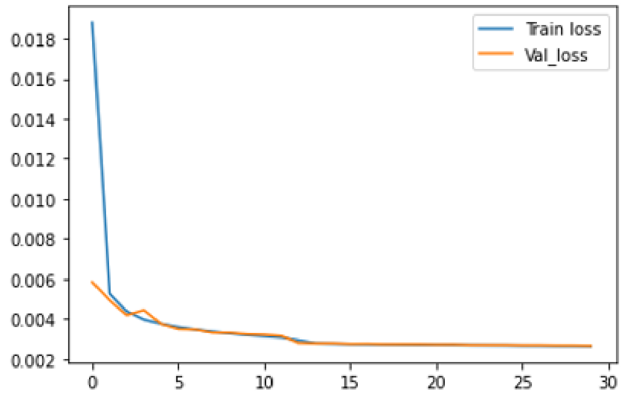


Figure 3. Single-task CNN's loss plot.

Table I
Disparity Mapping Run-Time Using Traditional Method

Correlation method	run-time (in seconds)
Cosine similarity	339.34827
Distance correlation	841.03754
Correlation Coefficient	1499.31873

outputs were then used along with the image pair patches to train the model to simulate the functions.

As the result in Table I shows, the time spent running these functions to calculate matching cost is extremely long, which shows us the the reason behind high demands for faster methods of calculating disparity values for stereo matching.

After training the model and successfully imitate the functions, we can start the multitasking process by creating matrix with shape = (disparity range, Height, Width, 2), with all the channel 0 being the pixels of left image patch and channel 1 being the pixels on the the right image patch. While creating the matrix, each of the channel 1 in the disparity range matrix will correspond to a right image patch in the disparity range. This way, each of the disparity range matrix will have a pair of left and right image patch, with each of

Table II
Disparity Mapping Run-Time Using Multitask Method

Correlation method	run-time (in seconds)
Cosing similarity	0.524169
Distance correlation	0.508038
Correlation Coefficient	0.505406

Table III
Average Percentage of the Error Pixels of the Disparity Map Output Compare to the Ground Truth

Image	CNN	Traditional (15×15)	Traditional (7×7)
Aloe	33.13%	28%	23.8%
Baby1	22.6%	86.9%	86.8%
Cloth1	7.6%	8.6%	90.43%
Wood1	30.6%	24.1%	21.7%
Cloth2	38%	85%	98.9%

the image patches being slightly adjusted to the right, that way we can simulate the process of calculating the disparity value in the disparity range. After creating the matrix and run it through the CNN, we will have an output matrix with shape = (disparity range, height - 14, width - 14, 1), with each height and width element having an array of disparity value similar to the disparity array when using the traditional method. We can then take the highest correlation value in each array and derive a disparity value/map that way.

$$d_p = \underset{d \in D}{\arg\max}(C(p,q))$$
$$\text{where } C(p,q) = f(W_p, W_q),$$
$$\text{and } q = (x_p - d, y_p).$$

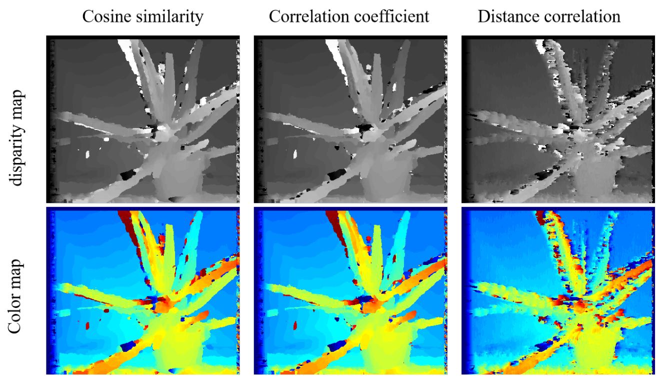
The function we simulate includes: Cosine similarity, correlation coefficient, and distance correlation.

We trained our network using stochastic gradient descent and back propagation with AdaGrad. Similar to moment-based stochastic gradient descent, AdaGrad adapts the gradient based on historical information [13]. Contrasting moment based methods it emphasizes rare but informative features. We used Mean Squared Error as our loss function.

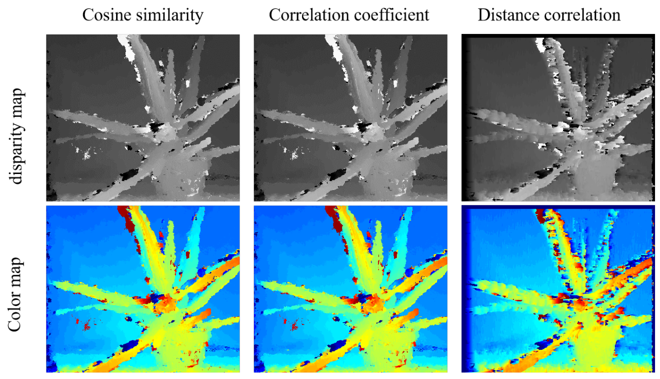
$$loss = \frac{1}{N}\sum_{i=1}^{N}(y_i - \overline{y}_i)^2.$$

This lost function gauge how close the model is to the output by calculating the difference between the prediction and ground truth squared. In our case we train our model using with 30 epochs and a learning rate of 1.0.

After training and testing with the single task approach, we achieved a fairly accurate imitation of the cosine similarity function with a run-time of only 0.77 seconds. We then trained and tested our multi-task approach and came up with similar results. The run-time of each of the task is in Table II.



(a) Traditional method



(b) Multitask method

Figure 4. Disparity maps with traditional and multitask method on Aloe image.

In order to figure out the output performance between the traditional method using function, we used these two methods, calculated the disparity map for multiple images, and use the output to calculate the average percentage of the error pixels between the output map and the ground truth. The error function is as follows:

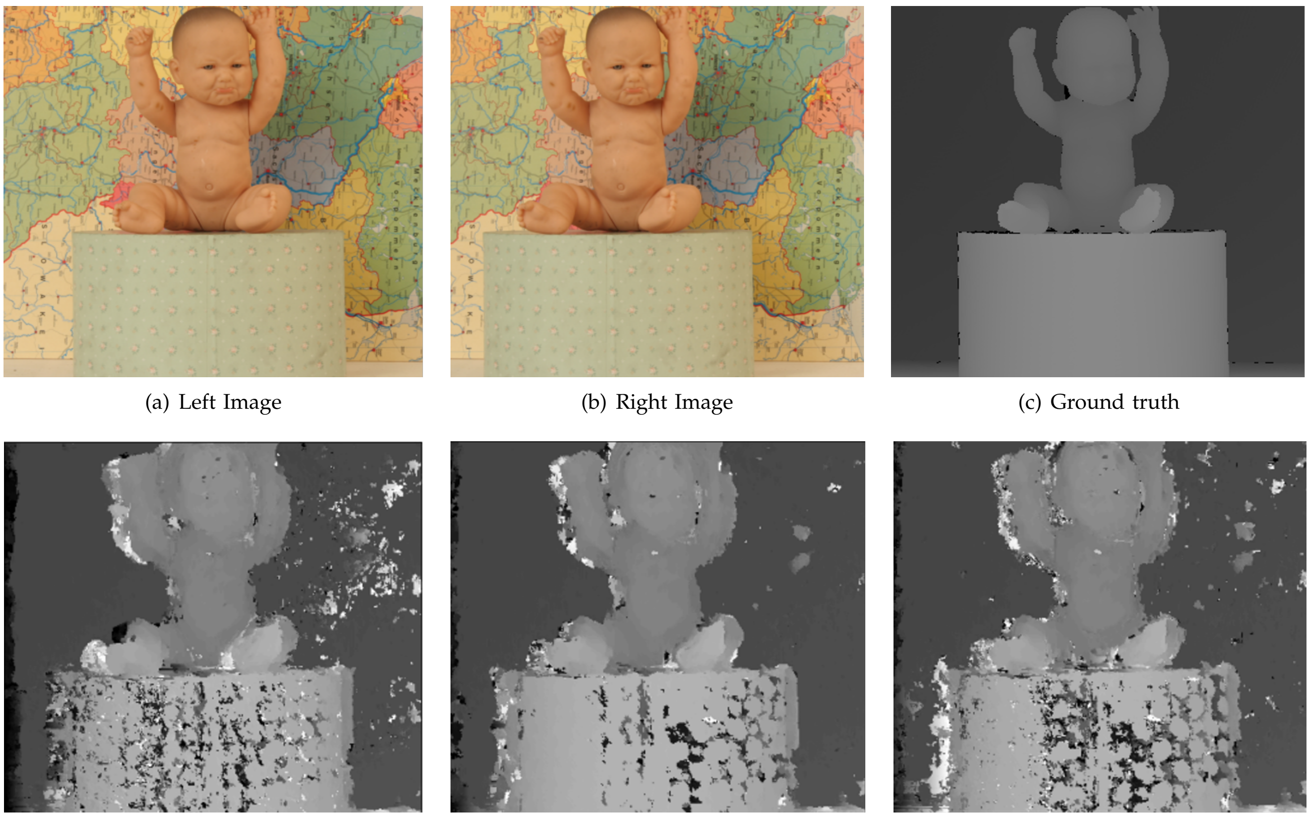
$$\epsilon(\%) = \frac{100}{\left|l_{pixels}\right|}\sum_{p \in l_{pixels}}\begin{cases}0, & \text{if } \left|\mathbf{D}_{Ypr}(p) - \mathbf{D}_{Ygr}(p)\right| < 1\\ 1, & \text{otherwise.}\end{cases}$$

where $\epsilon(\%)$ is the average percentage of the error pixels, l_{pixels} are the pixels in the disparity map, $|l_{pixels}|$ is the number of pixel in the map and $\mathbf{D}_{Ypr}(p)$ and $\mathbf{D}_{Ygr}(p)$ are the disparity values of both the prediction and the ground truth.

The image that were used to test are images from the Stereo Data of Middlebury data-set.

After calculating the average percentage of the error pixels, the result shows that our CNN approach not only cuts down the time it takes to produce a disparity map, but the disparity map results are on par with traditional methods, or in some case, produce even better results compare to traditional approach.

In Figure 4 are the examples of disparity mapping using traditional functions and using our multitask CNN method on Aloe image sample. Our method produced a fairly accurate representations of the functions' outputs with a fraction of the time. We also include an example of our method's disparity map compare to the ground truth in Figure 5.



(a) Left Image (b) Right Image (c) Ground truth

(d) Cosine similarity (e) Correlation coefficient (f) Distance correlation

Figure 5. Disparity maps with our method on Baby image with ground truth.

In our test, the single-task CNN and multitask CNN have about the same accuracy while the multitask method has the added benefit of calculating different cost functions simultaneously. This allows us to simulate more functions with the run-time reduced [14].

5 Conclusion

Using convolutional neural networks, our method was able to simulate all three of the correlation functions with high accuracy. And with some slight adjustments to our input, we were able to exploit the nature of CNNs to compute disparity maps similar to using correlation functions with a fraction of the time compared to using it. Therefore, our proposed method can be used in real-time systems such as self-driving cars and robotics with all the benefits of stereo matching while costing a fraction of the time.

References

[1] G. Lample and F. Charton, “Deep learning for symbolic mathematics,” *arXiv preprint arXiv:1912.01412*, 2019.
[2] H. Wang, B. Raj, and E. P. Xing, “On the origin of deep learning,” *arXiv preprint arXiv:1702.07800*, 2017.
[3] Z. Li, W. Yang, S. Peng, and F. Liu, “A survey of convolutional neural networks: Analysis, applications, and prospects,” *arXiv preprint arXiv:2004.02806*, 2020.
[4] W. Yin, K. Kann, M. Yu, and H. Schütze, “Comparative study of CNN and RNN for natural language processing,” *arXiv preprint arXiv:1702.01923*, 2017.
[5] V. Q. Dinh, V. D. Nguyen, H. V. Nguyen, and J. W. Jeon, “Fuzzy encoding pattern for stereo matching cost,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 7, pp. 1215–1228, Jul. 2016.
[6] V. Q. Dinh, V. D. Nguyen, and J. W. Jeon, “Robust matching cost function for stereo correspondence using matching by tone mapping and adaptive orthogonal integral image,” *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5416–5431, Dec. 2015.
[7] V. Q. Dinh, C. C. Pham, and J. W. Jeon, “Robust adaptive normalized cross-correlation for stereo matching cost computation,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 7, pp. 1421–1434, 2017.
[8] ——, “Matching cost function using robust soft rank transformations,” *IET Image Processing*, vol. 10, no. 7, pp. 561–569, 2016.
[9] V. Q. Dinh, F. Munir, A. M. Sheri, and M. Jeon, “Disparity estimation using stereo images with different focal lengths,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 12, pp. 5258–5270, 2020.
[10] V. Zhelezniak, A. Savkov, A. Shen, and N. Y. Hammerla, “Correlation coefficients and semantic textual similarity,” *arXiv preprint arXiv:1905.07790*, 2019.
[11] D. S. P. Richards, “Distance correlation: A new tool for detecting association and measuring correlation between data sets,” 2017.
[12] H. Hirschmuller and D. Scharstein, “Evaluation of cost functions for stereo matching,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp. 1–8.
[13] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.
[14] Y. Xie, Z. Xu, Z. Wang, and S. Ji, “Self-supervised learning of graph neural networks: A unified review,” *arXiv preprint arXiv:2102.10757*, 2021.

Van Quang Nguyen received an engineering degree from University of Science and Technology, The University of Da Nang, in 2022. He used to work as an AI researcher at AI VIETNAM. Currently, he is working as an AI engineer at a company in Vietnam. His research interests include optimization, machine learning, and computer vision.

Cao Duy Hoang received the B.S of Computer Science from Vietnamese-German University and Frankfurt University of Applied Sciences in 2022. Currently, he is working as an AI engineer at Quh Lab, as well as a Research Assistant at Vin University. His research interests include Machine Learning, Natural Language Processing, Robustness, and Deep Learning.

Hong Phuc Nguyen received the B.S of science service from Auckland University of Technology (AUT), New Zealand in 2013. She received Ph.D. degree in Electrical and Computer Engineering, Sungkyunkwan University in 2017. Currently, she is working as a postdoc at Information and Communication Engineering, GIST. Her research interests include optimization, machine learning, and computer vision.