Study of Convolutional Neural Networks applied to Image Stereo Matching

João Pedro Poloni Ponce, Ricardo Suyama
Universidade Federal do ABC
Santo André - SP, Brazil
Email: poloni.ponce@aluno.ufabc.edu.br, ricardo.suyama@ufabc.edu.br

***Abstract*— Stereo images are images formed from two or more sources that capture the same scene so that it is possible to infer the depth of the scene under analysis. The use of convolutional neural networks to compute these images has been shown to be a viable alternative due to its speed in finding the correspondence between the images. This raises questions related to the influence of structural parameters, such as size of *kernel*, *stride* and pooling policy on the performance of the neural network. To this end, this work sought to reproduce an article that deals with the topic and to explore the influence of the parameters mentioned above in function of the results of error rate and losses of the neural model. The results obtained reveal improvements. The influence of the parameters on the training time of the models was also notable, even using the GPU, the temporal difference in the training period between the maximum and minimum limits reached a ratio of six times.**

I. Introduction

In different computer vision applications, such as robot control and autonomous vehicles, it is required that the system be able to extract depth information from the acquired images, often provided by a binocular imaging system. In this scenario, depicted in Figure I, a single scene is captured from two different angles and the pixel disparity [1] (i.e., positional deviations of the corresponding points in the left and right images) provides information to estimate the depth. This information is often presented in the form of a *Depth Map*, as illustrated in Figure 2, in this case, a gray scale image indicating how further the objects are in the scene.

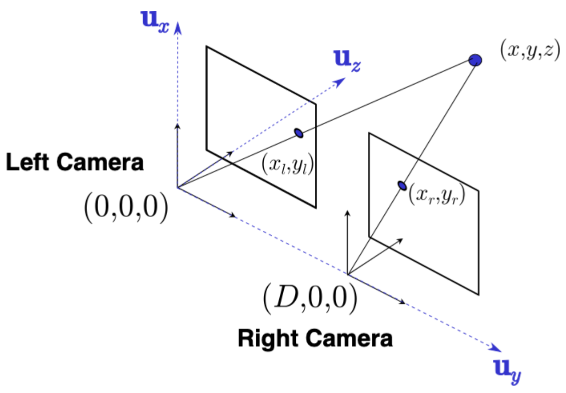


Fig. 1. Diagram of stereo caption.

A crucial step to obtain the depth maps is performed by *stereo matching algorithms*, which are used to find correspondances between pixels (or regions) in the left and right image. Several stereo matching algorithms have been proposed in the literature [2], [3], and are traditionally structured as a four step procedure: matching cost computation, cost aggregation, disparity computation/optimization and result refinement [4].

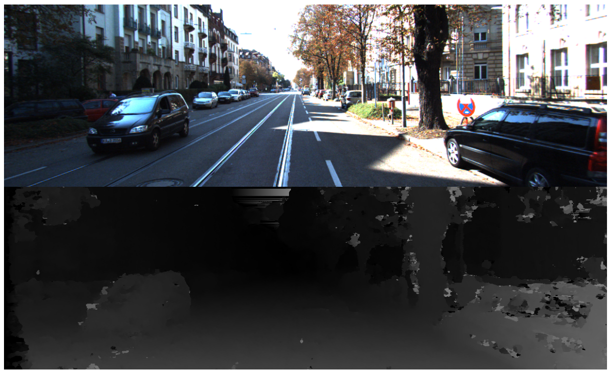


Fig. 2. Original scene and the corresponding Depth map in gray scale formed from a stereoscopic pair, where darker colors means further objetcs, while light colors means near objects.

More recently, other approaches exploring deep learning techniques [5]–[7], performing one or more steps in the traditional stereo matching approach [4], or designed as an end-to-end solution [1], [8], have been proposed. The use of neural networks in stereo matching have led to significant improvements in accuracy and speed [6], [8].

One such approach explores convolutional neural networks (CNN) for stereo matching [1], [5], processing each image through a set of convolutional layers, followed by rectified linear units. The representations of both images are then compared with a product layer, which computes the inner product between both representations to obtain the matching score. For example, Figure 2 illustrates the general struture used in [1], [5], composed of siamese branches, each one processing an image of the stereoscopic pair.

The purpose of this work is to study the approach described in [5], expanding the experiments already presented to assess the influence of structural parameters in the final result. Hence, the paper is organized as follows. First, a short introduction with the main concepts is presented and related papers exploring CNN for stereo matching are revisited. Afterwards, the methodology used in the paper is described and simulation

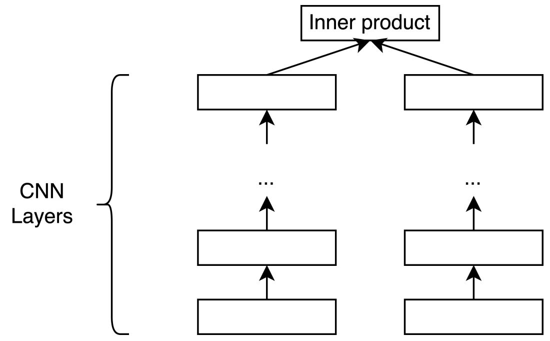


Fig. 3. Diagram representing the matching process of [5].

results are discussed. Finally, the conclusions and perspectives for future works are indicated in the final section of the paper.

II. Convolutional Neural Networks for Stereo Matching

CNNs are powerful structures for image processing and basically work by learning filters to extract characteristics for the different objects present in the scene [9]. Nevertheless, the configuration of a CNN can involve a large number of parameters, and its training may be computationally demanding. That is why dedicated Graphics Processing Units (GPU) are often used when working with these structures.

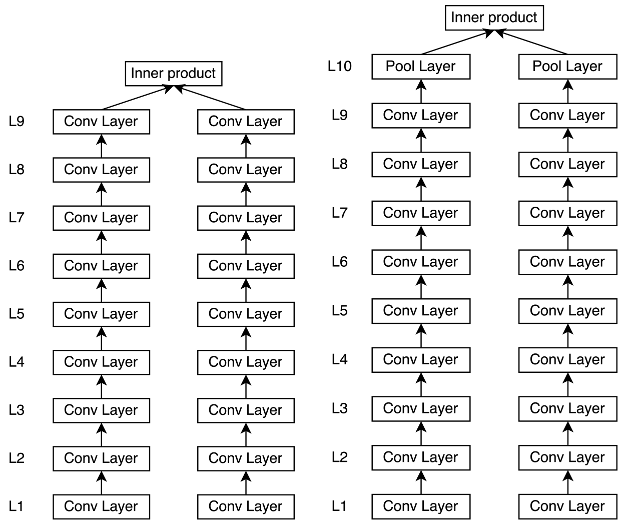
The use of CNN in stereo matching received a new burst since the publication of [6]. The structure proposed in [6] is trained with a small group of images with known disparities, and it is composed of nine layers: the first three layers are structured as two equal branches, each one for an image of the stereoscopic pair; the results of both branches are then concatenated and processed through the subsequent layers. A similar structure was studied in [5] and [10], but using more convolutional layers in the processing pipeline. Figure 3 illustrates the process of matching proposed in [5].

The result of stereo matching can then be used to estimate the depth in a stereo image [11]. It uses the model proposed in [6], incorporating few modifications such as the method proposed in [12], which compares the intensity for the three color channels. Another interesting approach related to this subject was proposed in [13], which uses a CNN cascade composed of two stages, where the first stage is responsible for producing an initial disparity space image (DSI) while the second takes on the task of refining the result [13].

III. Methodology

In this work we consider the structure proposed in [5], and presented in detail in Figure 4. The performance of the structure is evaluated in terms of the: 1- Size of convolutional kernel; 2- Size of pooling kernel; and 3- Differences between max and average pooling values.

Following the same approach as [14], in this work we consider a pixel to be correctly estimated if the disparity is <3px or if flow end-point error is $<5\%$. The flow end-point error is a good metric for moving objects because it measure



(a) Structure proposed in [5] and used for convolutional kernel experiment.

(b) Modifications proposed in this paper for pooling kernel study.

Fig. 4. Digram of architecture structure.

the euclidina distance between true point of the pixel and the detected point [15]. In this sense, the Stereo Error Rate (SER) is used as a performance metric and was calculated by identifying how many pixels did not match over the set. An additional metric, the cross entropy loss, is also used in the comparison, with values can go from 0 to infinity, where values closer to 0 indicate better models.

It is necessary to comment that the original network setup in [5] does not include a pooling layer, and the analysis in the paper does not make clear the reasons why it is not used. Therefore, as another contribution of this work, it will be added so that its effect in the performance can be evaluated. So just the kernel size experiment is able to compare with [5].

Simulation results are obtained varying a single parameter at a time. Performance is evaluated in terms of SER and loss values for the training and test set for each parameter change. Analysis will focus on understanding and explaining the observed behaviors, investigating the influence and efficiency of the parameter on the final values of the obtained model.

An implementation of the method presented in [5] was made available by the authors. It was written using the *Lua* programming language and makes use of the *Torch* framework to efficiently compute the final solution. Although the original software is public, a new version in *Python* was developed with *Tensorflow* as a library to efficiently manipulate machine learning models, and the results presented in this paper are based on the code described in [16]. The hardware used in this paper and in [5] is compared in Table I.

The pre-processing established in [5] is necessary to adapt the image to the neural network. The process consists of accessing the files that contain the images, already in the form of DSI, partition the files in training and validation, and identify in each image of the dataset which pixels will be analyzed, extracting their respective positions, both in the left

TABLE I
ENVIRONMENT USED IN [5] AND THIS PAPER.

	Original	This paper
Hardware	-	Virtual Machine cloud based
RAM	-	60 GB
SO	-	Ubuntu 18.04
GPU	NVIDIA TITAN-X	NVIDIA TESLA P100
Prog. Language	*Lua*	*Python 3.6*
Framework	*Torch*	*Tensorflow*

and right images. At the end, three files will be generated: two containing the position information of the pixels and the identification of which image is being analyzed because they have been exchanged and will be used for training and validation; and a file that identifies the images exchanged.

A. Dataset

The dataset used in the simulations was the Kitti dataset, which has been used as *benchmark* for applications involving machine learning applied to images. It contains 200 images, all composed by the stereoscopic pair [14].

B. *Kernel size*

The experiment was set up in order to obtain the accuracy and loss values of the network. The procedure adopted consists of creating a model for odd kernel sizes - three, five, seven, nine and eleven. This experiment follows the structure ilustrated on Figure 4a. For each kernel size the SER, and models's loss were evaluated. The SER value was acquired when the model has finished the training process and it is evaluated with a test image.

C. *Size of pooling kernel*

The experiment to verify the influence of the size of the pooling *kernel* has some interesting peculiarities. Since the original network does not include a pooling layer, its structure was modified, bearing in mind that structural change within the network might deteriorate its original properties. For this experiment it was applied the structure ilustrated in Figure 4b.

The pooling layer was inserted right after the last convolutional layer. Its location was chosen based on the purpose of the convolutional layer, which is to enhance the image characteristics by means of the convolution operation. Hence, after nine convolutional layers, the characteristics sought would already be well highlighted and ready to be pooled.

The measures acquired were the same as the experiment for the convolutional kernel size, this means the metrics extracted from the experiment were the SER and the model's loss. The loss was displayed during the training process and the SER when the model was ready, the test software extracts this value.

IV. EXPERIMENTAL EVALUATION

As described in the Methodology section, the experiments were divided into two parts, the first focused on reproducing the results of the original article within the existing limitations; while the second focuses on varying parameters of the convolutional neural network in order to verify their influence on the model's error rate and loss values.

TABLE II
METRICS COMPARISON BETWEEN THE ORIGINAL PAPER AND THIS ONE.

	Execution Time(s)	SER(%)
Original	0,34	7,23
This paper	0,3110	6

For the first part of the experiment, two metrics were considered: the execution time for the stereo matching of one image and the Stereo Error Rate (SER). Table II presents the results obtained in the simulations.

As can be seen in Table II the execution time of the experiment reproduction was close to the original, varying only 8.59%, difference is within the expected, since it is impossible to reproduce the original environment. Next, was acquired the SER and the results are found in Table II.

According to Table II the reproduction results are compatible with the original, despite the difference of 1.23% between the measurements, the difference seems reasonable to affirm that the reproduction of the results was guaranteed and ensured that it was possible to proceed with the proposed changes.

A. *Size of convolutional kernel*

The influence of the convolutional kernel size was evaluated in terms of the SER and the Loss function value, and the results are presented in Table III.

TABLE III
SER AND LOSS ACCORDING OF CONVOLUTIONAL KERNEL.

Kernel size	SER	Loss
3	6,442	2,095
5	6,482	1,870
7	7,662	1,805
9	2,726	1,675
11	2,308	1,800

The value of the error rate decreases with the increase in the size of the convolutional kernel, the only exception having occurred with the kernel size equal to 7. This fact is intriguing since larger kernels produce an increase in the area covered by the convolution, and therefore a greater generalization of the analyzed region, which should cause enhancement of the characteristics. The values of loss also decrease with the increase of kernel size, indicating that model is improving, the only exception was the size equals to 11, although the diference is less than 7%. It must be remembered that these values were obtained without making any structural changes to the original network, that is, at this point there is still no addition of a pooling layer within the network.

Regarding the training time, as expected, it was directly related to the kernel size. For the smallest kernel the training time was about five hours, while with the largest kernel took approximately 24 hours, using the GPU.

B. *Size of the pooling kernel*

Following the methodology presented in the previous section, it was inserted a max-pooling layer in the network and its kernel was evaluated for the following values: three, five, seven and nine. The previous nine convolutional layers were kept with kernel size set to five to preserve the original network in [5]. The impact of the pooling kernel size for max value extraction is summarized in Table IV.

TABLE IV
SER ACCORDING POOLING KERNEL SIZE WITH CONVOLUTIONAL KERNEL SET TO 5.

Kernel size	SER	Loss
3	5,510	1,964
5	4,176	1,755
7	5,186	2,253
9	6,276	2,261

Looking at Table IV one can notice that the best result is obtained when the size of the pooling kernel is equal to five, the same size used in the convolutional layer. It is also interesting to observe the loss values obtained by the models built. The model with the lowest loss value in this experiment used a pooling kernel with size five, thus representing the best setting for minimization of losses and maximization of accuracy, a situation diametrically opposite to the kernel of size nine, which obtained the highest values of loss and error rate among the analyzed values.

It should also be mentioned that the pooling layer is able to improve the performance of the network. Comparing the results with those presented in Table III for a convolutional layer kernel of size 5, it is clear that the additional processing layer is able to reduce the SER and the Loss value.

C. *Diferences between max and average pooling values*

As presented in the methodology section, simulations to compare pooling strategies were carried out and the results are displayed in Table V.

TABLE V
SER ACCORDING TO THE TYPE OF POOLING.

Kernel size	SER		Loss	
	Maximum	average	Maximum	average
3	5,510	6,734	1,964	1,881
5	4,176	9,406	1,755	1,719
7	5,186	3,862	2,253	1,994
9	6,276	7,432	2,261	2,088

Observing Table V we can compare the values for the two different pooling techniques, one trying to extract the maximum value of the area analyzed by the pooling kernel, whose values are the same as the experiment on the analysis of the effect of the layer of pooling on the structure; and another that extracts the average value of the analyzed region. Such a comparison considered a convolutional kernel of size five and convolutional and pooling strides set to one, with only the values of the pooling kernel being varied for both pooling techniques.

Analyzing the accuracy of the values found in the Table V, the maximum strategy outperforms the average strategy in most of the cases, with the only exception when the pooling kernel was seven. On the other hand, in terms of the loss value, the average strategy seems to perform better, the only exception being the kernel of size three.

The experiment showed that depending on the kernel size, it may be interesting to consider a different pooling strategy. It is not clear, however, how the choices are related. Nevertheless, as discussed in the previous subsection, these results reinforce that a additional pooling layer may be an interesting asset in structures for stereo matching.

V. CONCLUSION

In this work we studied the influence of structural parameters, such as size of *kernel* and pooling strategies on the performance of the neural network for stereo matching. To this end, this work sought to reproduce the results of an article that deals with the topic and to explore the influence of the parameters mentioned above in function of the results of error rate and losses of the neural model. The results obtained modifying the parameters, and inserting an additional pooling layer, revealed to be beneficial.

REFERENCES

[1] J. Žbontar and Y. LeCun, “Stereo matching by training a convolutional neural network to compare image patches,” 2015.

[2] F. Remondino, M. Spera, E. Nocerino, F. Menna, and F. Nex, “State of the art in high density image matching,” *Photogrammetric record*, vol. 29, no. 146, pp. 144–166, 2014.

[3] R. A. Hamzah and H. Ibrahim, “Literature survey on stereo vision disparity map algorithms,” *Journal of Sensors*, vol. 2016, 2016.

[4] D. Scharstein, R. Szeliski, and R. Zabih, “A taxonomy and evaluation of dense two-frame stereo correspondence algorithms,” in *Proceedings IEEE Workshop on Stereo and Multi-Baseline Vision (SMBV 2001)*, 2001, pp. 131–140.

[5] W. Luo, A. G. Schwing, and R. Urtasun, “Efficient deep learning for stereo matching,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5695–5703.

[6] J. Zbontar and Y. LeCun, “Computing the stereo matching cost with a convolutional neural network,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1592–1599.

[7] S. Tulyakov, A. Ivanov, and F. Fleuret, “Practical deep stereo (pds): Toward applications-friendly deep stereo matching,” 2018.

[8] J.-R. Chang and Y.-S. Chen, “Pyramid stereo matching network,” 2018.

[9] J. Schennings, “Deep convolutional neural networks for real-time single frame monocular depth estimation,” 2017.

[10] S. Zagoruyko and N. Komodakis, “Learning to compare image patches via convolutional neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4353–4361.

[11] T. S. Jordan, S. Shridhar, and J. Thatte, “Usings cnns to estimate depth from stereo imagery.”

[12] K. Zhang, J. Lu, and G. Lafruit, “Cross-based local stereo matching using orthogonal integral images,” *IEEE transactions on circuits and systems for video technology*, vol. 19, no. 7, pp. 1073–1079, 2009.

[13] J. Pang, W. Sun, J. S. Ren, C. Yang, and Q. Yan, “Cascade residual learning: A two-stage convolutional neural network for stereo matching,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 887–895.

[14] M. Menze and A. Geiger, “Object scene flow for autonomous vehicles,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[15] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, “A database and evaluation methodology for optical flow,” *International Journal of Computer Vision*, vol. 92, no. 1, pp. 1–31, Mar. 2011.

[16] Y. Wang, “stereo_matching,” Oct. 2019. [Online]. Available: https://github.com/wangy12/stereo_matching