Efficient Deep Learning for Stereo Matching

Wenjie Luo Alexander G. Schwing Raquel Urtasun
Department of Computer Science, University of Toronto
{wenjie, aschwing, urtasun}@cs.toronto.edu

Abstract

In the past year, convolutional neural networks have been shown to perform extremely well for stereo estimation. However, current architectures rely on siamese networks which exploit concatenation followed by further processing layers, requiring a minute of GPU computation per image pair. In contrast, in this paper we propose a matching network which is able to produce very accurate results in less than a second of GPU computation. Towards this goal, we exploit a product layer which simply computes the inner product between the two representations of a siamese architecture. We train our network by treating the problem as multi-class classification, where the classes are all possible disparities. This allows us to get calibrated scores, which result in much better matching performance when compared to existing approaches.

1. Introduction

Reconstructing the scene in 3D is key in many applications such as robotics and self-driving cars. To ease this process, 3D sensors such as LIDAR are commonly employed. Utilizing cameras is an attractive alternative, as it is typically a more cost-effective solution. However, despite decades of research, estimating depth from a stereo pair is still an open problem. Dealing with cclusions, large saturated areas and repetitive patterns are some of the remaining challenges.

Many approaches have been developed that try to aggregate information from local matches. Cost aggregation, for example, averages disparity estimates in a local neighborhood. Similarly, semi-global block matching and Markov random field based methods combine pixelwise predictions and local smoothness into an energy function. However all these approaches employ cost functions that are hand crafted, or where only a linear combination of features is learned from data.

In the past few years we have witnessed a revolution in high-level vision, where deep representations are learned directly from pixels to solve many scene understanding tasks with unprecedented performance. These approaches currently are the state-of-the-art in tasks such as detection, segmentation and classification.

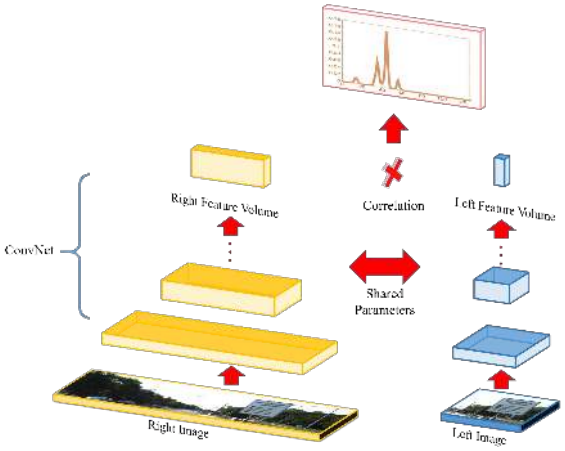


Figure 1: To learn informative image patch representations we employ a siamese network which extracts marginal distributions over all possible disparities for each pixel.

Very recently, convolutional networks have also been exploited to learn how to match for the task of stereo estimation [30, 28]. Current approaches learn the parameters of the matching network by treating the problem as binary classification; Given a patch in the left image, the task is to predict if a patch in the right image is the correct match. While [29] showed great performance in challenging benchmarks such as KITTI [11], it is computationally very expensive, requiring a minute of computation in the GPU. This is due to the fact that they exploited a siamese architecture followed by concatenation and further processing via a few more layers to compute the final score.

In contrast, in this paper we propose a matching network which is able to produce very accurate results in less than a second of GPU computation. Towards this goal, we exploit a product layer which simply computes the inner product between the two representations of a siamese architecture. We train our network by treating the problem as multi-class classification, where the classes are all possible disparities. This allows us to get calibrated scores, which result in much

better matching performance when compared to [29]. We refer the reader to Fig. 1 for an illustration of our approach. We demonstrate the effectiveness of our approach on the challenging KITTI benchmark and show competitive results when exploiting smoothing techniques. Our code and data can be fond online at: http://www.cs.toronto.edu/deepLowLevelVision.

2. Related Work

Over the past decades many stereo algorithms have been developed. Since a discussion of all existing approaches would exceed the scope of this paper, we restrict ourselves mostly to a subset of recent methods that exploit learning and can mostly be formulated as energy minimization.

Early learning based approaches focused on correcting an initially computed matching cost [16, 17]. Learning has been also utilized to tune the hyper-parameters of the energy-minimization task. Among the first to train these hyper-parameters were [31, 21, 19], which investigated different forms of probabilistic graphical models.

Slanted plane models model groups of pixels with slanted 3D planes. They are very competitive in autonomous driving scenarios, where robustness is key. They have a long history, dating back to [2] and were shown to be very successful on the Middleburry benchmark [22, 15, 3, 24] as well as on KITTI [25, 26, 27].

Holistic models which solve jointly many tasks have also been explored. The advantage being that many tasks in low-level and high level-vision are related, and thus one can benefit from solving them together. For example [5, 6, 4, 18, 13] jointly solved for stereo and semantic segmentation. Guney and Geiger [12] investigated the utility of high-level vision tasks such as object recognition and semantic segmentation for stereo matching.

Estimating the confidence of each match is key when employing stereo estimates as a part of a pipeline. Learning methods were successfully applied to this task, *e.g.*, by combining several confidence measures via a random forest classifier [14], or by incorporating random forest predictions into a Markov random field [23].

Convolutional neural networks(CNN) have been shown to perform very well on high-level vision tasks such as image classification, object detection and semantic segmentation. More recently, CNNs have been applied to low-level vision tasks such as optical flow prediction [10]. In the context of stereo estimation, [29] utilize CNN to compute the matching cost between two image patches. In particular, they used a siamese network which takes the same sized left and right image patches with a few fully-connected layers on top to predict the matching cost. They trained the model to minimize a binary cross-entropy loss. In similar spirit to [29], [28] investigated different CNN based architectures for comparing image patches. They found concatenating left and right image patches as different channels works best, at the cost of being very slow.

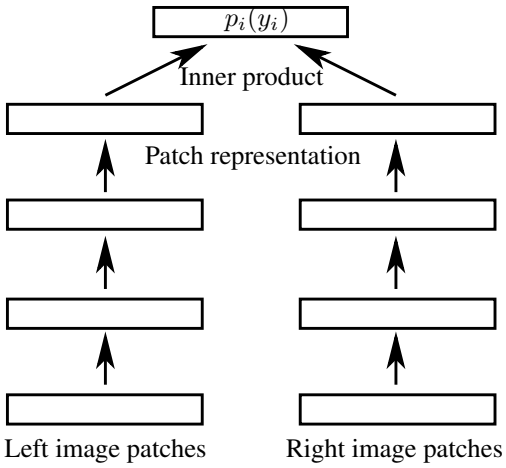


Figure 2: Our four-layer siamese network architecture which has a receptive field size of 9.

Our work is most similar to [29, 28] with two main differences. First, we propose to learn a probability distribution over all disparity values using a smooth target distribution. As a consequence we are able to capture correlations between the different disparities implicitly. This contrasts a [29] which performs independent binary predictions on image patches. Second, on top of the convolution layers we use a simple dot-product layer to join the two branches of the network. This allows us to do a orders of magnitude faster computation. We note that in concurrent work unpublished at the time of submission of our paper [30, 7] also introduced a dot-product layer.

3. Deep Learning for Stereo Matching

We are interested in computing a disparity image given a stereo pair. Throughout this paper we assume that the image pairs are rectified, thus the epipolar lines are aligned with the horizontal image axis. Let $y_i \in \mathcal{Y}_i$ represent the disparity associated with the i-th pixel, and let $|\mathcal{Y}_i|$ be the cardinality of the set (typically 128 or 256). Stereo algorithms estimate a 3-dimensional cost volume by computing for each pixel in the left image a score for each possible disparity value. This is typically done by exploiting a small patch around the given pixel and a simple hand-crafted representation of each patch. In contrast, in this paper we exploit convolutional neural networks to learn how to match.

Towards this goal, we utilize a siamese architecture, where each branch processes the left or right image respectively. In particular, each branch takes an image as input, and passes it through a set of layers, each consisting of a spatial convolution with a small filter-size (e.g., 5×5 or

3×3), followed by a spatial batch normalization and a rectified linear unit (ReLU). Note that we remove the ReLU from the last layer in order to not loose the information encoded in the negative values. In our experiments we exploit different number of filters per layer, either 32 or 64 and share the parameters between the two branches.

In contrast to existing approaches which exploit concatenation followed by further processing, we use a product layer which simply computes the inner product between the two representations to compute the matching score. This simple operation speeds up the computation significantly. We refer the reader to Fig. 2 which depicts an example of a 4-layer network with filter-size 3×3, which results in a receptive field of size 9×9.

Training: We use small left image patches extracted at random from the set of pixels for which ground truth is available to train the network. This strategy provides us with a diverse set of examples and is memory efficient. In particular, each left image patch is of size equivalent to the size of our network's receptive field. Let (x_i, y_i) be the image coordinates of the center of the patch extracted at random from the left image, and let d_{x_i,y_i} be the corresponding ground truth disparity. We use a larger patch for the right image which expands both the size of the receptive field as well as all possible disparities (i.e., displacements). The output of the two branches of the siamese network is hence a single 64-dimensional representation for the left branch, and $|\mathcal{Y}_i| \times 64$ for the right branch. These two vectors are then passed as input to an inner-product layer which computes a score for each of the $|\mathcal{Y}_i|$ disparities. This allow us to compute a softmax for each pixel over all possible disparities.

During training we minimize cross-entropy loss with respect to the weights $\mathbf{w}$ that parameterize the network

$$\min_{\mathbf{w}} \sum_{i,y_i} p_{\text{gt}}(y_i) \log p_i(y_i, \mathbf{w}).$$

Since we are interested in a 3-pixel error metric we use a smooth target distribution $p_{\text{gt}}(y_i)$, centered around the ground-truth y_i^{GT}, *i.e.*,

$$p_{\text{gt}}(y_i) = \begin{cases} \lambda_1 & \text{if } y_i = y_i^{GT} \\ \lambda_2 & \text{if } |y_i - y_i^{GT}| = 1 \\ \lambda_3 & \text{if } |y_i - y_i^{GT}| = 2 \\ 0 & \text{otherwise} \end{cases}.$$

For this paper we set $\lambda_1 = 0.5$, $\lambda_2 = 0.2$ and $\lambda_3 = 0.05$. Note that this contrasts cross entropy for classification, where $p_{gt}(y_i)$ is a delta function placing all its mass on the annotated groundtruth configuration.

We train our network using stochastic gradient descent back propagation with AdaGrad [8]. Similar to moment-based stochastic gradient descent, AdaGrad adapts the gradient based on historical information. Contrasting moment based methods it emphasizes rare but informative features. We adapt the learning rates every few thousand iterations as detailed in the experimental section.

Testing: Contrasting the training procedure where we compose a mini-batch by randomly sampling locations from different training images, we can improve the speed performance during testing. Our siamese network computes a 64-dimensional feature representation for every pixel i. To efficiently obtain the cost volume, we compute the 64-dimensional representation only once for every pixel i, and during computation of the cost volume we re-use its values for all disparities that involve this location.

4. Smoothing Deep Net Outputs

Given the unaries obtained with a CNN, we compute predictions for all disparities at each image location. Note that simply outputting the most likely configuration for every pixel is not competitive with modern stereo algorithms, which exploit different forms of cost aggregation, post processing and smoothing. This is particularly important to deal with complex regions with occlusions, saturation or repetitive patterns.

Over the past decade many different MRFs have been proposed to solve the stereo estimation problem. Most approaches define each random variable to be the disparity of a pixel, and encode smoothness between consecutive or nearby pixels. An alternative approach is to segment the image into regions and estimate a slanted 3D plane for each region. In this paper we investigate the effect of different smoothing techniques. Towards this goal, we formulate stereo matching as inference in several different Markov random fields (MRFs), with the goal of smoothing the matching results produced by our convolutional neural network. In particular, we look into cost aggregation, semi-global block matching as well as the slanted plane approach of [27] as means of smoothing. In the following we briefly review these techniques.

Cost aggregation: We exploited a very simple cost aggregation approach, which simply performs average pooling over a window of size 5×5.

Semi global block matching: Semi-global block matching augments the unary energy term obtained from convolutional neural nets by introducing additional pairwise potentials which encourage smooth disparities. Specifically,

$$E(y) = \sum_{i=1}^{N} E_i(y_i) + \sum_{(i,j)\in\mathcal{E}} E_{i,j}(y_i, y_j),$$

where $\mathcal{E}$ refers to 4-connected grid and the unary energy $E_i(y_i)$ is the output of the neural net.

	> 2 pixel		> 3 pixel		> 4 pixel		> 5 pixel		End-Point		Runtime(s)
	Non-Occ	All	Non-Occ	All	Non-Occ	All	Non-Occ	All	Non-Occ	All	
MC-CNN-acrt [29]	15.02	16.92	12.99	14.93	12.04	13.98	11.38	13.32	4.39 px	5.21 px	20.13
MC-CNN-fast [29]	17.72	19.56	15.53	17.41	14.41	16.31	13.60	15.51	4.77 px	5.63 px	0.20
Ours(19)	**10.87**	**12.86**	**8.61**	**10.64**	**7.62**	**9.65**	**7.00**	**9.03**	**3.31** px	**4.2** px	0.14

Table 1: Comparison of the output of the matching network across different error metrics on the KITTI 2012 validation set.

Unary	CA	SGM[30]	Post[30]	Slanted[27]	Ours(9)	Ours(19)	Ours(29)	Ours(37)	MC-CNN-acrt[29]	MC-CNN-fast[29]
✓					16.69	8.61	7.64	6.61	12.99	15.53
✓	✓				12.14	7.48	6.86	6.09	6.32	-
✓	✓	✓			4.57	3.99	4.12	3.96	3.34	4.53
✓	✓	✓	✓		4.11	3.73	3.99	3.88	3.22	3.73
✓	✓	✓	✓	✓	3.96	3.64	3.81	3.83	3.36	3.83

Table 2: Comparison of different smoothing methods. The table illustrates non-occluded 3 pixel error on the KITTI 2012 validation set.

We define the pairwise energy as

$$E_{i,j}(y_i, y_j) = \begin{cases} 0 & \text{if } y_i = y_j \\ c_1 & \text{if } |y_i - y_j| = 1 \\ c_2 & \text{otherwise} \end{cases},$$

with variable constants $c_1 < c_2$. We follow the approach of [29], where c_1 and c_2 is decreased if there is strong evidence for edges at the corresponding locations in either the left or the right image. We refer the reader to their paper for more details.

Slanted plane: To construct a depth-map, this approach performs block-coordinate descent on an energy involving appearance, location, disparity, smoothness and boundary energies. More specifically, we first over-segment the image using an extension of the SLIC energy [1]. For each superpixel we then compute slanted plane estimates [27] which should adhere to the depth-map evidence obtained from the convolutional neural network. We then iterate these two steps to minimize an energy function which takes into account appearance, location, disparity, smoothness and boundary energies. We refer the interested reader to [27] for details.

Sophisticated post-processing: In [30] a three-step post-processing is designed to perform interpolation, subpixel enhancement and a refinement. The interpolation step resolves conflicts between the disparity maps computed for the left and right images by performing a left-right consistency check. Subpixel enhancement fits a quadratic function to neighboring points to obtain an enhanced depth-map. To smooth the disparity map without blurring edges, the final refinement step applies a median filter and a bilateral filter. We only use the interpolation step as we found the other two don't always further improve performance in our case.

5. Experimental Evaluation

We evaluate the performance of different convolutional neural network structures and different smoothing techniques on the KITTI 2012 [11] and 2015 [20] datasets. Before training we normalize each image to have zero mean and standard deviation of one. We initialize the parameters of our networks using a uniform distribution. We employ the AdaGrad algorithm [8] and use a learning rate of $1e^{-2}$. The learning rate is decreased by a factor of 5 after 24k iterations and then further decreased by a factor of 5 every 8k iterations. We use a batch size of 128. We trained the network for 40k iterations which takes around 6.5 hours on an NVIDIA Titan-X.

5.1. KITTI 2012 Results

The KITTI 2012 dataset contains 194 training and 195 test images. To compare the different network architectures described below, we use as training set 160 image pairs randomly selected, and the remaining 34 image pairs as our validation set.

Comparison of Matching Networks: We first show our network's matching ability and compare it to existing matching networks [29, 30]. In this experiment we do not employ smoothing or post processing, but just utilize the raw output of the network. Following KITTI, we employ the percentage of pixels with disparity errors larger than a fixed threshold as well as end-point error as metrics. We refer to our architecture as 'Ours(19).' It consists of 9 layers of 3×3 convolutions resulting in a receptive field size of 19×19 pixels. As shown in Table 1, our 9-layer network achieves a 3-pixel non-occluded stereo error of 8.61% after only 0.14 seconds of computation. In contrast, [29] obtains 12.99% after a significantly longer time of 20.13 seconds.

	> 2 pixels		> 3 pixels		> 4 pixels		> 5 pixels		End-Point		Runtime
	Non-Occ	**All**	**Non-Occ**	**All**	**Non-Occ**	**All**	**Non-Occ**	**All**	**Non-Occ**	**All**	(s)
StereoSLIC [26]	5.76	7.20	3.92	5.11	3.04	4.04	2.49	3.33	0.9 px	1.0 px	2.3
PCBP-SS [26]	5.19	6.75	3.40	4.72	2.62	3.75	2.18	3.15	0.8 px	1.0 px	300
SPS-st [27]	4.98	6.28	3.39	4.41	2.72	3.52	2.33	3.00	0.9 px	1.0 px	2
Deep Embed [7]	5.05	6.47	3.10	4.24	2.32	3.25	1.92	2.68	0.9 px	1.1 px	3
MC-CNN-acrt [30]	3.90	5.45	2.43	3.63	1.90	2.85	1.64	2.39	0.7 px	0.9 px	67
Displets v2 [12]	3.43	4.46	2.37	3.09	1.97	2.52	1.72	2.17	0.7 px	0.8 px	265
Ours(19)	4.98	6.51	3.07	4.29	2.39	3.36	2.03	2.82	0.8 px	1.0 px	0.7

Table 3: Comparison to stereo state-of-the-art on the test set of the KITTI 2012 benchmark.

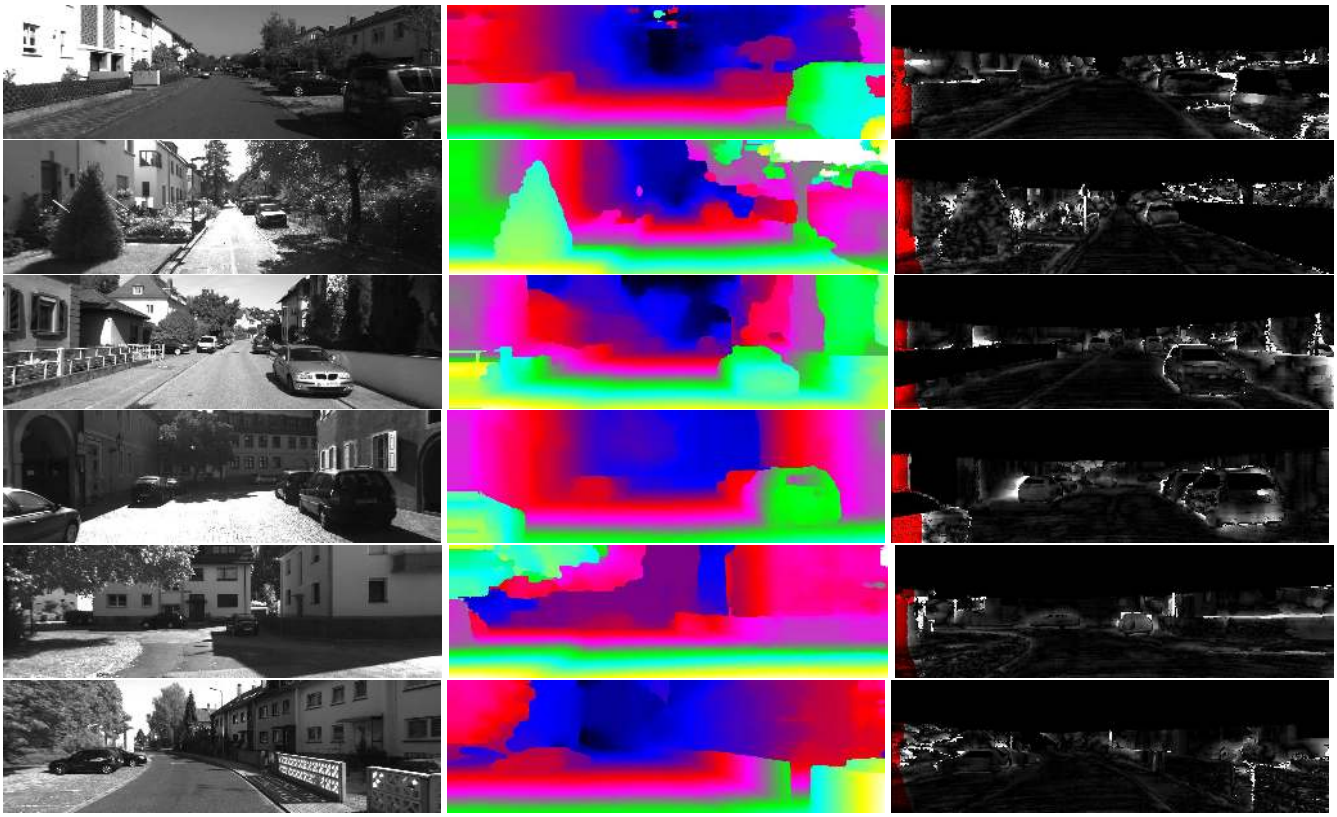


Figure 3: KITTI 2012 test set: (left) original image, (center) stereo estimates, (right) stereo errors.

Their faster version [30] requires 0.20 second which results in a much lower performance of 15.53%. As shown in the table, our network outperforms previously designed convolutional neural networks by a large margin on all criteria.

Smoothing Comparison: Next, we evaluate different algorithms for smoothing and post-processing when employing different network sizes. In particular, we evaluate cost aggregation, semi-global block matching and slanted plane smoothing, which are described in the previous section. We also experiment with different receptive field sizes for our network, which corresponds to changing the depth of our architecture. As before, we use 'Ours(n)' to refer to our architecture with a receptive field size of $n \times n$ pixel. We investigated $n = 9, 19, 29, 37$. We use kernels of size 3×3 for $n = 9$ and $n = 19$, while the kernels were of size 5×5 for $n = 39$. To achieve a receptive field of 29 we use 5 layers of 5×5 and 4 layers of 3×3. This keeps the number of layers bounded to 9.

As shown in Table 2, networks with different receptive field sizes result in errors ranging from 6.61% (for $n = 37$ to 16.69% for $n = 9$. The corresponding error for [30] is 12.99% for their slow and more accurate model, and 15.53% for their fast model. After smoothing, the differences in stereo error achieved by the networks are no longer significant. All of them achieve an error slightly less than 4%. Since depth-maps tend to be very smooth we think that an aggressive smoothing helps to flatten the noisy unary potentials. In addition we observe that utilizing simple cost aggregation to encourage local smoothness further helps to slightly improve the results. This is due to the fact that

	> 2 pixel		> 3 pixel		> 4 pixel		> 5 pixel		End-Point		Runtime(s)
	Non-Occ	All	Non-Occ	All	Non-Occ	All	Non-Occ	All	Non-Occ	All	
MC-CNN-acrt [29]	15.20	16.83	12.45	14.12	11.04	12.72	10.13	11.80	4.01 px	4.66 px	22.76
MC-CNN-fast [29]	18.47	20.04	14.96	16.59	13.18	14.83	12.02	13.67	4.27 px	4.93 px	0.21
Ours(37)	**9.96**	**11.67**	**7.23**	**8.97**	**5.89**	**7.62**	**5.04**	**6.78**	**1.84** px	**2.56** px	0.34

Table 4: Comparison of the output of the matching network across different error metrics on the KITTI 2015 validation set.

Unary	CA	SGM[30]	Post[30]	Slanted[27]	Ours(9)	Ours(19)	Ours(29)	Ours(37)	MC-CNN-acrt[29]	MC-CNN-fast[29]
✓					15.25	8.95	7.23	7.13	12.45	14.96
✓	✓				11.43	8.00	6.60	6.58	7.78	-
✓	✓	✓			5.18	4.74	4.62	4.73	3.48	5.05
✓	✓	✓	✓		4.41	4.23	4.31	4.38	3.10	4.74
✓	✓	✓	✓	✓	4.25	4.20	4.14	4.19	3.11	4.79

Table 5: Comparison of smoothing methods using different CNN output. The table illustrates the non-occluded 3 pixel error on the KITTI 2015 validation set.

such techniques eliminate small isolated noisy areas. While the post-processing proposed in [30] focuses on occlusions and sub-pixel enhancement, [27] adds extra robustness to non-textured areas by fitting slanted planes. Both methods improve the semi-global block matching output slightly. Our best performing model combination achieves a 3 pixel stereo error of 3.64%.

Comparison to State-of-the-art: To evaluate the test set performance we trained our model having a receptive field of 19 pixels, *i.e.*, "Ours(19)," on the entire training set. The obtained test set performance is shown in Table 3. Since we did not particularly focus on finding a good combination of smoothness and unaries, our performance is slightly below the current state-of-the-art.

Qualitative Analysis: Examples of stereo estimates by our approach are depicted in Fig. 3. We observe that our approach suffers from texture-less regions as well as regions with repetitive patterns such as fences.

5.2. KITTI 2015 Results

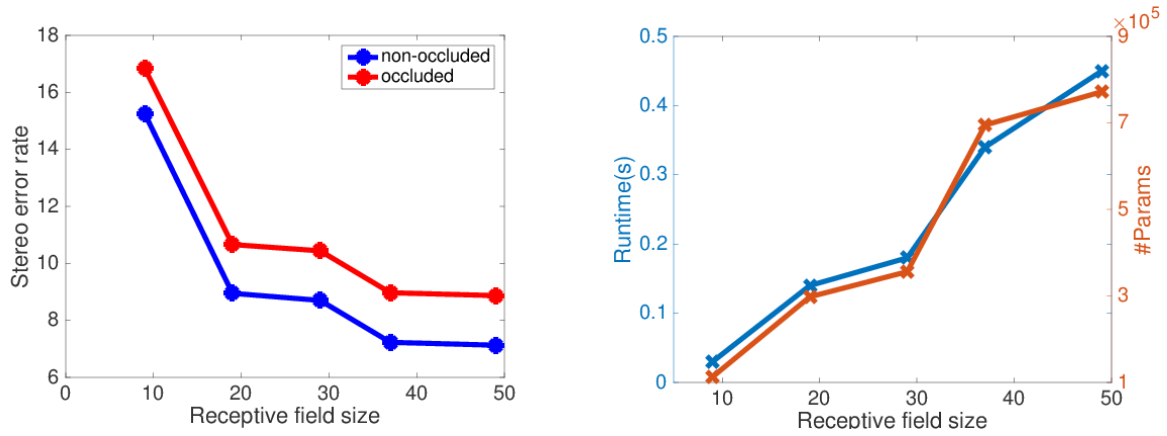
The KITTI 2015 dataset consists of 200 training and 200 test images. Instead of the gray-scale images used for the KITTI 2012 dataset we directly process the RGB data. To compare the different network architectures, we randomly selected 160 image pairs as training set and use the remaining 40 image pairs for validation purposes.

Comparison of Matching Networks: We first show our network's matching ability and compare it to existing matching networks [29, 30]. In this experiment we do not employ smoothing or post processing, but just utilize the raw output of the network. We refer to our architecture via 'Ours(37).' It consists of 9 layers of 5×5 convolutions resulting in a receptive field size of 37×37 pixels. As shown in Table 4, our 9-layer network achieves a 3-pixel stereo error of 7.23% after only 0.34 seconds of processing time, whereas [29] obtains 12.45% after a significantly longer processing time of 22.76 seconds. Their faster version [30] requires 0.21 seconds but results in a much lower performance of 14.96% when compared to our approach. Again, our network outperforms previously designed convolutional neural networks by a large margin on all criteria.

Smoothing Comparison: Table 5 shows results of applying different post processing techniques to different network architectures. As when processing KITTI 2012 images, we observe that the difference in network performance vanishes after applying smoothing techniques. Our best performing combination achieves a 3-pixel error of 4.14% on the validation set.

Influence of Depth and Filter Size Next, we evaluate the influence of the depth and receptive field size of our CNNs in terms of matching performance and running time. Fig. 4a shows matching performance as a function of the networks' receptive field size. We observe that an increasing receptive field size achieves better performance. However, when the receptive field is very large, the improvement is subtle, since the network starts to overlook the details of small objects and depth discontinuities. Our findings are consistent for both non-occluded and for all pixel. As shown in Fig. 4b, the running time and number of parameters are highly correlated. Note that models with larger receptive field do not necessarily have more parameters since the amount of trainable weights also depends on the number of filters and the channel size of each convolutional layer.

Comparison to State-of-the-art: To evaluate the test set performance, we choose the best model with current smoothing techniques, which has a receptive field of 37



(a) Stereo error using only the matching network (unaries)

(b) Runtime and number of parameters over receptive field

Figure 4: Evaluation of stereo error (a), runtime and number of parameters (b).

	All/All			All/Est			Noc/All			Noc/Est			Runtime
	D1-bg	D1-fg	D1-all	D1-bg	D1-fg	D1-all	D1-bg	D1-fg	D1-all	D1-bg	D1-fg	D1-all	(s)
MBM [9]	4.69	13.05	6.08	4.69	13.05	6.08	4.33	12.12	5.61	4.33	12.12	5.61	0.13
SPS-St [27]	3.84	12.67	5.31	3.84	12.67	5.31	3.50	11.61	4.84	3.50	11.61	4.84	2
MC-CNN [30]	2.89	8.88	3.89	2.89	8.88	3.88	2.48	7.64	3.33	2.48	7.64	3.33	67
Displets v2 [12]	3.00	5.56	3.43	3.00	5.56	3.43	2.73	4.95	3.09	2.73	4.95	3.09	265
Ours(37)	3.73	8.58	4.54	3.73	8.58	4.54	3.32	7.44	4.00	3.32	7.44	4.00	1

Table 6: Comparison to stereo state-of-the-art on the test set of KITTI 2015 benchmark.

pixel, *i.e.*, "Ours(37)." The obtained test set performance is shown in Table 6. We achieve on-par results in significantly less time.

Qualitative Results: We provide results from the test set in Fig. 5. Again, we observe that our algorithm suffers from texture-less regions as well as regions with repetitive patterns.

6. Conclusion

Convolutional neural networks have been recently shown to perform extremely well for stereo estimation. Current architectures rely on siamese networks which exploit concatenation follow by further processing layers, requiring a minute on the GPU to process a stereo pair. In contrast, in this paper we have proposed a matching network which is able to produce very accurate results in less than a second of GPU computation. Our key contribution is to replace the concatenation layer and subsequent processing layers by a single product layer, which computes the score. We trained the networks using cross-entropy over all possible disparities. This allows us to get calibrated scores, which result in much better matching performance when compared to existing approaches. We have also investigated the effect of different smoothing techniques to further improve performance. In the future we plan to utilize our approach for other low-level vision tasks such as optical flow. We also plan to build smoothing techniques that are tailored to our approach.

Acknowledgments: This work was partially supported by ONR-N00014-14-1-0232 and NSERC. We would like to thank NVIDIA for supporting our research by donating GPUs and Shenlong Wang for help with the figures.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *PAMI*, 2012. 4

[2] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *CVPR*, 1999. 2

[3] M. Bleyer and M. Gelautz. A layered stereo matching algorithm using image segmentation and global visibility constraints. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2005. 2

[4] M. Bleyer, C. Rhemann, and C. Rother. Extracting 3D scene-consistent object proposals and depth from stereo images. In *ECCV*, 2012. 2

[5] M. Bleyer, C. Rother, and P. Kohli. Surface stereo with soft segmentation. In *CVPR*, 2010. 2

[6] M. Bleyer, C. Rother, P. Kohli, D. Scharstein, and S. Sinha. Object stereo - joint stereo matching and object segmentation. In *CVPR*, 2011. 2

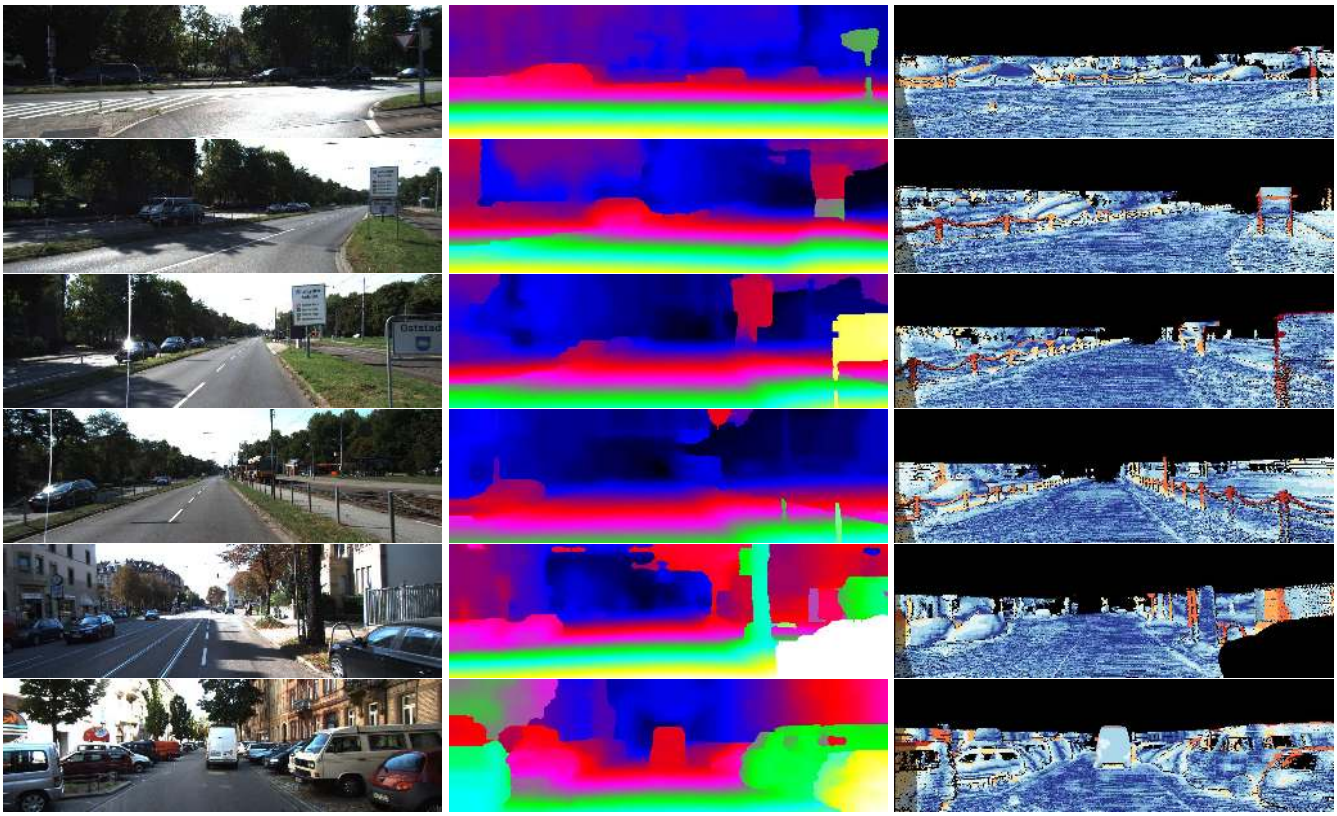


Figure 5: KITTI 2015 test set: (left) original image, (center) stereo estimates, (right) stereo errors.

[7] Z. Chen, X. Sun, L. Wang, Y. Yu, and C. Huang. A deep visual correspondence embedding model for stereo matching costs. In *ICCV*, 2015. 2, 5

[8] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *JMLR*, 12:2121–2159, 2011. 3, 4

[9] N. Einecke and J. Eggert. A multi-block-matching approach for stereo. In *Intelligent Vehicles Symposium (IV)*, 2015. 7

[10] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazirbaş, and V. Golkov. FlowNet: Learning Optical Flow with Convolutional Networks. In *ICCV*, 2015. 2

[11] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 1, 4

[12] F. Guney and A. Geiger. Displets: Resolving stereo ambiguities using object knowledge. In *CVPR*, 2015. 2, 5, 7

[13] C. Haene, L. Ladický, and M. Pollefeys. Direction Matters: Depth Estimation with a Surface Normal Classifier. In *CVPR*, 2015. 2

[14] R. Haeusler, R. Nair, and D. Kondermann. Ensemble learning for confidence measures in stereo vision. In *CVPR*, 2013. 2

[15] A. Klaus, M. Sormann, and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *ICPR*, 2006. 2

[16] D. Kong and H. Tao. A method for learning matching errors for stereo computation. In *BMVC*, 2004. 2

[17] D. Kong and H. Tao. Stereo matching via learning multiple experts behaviors. In *BMVC*, 2006. 2

[18] L. Ladický, J. Shi, and M. Pollefeys. Pulling Things out of Perspective. In *CVPR*, 2014. 2

[19] Y. Li and D. P. Huttenlocher. Learning for stereo vision using the structured support vector machine. In *CVPR*, 2008. 2

[20] M. Menze and A. Geiger. Object Scene Flow for Autonomous Vehicles. In *CVPR*, 2015. 4

[21] D. Scharstein and C. Pal. Learning conditional random fields for stereo. In *CVPR*, 2007. 2

[22] D. Scharstein and R. Szeliski. Middlebury stereo vision page. *Online at http://www. middlebury. edu/stereo*, 2002. 2

[23] A. Spyropoulos, N. Komodakis, and P. Mordohai. Learning to detect ground control points for improving the accuracy of stereo matching. In *CVPR*, 2014. 2

[24] Z.-F. Wang and Z.-G. Zheng. A region based stereo matching algorithm using cooperative optimization. In *CVPR*, 2008. 2

[25] K. Yamaguchi, T. Hazan, D. McAllester, and R. Urtasun. Continuous markov random fields for robust stereo estimation. In *ECCV*, 2012. 2

[26] K. Yamaguchi, D. McAllester, and R. Urtasun. Robust monocular epipolar flow estimation. In *CVPR*, 2013. 2, 5

[27] K. Yamaguchi, D. McAllester, and R. Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *ECCV*. 2014. 2, 3, 4, 5, 6, 7

[28] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, 2015. 1, 2

[29] J. Zbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *CVPR*, 2015. 1, 2, 4, 6

[30] J. Žbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *arXiv preprint arXiv:1510.05970*, 2015. 1, 2, 4, 5, 6, 7

[31] L. Zhang and S. M. Seitz. Estimating optimal parameters for MRF stereo from a single image pair. *PAMI*, 2007. 2