Computing the Stereo Matching Cost with a Convolutional Neural Network

Jure Žbontar
University of Ljubljana
jure.zbontar@fri.uni-lj.si

Yann LeCun
New York University
yann@cs.nyu.edu

Abstract

We present a method for extracting depth information from a rectified image pair. We train a convolutional neural network to predict how well two image patches match and use it to compute the stereo matching cost. The cost is refined by cross-based cost aggregation and semiglobal matching, followed by a left-right consistency check to eliminate errors in the occluded regions. Our stereo method achieves an error rate of 2.61 % on the KITTI stereo dataset and is currently (August 2014) the top performing method on this dataset.

1. Introduction

Consider the following problem: given two images taken from cameras at different horizontal positions, the goal is to compute the disparity d for each pixel in the left image. Disparity refers to the difference in horizontal location of an object in the left and right image—an object at position (x, y) in the left image will appear at position $(x - d, y)$ in the right image. Knowing the disparity d of an object, we can compute its depth z (*i.e*. the distance from the object to the camera) by using the following relation:

$$z = \frac{fB}{d}, \tag{1}$$

where f is the focal length of the camera and B is the distance between the camera centers.

The described problem is a subproblem of stereo reconstruction, where the goal is to extract 3D shape from one or more images. According to the taxonomy of Scharstein and Szeliski [14], a typical stereo algorithm consists of four steps: (1) matching cost computation, (2) cost aggregation, (3) optimization, and (4) disparity refinement. Following Hirschmuller and Scharstein [5], we refer to steps (1) and (2) as computing the matching cost and steps (3) and (4) as the stereo method.

We propose training a convolutional neural network [9] on pairs of small image patches where the true disparity is known (*e.g*. obtained by LIDAR). The output of the network is used to initialize the matching cost between a pair of patches. Matching costs are combined between neighboring pixels with similar image intensities using cross-based cost aggregation. Smoothness constraints are enforced by semiglobal matching and a left-right consistency check is used to detect and eliminate errors in occluded regions. We perform subpixel enhancement and apply a median filter and a bilateral filter to obtain the final disparity map. Figure 1 depicts the inputs to and the output from our method. The two contributions of this paper are:

- We describe how a convolutional neural network can be used to compute the stereo matching cost.
- We achieve an error rate of 2.61 % on the KITTI stereo dataset, improving on the previous best result of 2.83 %.

2. Related work

Before the introduction of large stereo datasets [2, 13], relatively few stereo algorithms used ground-truth information to learn parameters of their models; in this section, we review the ones that did. For a general overview of stereo algorithms see [14].

Kong and Tao [6] used sum of squared distances to compute an initial matching cost. They trained a model to predict the probability distribution over three classes: the initial disparity is correct, the initial disparity is incorrect due to fattening of a foreground object, and the initial disparity is incorrect due to other reasons. The predicted probabilities were used to adjust the initial matching cost. Kong and Tao [7] later extend their work by combining predictions obtained by computing normalized cross-correlation over different window sizes and centers. Peris et al. [12] initialized the matching cost with AD-Census [11] and used multiclass linear discriminant analysis to learn a mapping from the computed matching cost to the final disparity.

Ground-truth data was also used to learn parameters of graphical models. Zhang and Seitz [22] used an alternative optimization algorithm to estimate optimal values of Markov random field hyperparameters. Scharstein and Pal

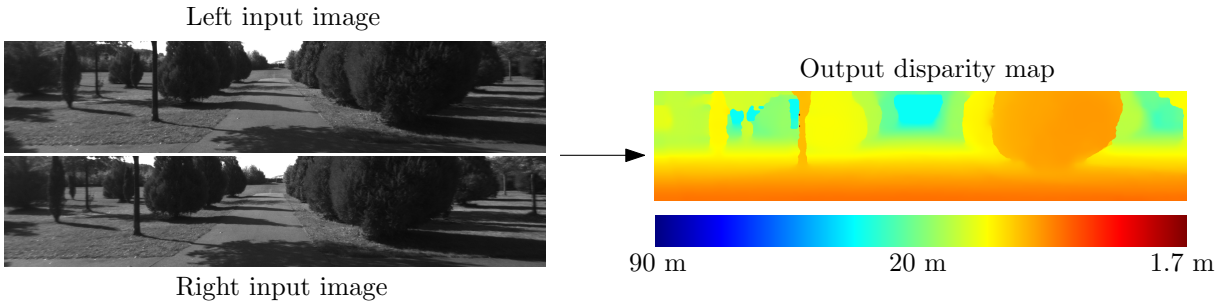

Figure 1. The input is a pair of images from the left and right camera. The two input images differ mostly in horizontal locations of objects. Note that objects closer to the camera have larger disparities than objects farther away. The output is a dense disparity map shown on the right, with warmer colors representing larger values of disparity (and smaller values of depth).

[13] constructed a new dataset of 30 stereo pairs and used it to learn parameters of a conditional random field. Li and Huttenlocher [10] presented a conditional random field model with a non-parametric cost function and used a structured support vector machine to learn the model parameters.

Recent work [3, 15] focused on estimating the confidence of the computed matching cost. Haeusler et al. [3] used a random forest classifier to combine several confidence measures. Similarly, Spyropoulos et al. [15] trained a random forest classifier to predict the confidence of the matching cost and used the predictions as soft constraints in a Markov random field to decrease the error of the stereo method.

3. Computing the matching cost

A typical stereo algorithm begins by computing a matching cost $C(\mathbf{p}, d)$ at each position $\mathbf{p}$ for all disparities d under consideration. A simple example is the sum of absolute differences:

$$C_{\text{AD}}(\mathbf{p}, d) = \sum_{\mathbf{q} \in \mathcal{N}_{\mathbf{p}}} |I^L(\mathbf{q}) - I^R(\mathbf{qd})|, \tag{2}$$

where $I^L(\mathbf{p})$ and $I^R(\mathbf{p})$ are image intensities at position $\mathbf{p}$ of the left and right image and $\mathcal{N}_{\mathbf{p}}$ is the set of locations within a fixed rectangular window centered at $\mathbf{p}$. We use bold lowercase letters ($\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$) to denote pairs of real numbers. Appending a lowercase $\mathbf{d}$ has the following meaning: if $\mathbf{p} = (x, y)$ then $\mathbf{pd} = (x - d, y)$.

Equation (2) can be interpreted as measuring the cost associated with matching a patch from the left image, centered at position $\mathbf{p}$, with a patch from the right image, centered at position $\mathbf{pd}$. Since examples of good and bad matches can be obtained from publicly available datasets, *e.g.* KITTI [2] and Middlebury [14], we can attempt to solve the matching problem by a supervised learning approach. Inspired by the successful applications of convolutional neural networks to vision problems [8], we used them to evaluate how well two small image patches match.

3.1. Creating the dataset

A training example comprises two patches, one from the left and one from the right image:

$$< \mathcal{P}^L_{9\times 9}(\mathbf{p}), \mathcal{P}^R_{9\times 9}(\mathbf{q}) >, \tag{3}$$

where $\mathcal{P}^L_{9\times 9}(\mathbf{p})$ denotes a 9×9 patch from the left image, centered at $\mathbf{p} = (x, y)$. For each location where the true disparity d is known, we extract one negative and one positive example. A negative example is obtained by setting the center of the right patch $\mathbf{q}$ to

$$\mathbf{q} = (x - d + o_{\text{neg}}, y), \tag{4}$$

where o_{neg} is an offset corrupting the match, chosen randomly from the set $\{-N_{\text{hi}}, \ldots, -N_{\text{lo}}, N_{\text{lo}}, \ldots, N_{\text{hi}}\}$. Similarly, a positive example is derived by setting

$$\mathbf{q} = (x - d + o_{\text{pos}}, y), \tag{5}$$

where o_{pos} is chosen randomly from the set $\{-P_{\text{hi}}, \ldots, P_{\text{hi}}\}$. The reason for including o_{pos}, instead of setting it to zero, has to do with the stereo method used later on. In particular, we found that cross-based cost aggregation performs better when the network assigns low matching costs to good matches as well as near matches. N_{lo}, N_{hi}, P_{hi}, and the size of the image patches n are hyperparameters of the method.

3.2. Network architecture

The architecture we used is depicted in Figure 2. The network consists of eight layers, $L1$ through $L8$. The first layer is convolutional, while all other layers are fully-connected. The inputs to the network are two 9×9 gray image patches. The first convolutional layer consists of 32 kernels of size $5 \times 5 \times 1$. Layers $L2$ and $L3$ are fully-connected with 200 neurons each. After $L3$ the two 200 dimensional vectors are concatenated into a 400 dimensional vector and passed through four fully-connected layers, $L4$

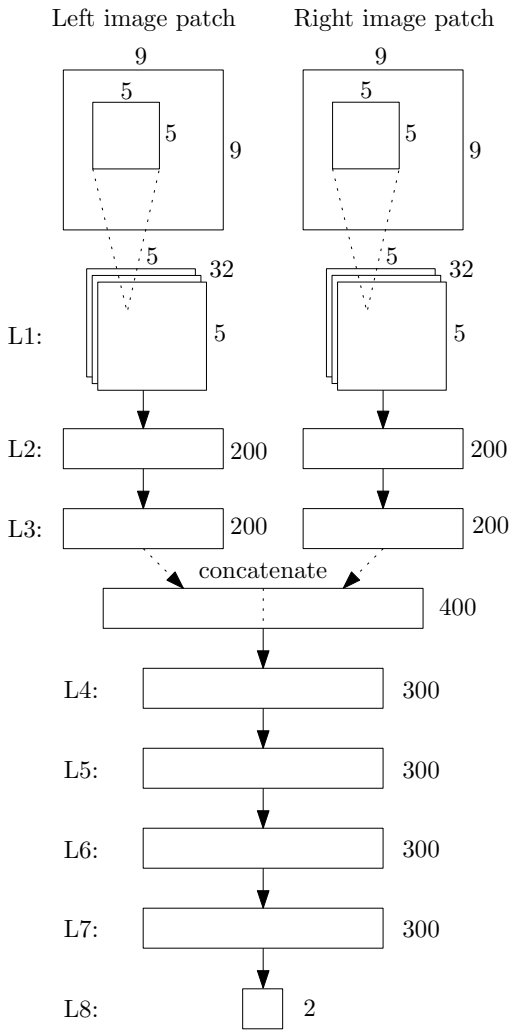

Figure 2. The architecture of our convolutional neural network.

through $L7$, with 300 neurons each. The final layer, $L8$, projects the output to two real numbers that are fed through a softmax function, producing a distribution over the two classes (good match and bad match). The weights in $L1$, $L2$, and $L3$ of the networks for the left and right image patch are tied. Rectified linear units follow each layer, except $L8$. We did not use pooling in our architecture. The network contains almost 600 thousand parameters. The architecture is appropriate for gray images, but can easily be extended to handle RGB images by learning $5 \times 5 \times 3$, instead of $5 \times 5 \times 1$ filters in $L1$. The best hyperparameters of the network (such as the number of layers, the number of neurons in each layer, and the size of input patches) will differ from one dataset to another. We chose this architecture because it performed well on the KITTI stereo dataset.

3.3. Matching cost

The matching cost $C_{\text{CNN}}(\mathbf{p}, d)$ is computed directly from the output of the network:

$$C_{\text{CNN}}(\mathbf{p}, d) = f_{\text{neg}}(< \mathcal{P}^L_{9\times 9}(\mathbf{p}), \mathcal{P}^R_{9\times 9}(\mathbf{pd}) >), \quad (6)$$

where $f_{\text{neg}}(< \mathcal{P}^L, \mathcal{P}^R >)$ is the output of the network for the negative class when run on input patches $\mathcal{P}^L$ and $\mathcal{P}^R$.

Naively, we would have to perform the forward pass for each image location $\mathbf{p}$ and each disparity d under consideration. The following three implementation details kept the runtime manageable:

1. The output of layers $L1$, $L2$, and $L3$ need to be computed only once per location $\mathbf{p}$ and need not be recomputed for every disparity d.

2. The output of $L3$ can be computed for all locations in a single forward pass by feeding the network full-resolution images, instead of 9×9 image patches. To achieve this, we apply layers $L2$ and $L3$ convolutionally—layer $L2$ with filters of size $5\times5\times32$ and layer $L3$ with filters of size $1 \times 1 \times 200$, both outputting 200 feature maps.

3. Similarly, $L4$ through $L8$ can be replaced with convolutional filters of size 1×1 in order to compute the output of all locations in a single forward pass. Unfortunately, we still have to perform the forward pass for each disparity under consideration.

4. Stereo method

In order to meaningfully evaluate the matching cost, we need to pair it with a stereo method. The stereo method we used was influenced by Mei et al. [11].

4.1. Cross-based cost aggregation

Information from neighboring pixels can be combined by averaging the matching cost over a fixed window. This approach fails near depth discontinuities where the assumption of constant depth within a window is violated. We might prefer a method that adaptively selects the neighborhood for each pixel so that support is collected only from pixels with similar disparities. In cross-based cost aggregation [21] we build a local neighborhood around each location comprising pixels with similar image intensity values.

Cross-based cost aggregation begins by constructing an upright cross at each position. The left arm $\mathbf{p}_l$ at position $\mathbf{p}$ extends left as long as the following two conditions hold:

- $|I(\mathbf{p}) - I(\mathbf{p}_l)| < \tau$. The absolute difference in image intensities at positions $\mathbf{p}$ and $\mathbf{p}_l$ is smaller than τ.

- $\|\mathbf{p} - \mathbf{p}_l\| < \eta$. The horizontal distance (or vertical distance, in case of top and bottom arms) between $\mathbf{p}$ and $\mathbf{p}_l$ is less than η.

The right, bottom, and top arms are constructed analogously. Once the four arms are known, we can define the support region $U(\mathbf{p})$ as the union of horizontal arms of all positions $\mathbf{q}$ laying on $\mathbf{p}$'s vertical arm (see Figure 3). Zhang

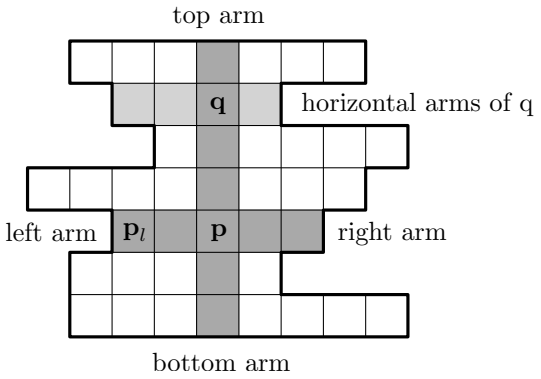

Figure 3. The support region for position $\mathbf{p}$, is the union of horizontal arms of all positions $\mathbf{q}$ on $\mathbf{p}$'s vertical arm.

et al. [21] suggest that aggregation should consider the support regions of both images in a stereo pair. Let U^L and U^R denote the support regions in the left and right image. We define the combined support region U_d as

$$U_d(\mathbf{p}) = \{\mathbf{q} | \mathbf{q} \in U^L(\mathbf{p}), \mathbf{qd} \in U^R(\mathbf{pd})\}. \tag{7}$$

The matching cost is averaged over the combined support region:

$$C^0_{\text{CBCA}}(\mathbf{p}, d) = C_{\text{CNN}}(\mathbf{p}, d), \tag{8}$$

$$C^i_{\text{CBCA}}(\mathbf{p}, d) = \frac{1}{|U_d(\mathbf{p})|} \sum_{\mathbf{q} \in U_d(\mathbf{p})} C^{i-1}_{\text{CBCA}}(\mathbf{q}, d), \tag{9}$$

where i is the iteration number. We repeat the averaging four times; the output of cross-based cost aggregation is C^4_{CBCA}.

4.2. Semiglobal matching

We refine the matching cost by enforcing smoothness constraints on the disparity image. Following Hirschmuller [4], we define an energy function $E(D)$ that depends on the disparity image D:

$$E(D) = \sum_{\mathbf{p}} \Bigg(C^4_{\text{CBCA}}(\mathbf{p}, D(\mathbf{p})) + \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{p}} P_1 \times 1\{|D(\mathbf{p}) - D(\mathbf{q})| = 1\} + \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{p}} P_2 \times 1\{|D(\mathbf{p}) - D(\mathbf{q})| > 1\} \Bigg), \tag{10}$$

where $1\{\cdot\}$ denotes the indicator function. The first term penalizes disparities $D(\mathbf{p})$ with high matching costs. The second term adds a penalty P_1 when the disparity of neighboring pixels differ by one. The third term adds a larger penalty P_2 when the neighboring disparities differ by more than one. Rather than minimizing $E(D)$ in 2D, we perform the minimization in a single direction with dynamic programming. This solution introduces unwanted streaking effects, since there is no incentive to make the disparity image smooth in the directions we are not optimizing over. In semiglobal matching we minimize the energy $E(D)$ in many directions and average to obtain the final result. Although Hirschmuller [4] suggests choosing sixteen direction, we only optimized along the two horizontal and the two vertical directions; adding the diagonal directions did not improve the accuracy of our system.

To minimize $E(D)$ in direction $\mathbf{r}$, we define a matching cost $C_\mathbf{r}(\mathbf{p}, d)$ with the following recurrence relation:

$$C_\mathbf{r}(\mathbf{p}, d) = C^4_{\text{CBCA}}(\mathbf{p}, d) - \min_k C_r(\mathbf{p} - \mathbf{r}, k) + \min \Big\{ C_r(\mathbf{p} - \mathbf{r}, d), C_r(\mathbf{p} - \mathbf{r}, d-1) + P_1, C_r(\mathbf{p} - \mathbf{r}, d+1) + P_1, \min_k C_\mathbf{r}(\mathbf{p} - \mathbf{r}, k) + P_2 \Big\}. \tag{11}$$

The second term is included to prevent values of $C_\mathbf{r}(\mathbf{p}, d)$ from growing too large and does not affect the optimal disparity map. The parameters P_1 and P_2 are set according to the image gradient so that jumps in disparity coincide with edges in the image. Let $D_1 = |I^L(\mathbf{p}) - I^L(\mathbf{p} - \mathbf{r})|$ and $D_2 = |I^R(\mathbf{pd}) - I^R(\mathbf{pd} - \mathbf{r})|$. We set P_1 and P_2 according to the following rules:

$$\begin{array}{llll} P_1 = \Pi_1, & P_2 = \Pi_2 & \text{if } D_1 < \tau_{\text{SO}}, D_2 < \tau_{\text{SO}}, \\ P_1 = \Pi_1/4, & P_2 = \Pi_2/4 & \text{if } D_1 \geq \tau_{\text{SO}}, D_2 < \tau_{\text{SO}}, \\ P_1 = \Pi_1/4, & P_2 = \Pi_2/4 & \text{if } D_1 < \tau_{\text{SO}}, D_2 \geq \tau_{\text{SO}}, \\ P_1 = \Pi_1/10, & P_2 = \Pi_2/10 & \text{if } D_1 \geq \tau_{\text{SO}}, D_2 \geq \tau_{\text{SO}}; \end{array}$$

where Π_1, Π_2, and τ_{SO} are hyperparameters. The value of P_1 is halved when minimizing in the vertical directions. The final cost $C_{\text{SGM}}(\mathbf{p}, d)$ is computed by taking the average across all four directions:

$$C_{\text{SGM}}(\mathbf{p}, d) = \frac{1}{4} \sum_{\mathbf{r}} C_\mathbf{r}(\mathbf{p}, d). \tag{12}$$

After semiglobal matching we repeat cross-based cost aggregation, as described in the previous section.

4.3. Computing the disparity image

The disparity image D is computed by the winner-take-all strategy, *i.e.* by finding the disparity d that minimizes $C(\mathbf{p}, d)$,

$$D(\mathbf{p}) = \underset{d}{\operatorname{argmin}}\, C(\mathbf{p}, d). \tag{13}$$

4.3.1 Interpolation

Let D^L denote the disparity map obtained by treating the left image as the reference image—this was the case so far, *i.e.* $D^L(\mathbf{p}) = D(\mathbf{p})$—and let D^R denote the disparity map obtained by treating the right image as the reference image. Both D^L and D^R contain errors in occluded regions. We attempt to detect these errors by performing a left-right consistency check. We label each position $\mathbf{p}$ as either

$$\begin{array}{ll} correct & \text{if } |d - D^R(\mathbf{pd})| \leq 1 \text{ for } d = D^L(\mathbf{p}), \\ mismatch & \text{if } |d - D^R(\mathbf{pd})| \leq 1 \text{ for any other } d, \\ occlusion & \text{otherwise.} \end{array}$$

For positions marked as *occlusion*, we want the new disparity value to come from the background. We interpolate by moving left until we find a position labeled *correct* and use its value. For positions marked as *mismatch*, we find the nearest *correct* pixels in 16 different directions and use the median of their disparities for interpolation. We refer to the interpolated disparity map as D_{INT}.

4.3.2 Subpixel enhancement

Subpixel enhancement provides an easy way to increase the resolution of a stereo algorithm. We fit a quadratic curve through the neighboring costs to obtain a new disparity image:

$$D_{\text{SE}}(\mathbf{p}) = d - \frac{C_+ - C_-}{2(C_+ - 2C + C_-)}, \tag{14}$$

where $d = D_{\text{INT}}(\mathbf{p})$, $C_- = C_{\text{SGM}}(\mathbf{p}, d-1)$, $C = C_{\text{SGM}}(\mathbf{p}, d)$, and $C_+ = C_{\text{SGM}}(\mathbf{p}, d+1)$.

4.3.3 Refinement

The size of the disparity image D_{SE} is smaller than the size of the original image, due to the bordering effects of convolution. The disparity image is enlarged to match the size of the input by copying the disparities of the border pixels. We proceed by applying a 5×5 median filter and the following bilateral filter:

$$D_{\text{BF}}(\mathbf{p}) = \frac{1}{W(\mathbf{p})} \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{p}} D_{\text{SE}}(\mathbf{q}) \cdot g(\|\mathbf{p} - \mathbf{q}\|) \cdot 1\{|I^L(\mathbf{p}) - I^L(\mathbf{q})| < \tau_{\text{BF}}\}, \tag{15}$$

where $g(x)$ is the probability density function of a zero mean normal distribution with standard deviation σ and $W(\mathbf{p})$ is the normalizing constant:

$$W(\mathbf{p}) = \sum_{\mathbf{q} \in \mathcal{N}_\mathbf{p}} g(\|\mathbf{p} - \mathbf{q}\|) \cdot 1\{|I^L(\mathbf{p}) - I^L(\mathbf{q})| < \tau_{\text{BF}}\}. \tag{16}$$

τ_{BF} and σ are hyperparameters. D_{BF} is the final output of our stereo method.

5. Experimental results

We evaluate our method on the KITTI stereo dataset, because of its large training set size required to learn the weights of the convolutional neural network.

5.1. KITTI stereo dataset

The KITTI stereo dataset [2] is a collection of gray image pairs taken from two video cameras mounted on the roof of a car, roughly 54 centimeters apart. The images are recorded while driving in and around the city of Karlsruhe, in sunny and cloudy weather, at daytime. The dataset comprises 194 training and 195 test image pairs at resolution 1240×376. Each image pair is rectified, *i.e.* transformed in such a way that an object appears on the same vertical position in both images. A rotating laser scanner, mounted behind the left camera, provides ground truth depth. The true disparities for the test set are withheld and an online leaderboard[1] is provided where researchers can evaluate their method on the test set. Submissions are allowed only once every three days. The goal of the KITTI stereo dataset is to predict the disparity for each pixel on the left image. Error is measured by the percentage of pixels where the true disparity and the predicted disparity differ by more than three pixels. Translated into depth, this means that, for example, the error tolerance is ± 3 centimeters for objects 2 meters from the camera and ± 80 centimeters for objects 10 meters from the camera.

5.2. Details of learning

We train the network using stochastic gradient descent to minimize the cross-entropy loss. The batch size was set to 128. We trained for 16 epochs with the learning rate initially set to 0.01 and decreased by a factor of 10 on the 12$^{\text{th}}$ and 15$^{\text{th}}$ iteration. We shuffle the training examples prior to learning. From the 194 training image pairs we extracted 45 million examples. Half belonging to the positive class; half to the negative class. We preprocessed each image by subtracting the mean and dividing by the standard deviation of its pixel intensity values. The stereo method is implemented in CUDA, while the network training is done with

[1] http://www.cvlibs.net/datasets/kitti/eval_stereo_flow.php?benchmark=stereo

the Torch7 environment [1]. The hyperparameters of the stereo method were:

$$N_{\text{lo}} = 4, \quad \eta = 4, \quad \Pi_1 = 1, \quad \sigma = 5.656,$$
$$N_{\text{hi}} = 8, \quad \tau = 0.0442, \quad \Pi_2 = 32, \quad \tau_{\text{BF}} = 5,$$
$$P_{\text{hi}} = 1, \quad \tau_{\text{SO}} = 0.0625.$$

5.3. Results

Our method achieves an error rate of 2.61 % on the KITTI stereo test set and is currently ranked first on the online leaderboard. Table 1 compares the error rates of the best performing stereo algorithms on this dataset.

Rank	Method		Error
1	MC-CNN	This paper	2.61 %
2	SPS-StFl	Yamaguchi et al. [20]	2.83 %
3	VC-SF	Vogel et al. [16]	3.05 %
4	CoP	Anonymous submission	3.30 %
5	SPS-St	Yamaguchi et al. [20]	3.39 %
6	PCBP-SS	Yamaguchi et al. [19]	3.40 %
7	DDS-SS	Anonymous submission	3.83 %
8	StereoSLIC	Yamaguchi et al. [19]	3.92 %
9	PR-Sf+E	Vogel et al. [17]	4.02 %
10	PCBP	Yamaguchi et al. [18]	4.04 %

Table 1. The KITTI stereo leaderboard as it stands in November 2014.

A selected set of examples, together with predictions from our method, are shown in Figure 5.

5.4. Runtime

We measure the runtime of our implementation on a computer with a Nvidia GeForce GTX Titan GPU. Training takes 5 hours. Predicting a single image pair takes 100 seconds. It is evident from Table 2 that the majority of time during prediction is spent in the forward pass of the convolutional neural network.

Component	Runtime
Convolutional neural network	95 s
Semiglobal matching	3 s
Cross-based cost aggregation	2 s
Everything else	0.03 s

Table 2. Time required for prediction of each component.

5.5. Training set size

We would like to know if more training data would lead to a better stereo method. To answer this question, we train our convolutional neural network on many instances of the KITTI stereo dataset while varying the training set size. The results of the experiment are depicted in Figure 4. We observe an almost linear relationship between the training set size and error on the test set. These results imply that our method will improve as larger datasets become available in the future.

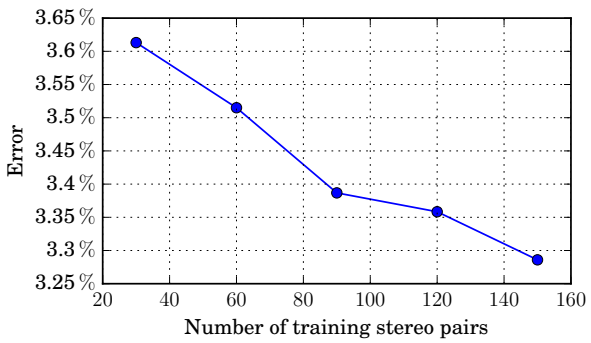

Figure 4. The error on the test set as a function of the number of stereo pairs in the training set.

6. Conclusion

Our result on the KITTI stereo dataset seems to suggest that convolutional neural networks are a good fit for computing the stereo matching cost. Training on bigger datasets will reduce the error rate even further. Using supervised learning in the stereo method itself could also be beneficial. Our method is not yet suitable for real-time applications such as robot navigation. Future work will focus on improving the network's runtime performance.

References

[1] Collobert, R., Kavukcuoglu, K., and Farabet, C. (2011). Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, number EPFL-CONF-192376.

[2] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)*.

[3] Haeusler, R., Nair, R., and Kondermann, D. (2013). Ensemble learning for confidence measures in stereo vision. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 305–312. IEEE.

[4] Hirschmuller, H. (2008). Stereo processing by semiglobal matching and mutual information. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2):328–341.

[5] Hirschmuller, H. and Scharstein, D. (2009). Evaluation of stereo matching costs on images with radiometric

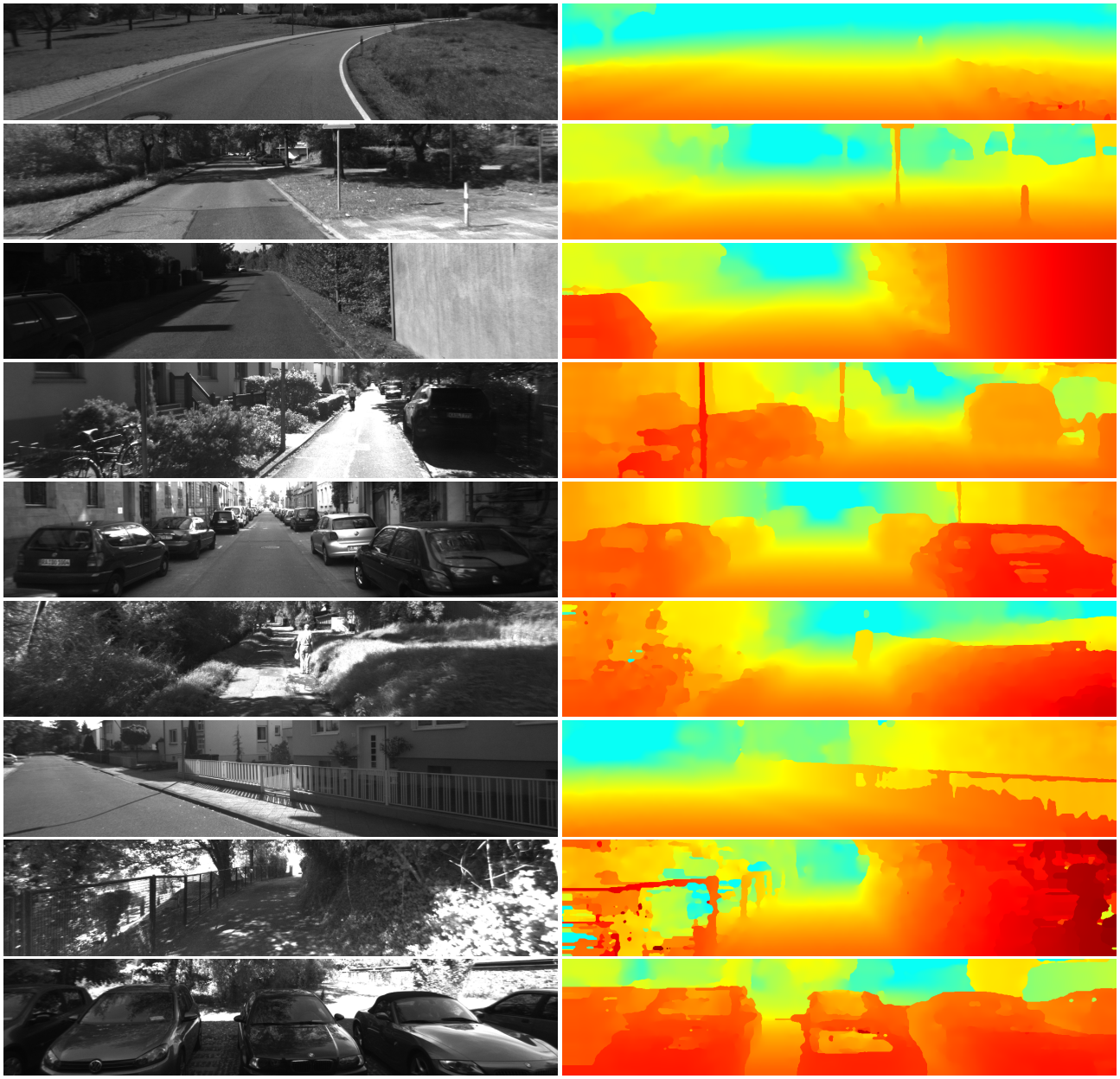

Figure 5. The left column displays the left input image, while the right column displays the output of our stereo method. Examples are sorted by difficulty, with easy examples appearing at the top. Some of the difficulties include reflective surfaces, occlusions, as well as regions with many jumps in disparity, *e.g*. fences and shrubbery. The examples towards the bottom were selected to highlight the flaws in our method and to demonstrate the inherent difficulties of stereo matching on real-world images.

differences. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(9):1582–1599.

[6] Kong, D. and Tao, H. (2004). A method for learning matching errors for stereo computation. In *BMVC*, pages 1–10.

[7] Kong, D. and Tao, H. (2006). Stereo matching via learning multiple experts behaviors. In *BMVC*, pages 97–106.

[8] Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1106–1114.

[9] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P.

(1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

[10] Li, Y. and Huttenlocher, D. P. (2008). Learning for stereo vision using the structured support vector machine. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE.

[11] Mei, X., Sun, X., Zhou, M., Wang, H., Zhang, X., et al. (2011). On building an accurate stereo matching system on graphics hardware. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 467–474. IEEE.

[12] Peris, M., Maki, A., Martull, S., Ohkawa, Y., and Fukui, K. (2012). Towards a simulation driven stereo vision system. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 1038–1042. IEEE.

[13] Scharstein, D. and Pal, C. (2007). Learning conditional random fields for stereo. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE.

[14] Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42.

[15] Spyropoulos, A., Komodakis, N., and Mordohai, P. (2014). Learning to detect ground control points for improving the accuracy of stereo matching. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1621–1628. IEEE.

[16] Vogel, C., Roth, S., and Schindler, K. (2014). View-consistent 3d scene flow estimation over multiple frames. In *Computer Vision–ECCV 2014*, pages 263–278. Springer.

[17] Vogel, C., Schindler, K., and Roth, S. (2013). Piecewise rigid scene flow. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1377–1384. IEEE.

[18] Yamaguchi, K., Hazan, T., McAllester, D., and Urtasun, R. (2012). Continuous markov random fields for robust stereo estimation. In *Computer Vision–ECCV 2012*, pages 45–58. Springer.

[19] Yamaguchi, K., McAllester, D., and Urtasun, R. (2013). Robust monocular epipolar flow estimation. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 1862–1869. IEEE.

[20] Yamaguchi, K., McAllester, D., and Urtasun, R. (2014). Efficient joint segmentation, occlusion labeling, stereo and flow estimation. In *Computer Vision–ECCV 2014*, pages 756–771. Springer.

[21] Zhang, K., Lu, J., and Lafruit, G. (2009). Cross-based local stereo matching using orthogonal integral images. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(7):1073–1079.

[22] Zhang, L. and Seitz, S. M. (2007). Estimating optimal parameters for mrf stereo from a single image pair. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(2):331–342.