End-to-End Training of Hybrid CNN-CRF Models for Stereo Supplementary Material

A. Technical Details

A.1. Approximate Subgradient

Here we proof Proposition 4.1, restated below for convenience.

Proposition 4.1. Let $\bar{x}^1$ and $\bar{x}^2$ be minimizers of horizontal and vertical chain subproblems in (16) for a given λ. Let $\Omega_{\neq}$ be a subset of nodes for which $\bar{x}_i^1 \neq \bar{x}_i^2$. Then a subgradient g of the loss upper bound (17) w.r.t. $f_{\mathcal{V}} = (f_i \,|\, i \in \mathcal{V})$ has the following expression in components

$$g_i(k) = \big(\delta(x^*) - \delta(\bar{x}^1)\big)_i(k) + \sum_{j \in \Omega_{\neq}} \big(J_{ij}(k, \bar{x}_i^2) - J_{ij}(k, \bar{x}_i^1)\big), \tag{19}$$

Proof. The loss upper bound (17) involves the minimum over x^1, x^2 as well as many minima inside the dynamic programming defining λ. A subgradient can be obtained by fixing particular minimizers in all these steps and evaluating the gradient of the resulting function. It follows that a subgradient of the point-wise minimum of $(\bar{f}^1 + \lambda)(x^1) + (\bar{f}^2 - \lambda)(x^2)$ over x^1, x^2 can be chosen as $g =$

$$\nabla_{f_{\mathcal{V}}}(\bar{f}^1(\bar{x}^1) + \bar{f}^2(\bar{x}^2)) + \nabla_{\lambda}(\lambda(\bar{x}^1) - \lambda(\bar{x}^2))J, \tag{21}$$

where $J_{i,j}(k,l)$ is a sub-Jacobian matching $\frac{d\lambda_j(l)}{df_i(k)}$ for the directions $df_{\mathcal{V}}$ such that $\lambda(f + df_{\mathcal{V}})$ has the same minimizers inside dynamic programming as $\lambda(f)$.

In the first part of the expression (21), the pairwise components and the loss $l(\bar{x}^1, x^*)$ do not depend on f_i and may be dropped, leaving only $(\nabla_{f_{\mathcal{V}}} \sum_{j \in \mathcal{V}} f_j(\bar{x}_j^1))_i = \delta(\bar{x}^1)_i$.

Let h denote the second expression in (21). Its component $h_i(k)$ expands as

$$h_i(k) = \sum_{j \in \mathcal{V}} \sum_{l \in \mathcal{L}} \frac{\partial}{\partial \lambda_j(l)} (\lambda_j(\bar{x}_j^1) - \lambda_j(\bar{x}_j^2)) J_{ij}(k,l) \tag{22a}$$

$$= \sum_{j \in \Omega_{\neq}} \sum_{l \in \mathcal{L}} ([\![\bar{x}_j^1 = l]\!] - [\![\bar{x}_j^2 = l]\!]) J_{ij}(k,l) \tag{22b}$$

$$= \sum_{j \in \Omega_{\neq}} (J_{ij}(k, x_j^1) - J_{ij}(k, x_j^2)). \tag{22c}$$

□

Our intuition to neglect the sum (22c) is as follows. We expect that variation of f_i for a pixel i far enough from $j \in \Omega_{\neq}$ will not have a significant effect on λ_j and thus J_{ij} will be small over $\Omega_{\neq}$.

B. Training insights

We train our full joint model gradually as explained in § 4 in the main paper. To give more insights on how the joint training evolves until we get our final parameters, we show a training plot in Fig. B.1. This plot shows the evolution of the *bad4* error on the Middlebury dataset for our 7-layer model. We can identify three key steps during the training procedure. (A) shows the training of our *Unary-CNN* using ML § 4.1. In (B) we add the CRF with contrast-sensitive weights with an optimal choice of parameters $(\alpha, \beta, P_1, P_2)$. Finally, in (C) we jointly optimization the complete model §§ 4.2 and 4.3. Observe that the gap between training and validation errors is significantly smaller in (C).

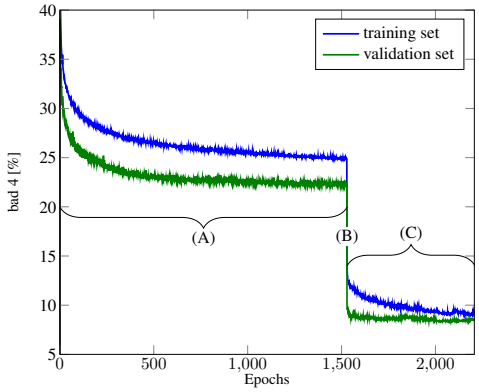

Figure B.1: Performance w.r.t. the real objective for key complexity steps of our model during training.

C. Additional Experiments

C.1. Timing

In Table C.1 we report the runtime of individual components of our method for different image sizes and number of labels (=disparties). All experiments are carried out on a Linux PC with a Intel Core i7-5820K CPU with 3.30GHz and a NVidia GTX TitanX using CUDA 8.0. For Kitti 2015, the image size is 1242×375. For Middlebury V3 we selected the *Jadeplant* data set with *half* resolution, leading to an image size of 1318×994. We observe that with a constant number of layers in the Unary CNN and disparity range, the runtime depends linearly on the number of pixels in the input images. Correlation and CRF layer also depend on the number of estimated disparities, where we report numbers using 128 and 256 disparities.

C.2. Sublabel Enhancement

A drawback of our CRF method based on dynamic programming is the discrete nature of the solution. For some benchmarks like Middlebury the discretization artifacts negatively influence the quantitative performance. Therefore, most related stereo methods perform some kind of sub-label refinement (*e.g.* [28, 55]). For the submission to online benchmarks we deliberately chose to *discard* any form of non-trainable post-processing. However, we performed additional experiments with fitting a quadratic function to

Component	# Disp.	Kitti 2015 0.4 MP	Middlebury 1.3 MP	Real-Time 0.3 MP
Input processing		7.58	6.40	6.02
Pairwise CNN		21.12	59.46	13.75
Unary CNN		262.48	664.19	62.54
Correlation	128	154.86	437.02	46.70
Correlation	256	286.87	802.86	–
CRF	128	309.48	883.57	155.85
CRF	256	605.35	1739.34	–
Total	128	755.52	2050.64	284.86
Total	256	1183.40	3272.25	–

Table C.1: Timing experiments for 7 layer CNN and 5 CRF iterations (3 layer and 4 iterations for **Real-Time**). Runtimes in ms.

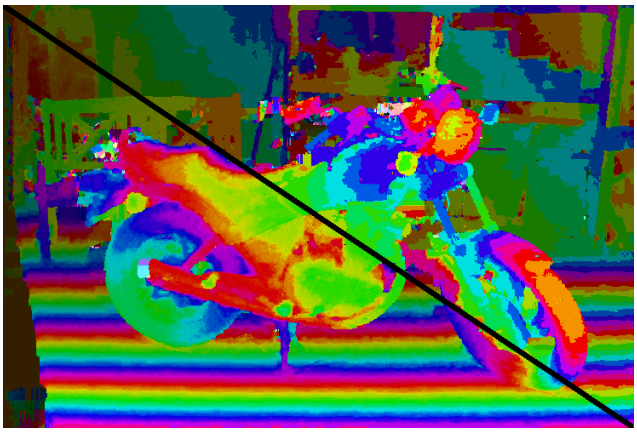

Figure C.1: Qualitative comparison on *Motorcycle* of discrete (upper-right) and sublabel enhanced (bottom-left) solution. Note how smooth the transitions are in the sublabel enhanced region (*e.g.* at the floor or the rear wheel).

the output cost volume of the CRF method around the discrete solution. The refined disparity is then given by

$$d_{se} = d + \frac{C(d-h) - C(d+h)}{2(C(d+h) - 2C(d) + C(d-h))} \tag{23}$$

where $C(d)$ is the cost of disparity d. A qualitative experiment on the *Motorcycle* image of Middlebury stereo can be seen in Fig. C.1. Quantitative experiments have been conducted on both Kitti 2015 and Middlebury and will be reported in the follow sections (columns **w. ref.** in Tables C.2 and C.3). Again, in the main paper and in the submitted images we always report the performance of the *discrete* solution in order to keep the method pure.

C.3. Middlebury Stereo v3

In this section we report a complete overview of all tested variants of our proposed hybrid CNN-CRF model on the stereo benchmark of Middlebury Stereo v3. We report the mean error (error metric *percent of non-occluded pixels with an error bigger 4 pixels*). All results are calculated on quarter resolution and upsampled to the original image size. We present the results in Fig. C.2 and Table C.2. Note, how the quality increases when we add more parameters and therefore allow a more general model (visualized from left to right in Fig. C.2. The last row shows the *Vintage* image, where our model produces a rather high error. The reason for that lies in the (almost) completely untextured region in the top-left corner. Our full model is able to recover some disparities in this region, but not all. A very interesting byproduct visible in Fig. C.2 concerns our small 3-layer model. Visually, one can hardly see any difference to the deeper 7-layer model, when our models are full jointly trained. Hence, this small model is suited very well for a real-time application.

Additionally, we compared to the performance of the model learned on Kitti, denoted Kitti-CNN in Table C.2. The performance is inferior, which means that the model trained on Kitti does not generalize well to Middlebury. Generalizing from Middlebury to Kitti, on the other hand is much better, as discussed in the next section.

Method	w/o. ref.	w. ref.
CNN3	23.89	-
CNN3+CRF	11.18	10.50
CNN3 Joint	9.48	8.75
CNN3 PW+Joint	9.45	8.70
CNN7	18.58	-
CNN7+CRF	9.35	8.68
CNN7 Joint	8.05	7.32
CNN7 PW+Joint	7.88	7.09
Kitti-CNN	15.22	14.43

Table C.2: Comparison of differently trained models and their performance on the official training images of the Middlebury V3 stereo benchmark. The results are given in % of pixels farther away than 4 disparities from the ground-truth on all pixels.

C.4. Kitti 2015

In this section we report a complete overview of all tested variants of our proposed hybrid CNN-CRF model on the stereo benchmark of KITTI 2015. We report the mean error (official error metric *percent of pixel with an error bigger 3 pixels*) on the complete design set. Table C.3 shows a performance overview of our models. In the last row of Table C.3 we apply our best performing model on Middlebury to the Kitti design set. Interestingly, the performance decreases only by $\approx 1.5\%$ on all pixels. This experiment indicates, that our models generalize well to the scenes of the Kitti benchmark.

Due to lack of space in the main paper, we could only show a few qualitative results of the submitted method. In Fig. C.4 we show additional results, more of which can be viewed online.

Looking at Kitti results in more detail, we observe that most of the errors happen in either occluded regions or due to a fattened ground-truth. Since we train edge-weights to courage label-jumps at strong object boundaries, our model yields very sharp results. It is these sharp edges in our solution which introduce some errors on the benchmark, even when our prediction is correct. Fig. C.3 shows some examples on the *test* set (provided by the online submission system).

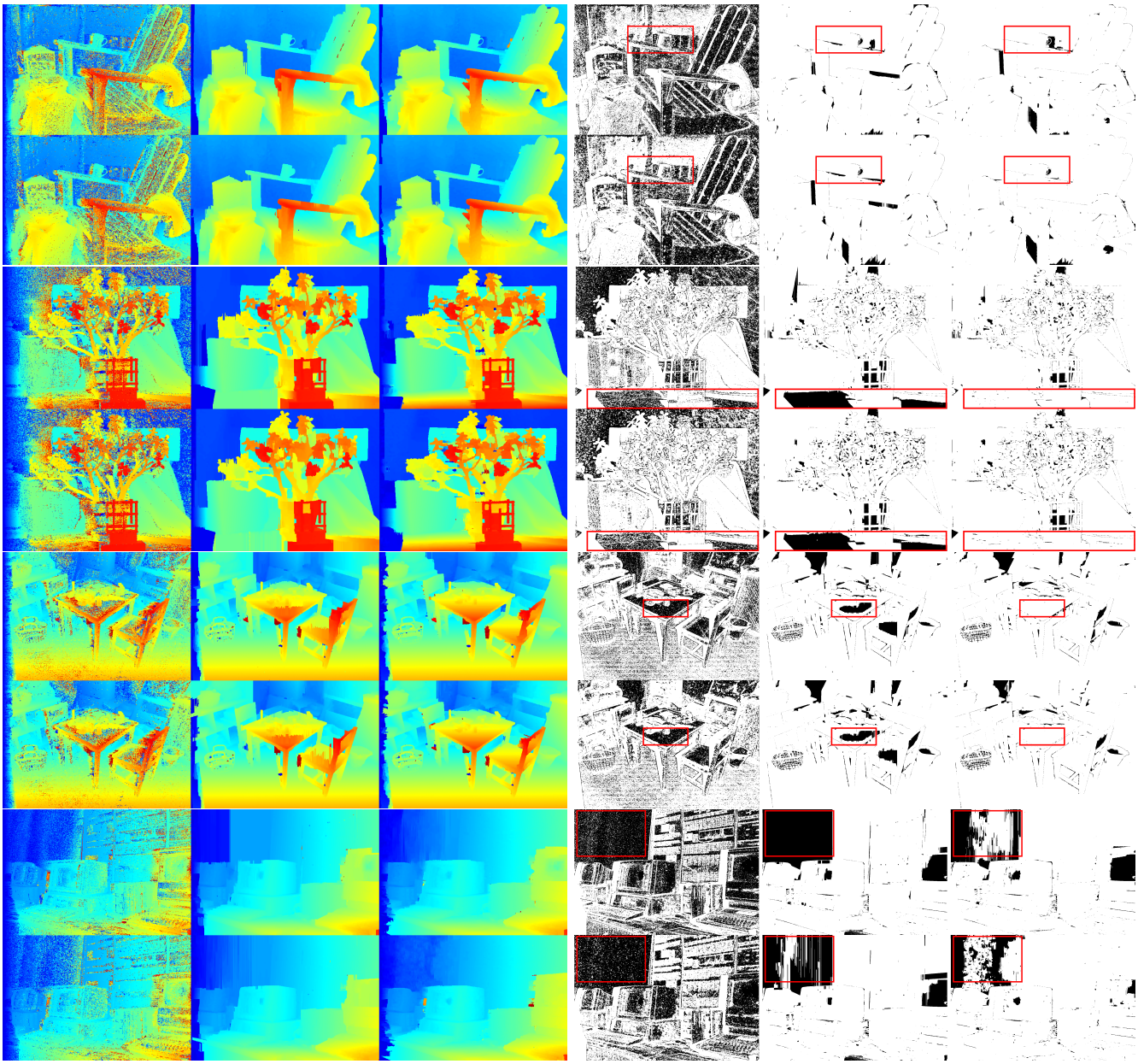

Figure C.2: Qualitative comparison of our models on Middlebury. For each image, the first row shows our 3-layer model and the second row shows the result of our 7-layer model. The first column shows out *Unary-CNN* with argmax desicion rule, the second column *CNNx+CRF* and the third column shows the result of *CNNx+CRF+Joint+PW*. The remaining columns show the respective error-plots for the different models, where white indicates correct and black indicates wrong disparities. The red boxes highlight differences between our models. Disparity maps are color-coded from blue (small disparities) to red (large disparities).

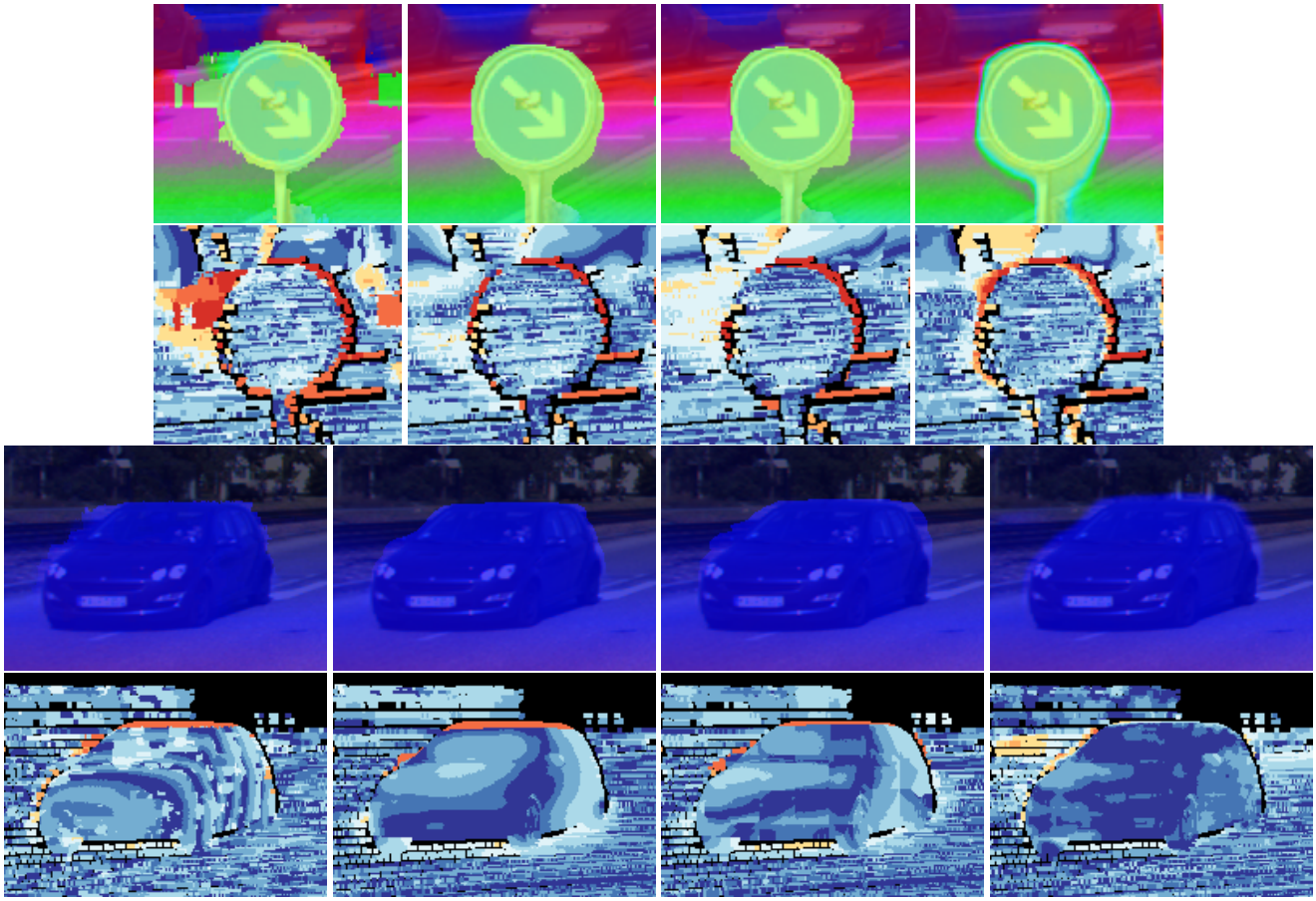

Figure C.3: Error comparison on magnified parts of Kitti 2015 test images: The first and third row show the color-coded disparity map of *Ours*, *MC-CNN*, *ContentCNN* and *DispNetC*. The second and last row show the corresponding error-plots, where shades of blue mean correct and shades of orange mean wrong. Note, how our model accurately follows object boundaries, whereas all other approaches fatten the object. Nevertheless, in terms of correct or wrong we make more wrong predictions, because the ground-truth seems to be fattened as well.

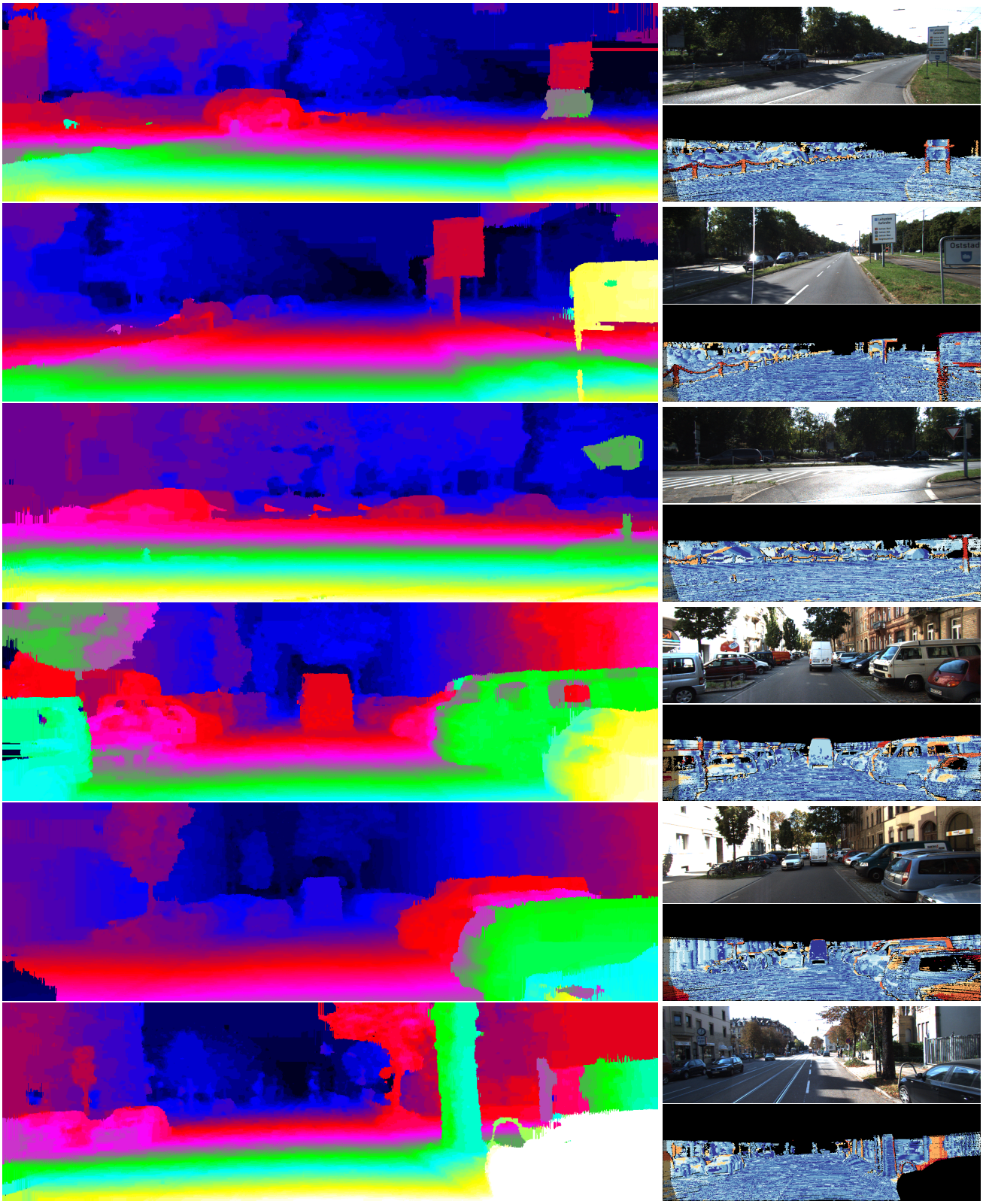

Figure C.4: Qualitative comparison on the test set of KITTI 2015.

Method	w/o. ref.		w. ref.	
	all	**non occ.**	**all**	**non occ.**
CNN3	29.58	28.38	-	-
CNN3+CRF	7.88	6.33	7.77	6.22
CNN3 Joint	7.66	6.11	7.57	6.02
CNN3 PW+Joint	6.25	4.75	6.14	4.65
CNN7	14.55	13.08	-	-
CNN7+CRF	5.85	4.79	5.76	4.70
CNN7 Joint	5.98	4.60	5.89	4.50
CNN7 PW+Joint	5.25	4.04	5.18	3.96
[55]+CRF	6.10	4.45	5.74	4.08
[28]+CRF	5.89	4.31	5.81	4.21
[55]	15.02	13.56	-	-
[28]	7.54	5.99	-	-
MB-CNN	6.82	5.35	6.69	5.21

Table C.3: Comparison of differently trained models and their performance on the design set images of the KITTI 2015 stereo benchmark. The results are given in % of pixels farther away than 3 disparities from the ground-truth on all pixels.