Belief Propagation Reloaded: Learning BP-Layers for Labeling Problems
Supplementary Material

Contents

A. Differentiable TRW and TBCA algorithms

Here we consider two other inference methods that have similar properties of long-range spatial propagation and parallelization and can be implemented with same or similar subroutines. As they improve on the issues of BP in loopy graphs, this makes them potential candidates for drop-in replacement of our sweep BP-layer.

Tree Re-weighted BP Wainwright et al. [48] proposed a correction to BP, which turns it into a variational inference algorithm optimizing the dual of the LP relaxation. Suppose that we are given an edge-disjoint decomposition of the graph into trees. For our models it is convenient to take horizontal and vertical chain subproblems. The TRW-T algorithm [48] can be implemented as proposed in Algorithm 5. In this representation we keep the decomposition into subproblems explicitly and messages are encapsulated in the computation of max-marginals. This is in order to reuse the same subroutines we already have for BP-Layer. An explicit form of updates in terms of messages only which reveals the similarity to loopy belief propagation with weighting coefficients can be also given [48]. This algorithm is not guaranteed to be monotonous because it does block-coordinate ascent steps in multiple blocks in parallel. However thanks to parallelization it is fast to compute (in particular on a GPU), incorporates long-range interactions and avoids the over-counting problems associated with loopy BP [48].

Tree Block Coordinate Ascent The TBCA method [41] is an inference algorithm optimizing the dual of the LP relaxation. It does so by a block-coordinate ascent in the variables associated with tree-structured subproblems. The variables are the same as the messages in BP. At each iteration a subtree $(\mathcal{V}', \mathcal{E}')$ from the graph is selected. For simplicity and

Algorithm 5: Tree Reweighted BP (TRW-T)

Input: CRF scores $g \in \mathbb{R}^{\mathcal{V}\times\mathcal{L}}$, $f \in \mathbb{R}^{\mathcal{E}\times\mathcal{L}^2}$;
Output: Beliefs $B \in \mathbb{R}^{\mathcal{V}\times\mathcal{L}}$;

$g^{\mathrm{h}} := g^{\mathrm{v}} := \frac{1}{2}g$;
for *iteration* $t = 1 \dots T$ **do**
  `/* Compute max-marginals: */`
  par. for *horizontal chain subgraphs* $(\mathcal{V}', \mathcal{E}')$ **do**
    $b^{\mathrm{h}}_{\mathcal{V}'} := \texttt{max\_marginals}(g^{\mathrm{h}}_{\mathcal{V}'}, f_{\mathcal{E}'})$;
  par. for *vertical chain subgraphs* $(\mathcal{V}', \mathcal{E}')$ **do**
    $b^{\mathrm{v}}_{\mathcal{V}'} := \texttt{max\_marginals}(g^{\mathrm{v}}_{\mathcal{V}'}, f_{\mathcal{E}'})$;
  `/* Enforce consistency: */`
  $b := (b^{\mathrm{h}} + b^{\mathrm{v}})$;
  $g^{\mathrm{h}} \mathrel{+}= (\frac{1}{2}b - b^{h})$;
  $g^{\mathrm{v}} \mathrel{+}= (\frac{1}{2}b - b^{v})$;
return Log-beliefs b;

ease of parallelization we will assume $(\mathcal{V}', \mathcal{E}')$ is a horizontal chain and consider it to be ordered from left to right. The following updates are performed on this chain:

- Compute the current reparametrized costs, excluding the messages from inside the chain:

$$a_i(s) = g_i(s) + \sum_{(i,j)\in\mathcal{E}\setminus\mathcal{E}'} m_{ji}(s) \forall i \in \mathcal{V}'. \tag{19}$$

- Compute the right messages m^{R} by DP in the direction R→L.
- Compute the left messages m^{L} by a *redistribution DP* (rDP) in the direction R→L.

We can write the rDP update equation [41] in the form

$$m^{\mathrm{L}}_{i+1}(t) := \max_s \Big(\tilde{g}_i(s) + r_i m^{\mathrm{L}}_i(s) + f_{i,i+1}(s,t)\Big), \tag{20}$$

where $\tilde{g}_i(s) = g_i(s) + (1 - r_i)m^{\mathrm{R}}_i(s)$ and $r_i \in [0, 1]$ are the redistribution coefficients. For $r = 1$, this recovers the regular dynamic programming. Similarly to DP, the update is linear and depend on the current maximizers that we record as $o_{i+1}(t)$. It differs from DP in two ways: i) it depends on the right messages, which we have taken into account by incorporating them to the unary costs in $\tilde{g}_i(s)$ and ii) there are coefficients r_i in the recursion. To handle the latter, we only need to modify Line 5 of Algorithm 3 to

$$z := đm_{i+1}(t) + r_{i+1} đg_{i+1}(t). \tag{21}$$

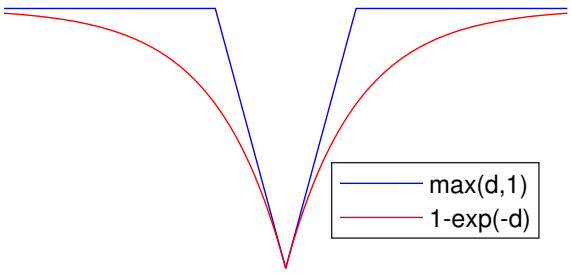

Figure A.1: The cost $-g_i(k)$ as a function of $d = \|f^0(i) - f^1(i-k)\|_1$ in our model is similar to robust costs $\max(d, \tau)$ previously used to better handle occlusions [24].

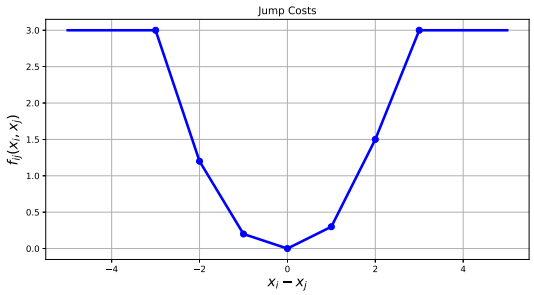

Figure B.1: Robust penalty function. Similar as the $P1, P2$ model in SGM, but with one additional learnable step. We allow to learn this function asymmetrically, because positive occlusions appear only on left-sided object boundaries.

It follows that we have defined the TBCA subproblem update with standard operations on tensors and the two new operations DP and rDP, for which we have shown efficient backprop methods. The TBCA method [41], when specialized to horizontal and vertical chains, would then alternate the above updates in parallel for all horizontal chains and then for all vertical chains. This method also achieves high parallelization efficiency and long-range propagation. Thanks to the redistribution mechanism it is also guaranteed to be monotonous. However, this monotonicity may slow down the information propagation, which can make it less suitable as a truncated inference technique in deep learning.

B. Implementation Details

We implemented our model in PyTorch[5] and the core of the BP-layer as a highly efficient CUDA kernel. For geometrical problems such as stereo and optical flow, we use a truncated compatibility function (see Fig. B.1). This is allows us to decrease the asymptotic runtime for K labels to $\mathcal{O}(K)$ and makes very efficient inference and training possible. For semantic segmentation we want to learn the full compatibility matrix. Nevertheless, since we learn the cost from any source label to any target label, the runtime is $\mathcal{O}(K^2)$ and thus quadratic in the number of labels. The practical impact on the runtime can be seen in Tables 1 and 4.

In our optimized CUDA implementation we utilize the following parallelization: All chains of the same direction as well as the chains in the opposing directions can be processed in parallel. Furthermore, the message-passing also parallelizes over the labels. For an image of size $N \times N$, assuming that the number of disparities also grows as $K = O(N)$, our implementation achieves parallelism of $O(N^2)$ while requiring sequential processing $O(N)$, which is an acceptable scaling with the image size. The backprop operation of the DP, has the same level of parallelism, which is important for large-scale learning. These implementations are connected as extensions to PyTorch, which allows them to be used in any computation graphs. In order to increase numerical accuracy, we also normalize the messages by subtracting the maximum over all labels on each step of DP. This does not affect the output beliefs, as the normalization cancels in the $\mathrm{softmax}$ operation.

We trained the model with the Adam optimizer [19] with a learning rate of $3 \cdot 10^{-3}$. We always start with a pre-training for 300k iterations on large scale synthetic data for stereo and optical flow to get a good initialization for our model. Finally, we fine-tune the pre-trained models on the target dataset for 1000 epochs using a learn-rate of 10^{-5}.

B.1. Runtime Analysis

We give a brief comparison of the runtime of the proposed BP-Layer and 3D convolutions here. Compared to other networks such as [4, 16, 53] we completely avoid the usage of the very costly 3D convolution layers. 3D convolution layers have a runtime of $\mathcal{O}(MNKCP^3)$ while our proposed BP-Layer has a runtime of $\mathcal{O}(MNK)$, where M and N are the width and the height of the image, K is the number of disparities, C is the number of feature channels and P is the size of the 3D kernel. Although Zhang et al. [53] have a similar runtime of their SGA Layer, they still use 15 3D conv layers with 48 feature volumes in every layer in their full model which is very expensive. Note that their LGA Layer also operates on a 4D input, *i.e.* on multiple 3D feature volumes, where in difference we use only one 3D volume in all stereo experiments. Chang and Chen [4], Kendall et al. [16] use 19 and 25 3D conv layers, respectively. In difference, as our ablation study in the main paper shows, we are on-par with these methods on several metrics. Furthermore, our method is the only method which is also able to achieve high quality results on the difficult Middlebury 2014 benchmark.

B.2. Model Architecture

Table B.1 shows our very lightweight architecture which we use for feature extraction. We actually maintain two copies of this networks with non-shared parameters. The first one is used as the feature network for matching and the second one is the feature network for predicting the pairwise

[5] https://pytorch.org

Layer	KS	Resolution	Channels	Input
conv00	3	$W \times H$ / $W \times H$	3 / 16	Image
conv01	3	$W \times H$ / $W \times H$	16 / 16	conv00
pool0	2	$W \times H$ / $\frac{W}{2} \times \frac{H}{2}$	16 / 16	conv01
conv10	3	$\frac{W}{2} \times \frac{H}{2}$ / $\frac{W}{2} \times \frac{H}{2}$	16 / 32	pool0
conv11	3	$\frac{W}{2} \times \frac{H}{2}$ / $\frac{W}{2} \times \frac{H}{2}$	32 / 32	conv10
pool1	2	$\frac{W}{2} \times \frac{H}{2}$ / $\frac{W}{4} \times \frac{H}{4}$	32 / 32	conv10
conv20	3	$\frac{W}{4} \times \frac{H}{4}$ / $\frac{W}{4} \times \frac{H}{4}$	32 / 64	pool1
conv21	3	$\frac{W}{4} \times \frac{H}{4}$ / $\frac{W}{4} \times \frac{H}{4}$	64 / 64	conv20
bilin1	-	$\frac{W}{4} \times \frac{H}{4}$ / $\frac{W}{2} \times \frac{H}{2}$	64 / 64	conv21
conv12	3	$\frac{W}{2} \times \frac{H}{2}$ / $\frac{W}{2} \times \frac{H}{2}$	96 / 32	{bilin1, conv11}
conv13	3	$\frac{W}{2} \times \frac{H}{2}$ / $\frac{W}{2} \times \frac{H}{2}$	32 / 32	conv12
bilin0	-	$\frac{W}{2} \times \frac{H}{2}$ / $W \times H$	32 / 32	conv12
conv02	3	$W \times H$ / $W \times H$	48 / 32	{bilin0, conv01}
conv03	3	$W \times H$ / $W \times H$	32 / 32	conv02

Table B.1: Detailed Architecture of our UNet for feature extraction.

jump-scores. Figs. A.1 and B.1 show the functions used for unary costs and pairwise costs respectively. Note, that both functions are robust due to the truncation.

On every hierarchical level we add one convolution layer to map the features to pixel-wise descriptors used for matching and to pixel-wise jump-scores respectively. We denote the convolutions as "convD{0,1,2}" and "convS{0,1,2}", where D stands for disparity and S for scores. The highest resolution is here level 0 and the lowest resolution is level 2 in our setting. In the last group in Table B.2 we show the hierarchical inference block. We apply our BP-Layer on the score-volume with the coarsest scale, *i.e.* level 2, upsample the result trilinearly and combine it with SAD matching from the next level. We apply this procedure until we get a regularized score-volume on the finest level, *i.e.* level 0.

Note that the resolutions given in Tables B.1 and B.2 are relative to the input image size. We use with a factor 2 bilinearly downsampled images as the input to our feature networks in all experiments but Kitti. In Kitti we do all computations on the full-size images directly.

C. More Details on Experiments

Due to the limited space in the main paper, we add additional qualitative results and interpretations of these results here. In the following sections, we discuss additional experiments which were performed for Stereo, Semantic Segmentation and Optical Flow.

Layer	KS	Resolution	Channels	Input
convD2	3	$\frac{W}{4} \times \frac{W}{4}$ / $\frac{H}{4} \times \frac{H}{4}$	64 / 32	conv21
convD1	3	$\frac{W}{2} \times \frac{W}{2}$ / $\frac{H}{2} \times \frac{H}{2}$	32 / 32	conv13
convD0	3	$W \times W$ / $H \times H$	32 / 32	conv03
convS2	3	$\frac{W}{4} \times \frac{W}{4}$ / $\frac{H}{4} \times \frac{H}{4}$	64 / 32	conv21
convS1	3	$\frac{W}{2} \times \frac{W}{2}$ / $\frac{H}{2} \times \frac{H}{2}$	32 / 32	conv13
convS0	3	$W \times W$ / $H \times H$	32 / 32	conv03
sad2	-	$\frac{W}{4} \times \frac{W}{4}$ / $\frac{H}{4} \times \frac{H}{4}$	32 / $\frac{D}{4}$	convD2_0, convD2_1
sad1	-	$\frac{W}{2} \times \frac{W}{2}$ / $\frac{H}{2} \times \frac{H}{2}$	32 / $\frac{D}{2}$	convD1_0, convD1_1
sad0	-	$W \times W$ / $H \times H$	32 / D	convD0_0, convD0_1
BP2	-	$\frac{W}{4} \times \frac{W}{4}$ / $\frac{H}{4} \times \frac{H}{4}$	$\frac{D}{4}$ / $\frac{D}{4}$	sad2, convS2
BP2_up	-	$\frac{W}{4} \times \frac{W}{4}$ / $\frac{H}{2} \times \frac{H}{2}$	$\frac{D}{4}$ / $\frac{D}{2}$	BP2
BP1	-	$\frac{W}{2} \times \frac{W}{2}$ / $\frac{H}{2} \times \frac{H}{2}$	$\frac{D}{2}$ / $\frac{D}{2}$	sad1 + BP2_up, convS1
BP1_up	-	$\frac{W}{2} \times \frac{W}{2}$ / $W \times H$	$\frac{D}{2}$ / D	BP1
BP0	-	$W \times W$ / $H \times H$	D / D	sad0 + BP1_up, convS0

Table B.2: Hierarchical BP inference block. We add convolutions to map the features from the feature net to appropriate input to our BP-Layer. The plus operation '+' indicates a point-wise addition.

C.1. Stereo

Fig. C.1 shows a qualitative ablation study comparing our model variants on selected images. Note that we show here exactly the same model variants as in Table 1. The visual ablation study shows interesting insights about our models: First, the WTA result (2nd row in Fig. C.1) is already a very good initialization on all matchable pixels although we use a very efficient network (Table B.1) which uses only 130k parameters. The BP-Layer regularizes the WTA solution by removing most of the artifacts, especially in occluded regions as can be seen in the 3rd row. However, due to the NLL loss function the discretization artifacts are visible in *e.g.* the 3rd example from left. The multi-scale variant adds robustness in large, untextured regions as can be seen in *e.g.* example 1 on the gray box. Training with the Huber loss (row 5) enables sub-pixel accurate solutions. Note how this model captures fine details such as the bar better than the previous models. Our final model can then be used to recover very fine details such as the spokes of the motorcycle in the first example.

Figs. C.2 and C.3 show additional qualitative results on the Middlebury 2014 test set and the Kitti 2015 test set. We include the input image and the error images which are provided by the respective benchmarks.

In Fig. C.4 we compare our prediction with the prediction of current state-of-the-art models. While GA-Net [53], HD3-Stereo [51] and PSM-Net [4] predict precise disparity maps

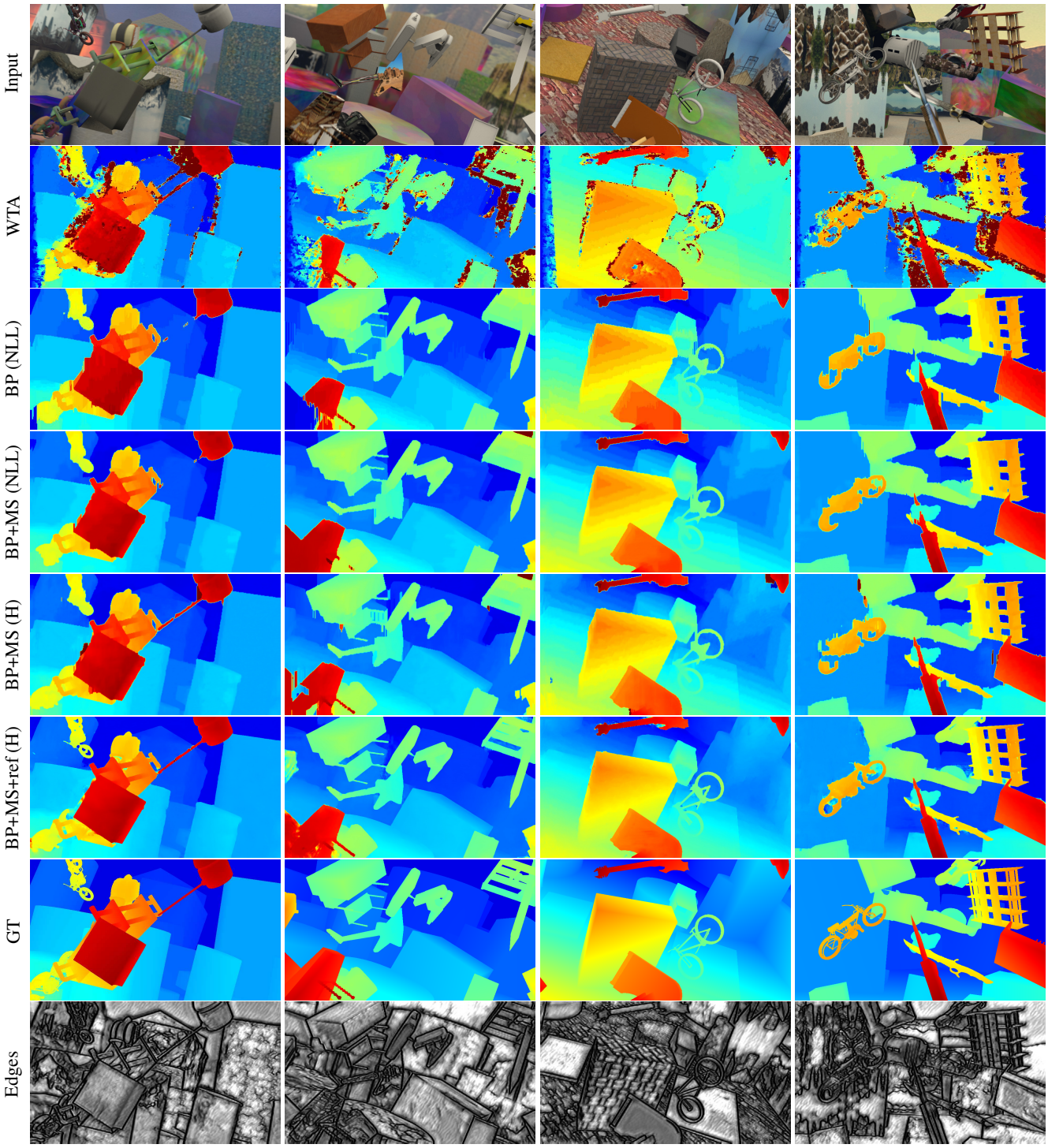

Figure C.1: Visual ablation study. The methods are the same as used in the quantitative ablation study (Table 1) and compared from top to bottom. The last row shows the learned jump-costs of BP+MS+Ref (H) used in our BP-Layer, where black=low cost and white=high cost. The edge images are easily interpretable. We can see that the object edges and depth discontinuities are precisely captured.

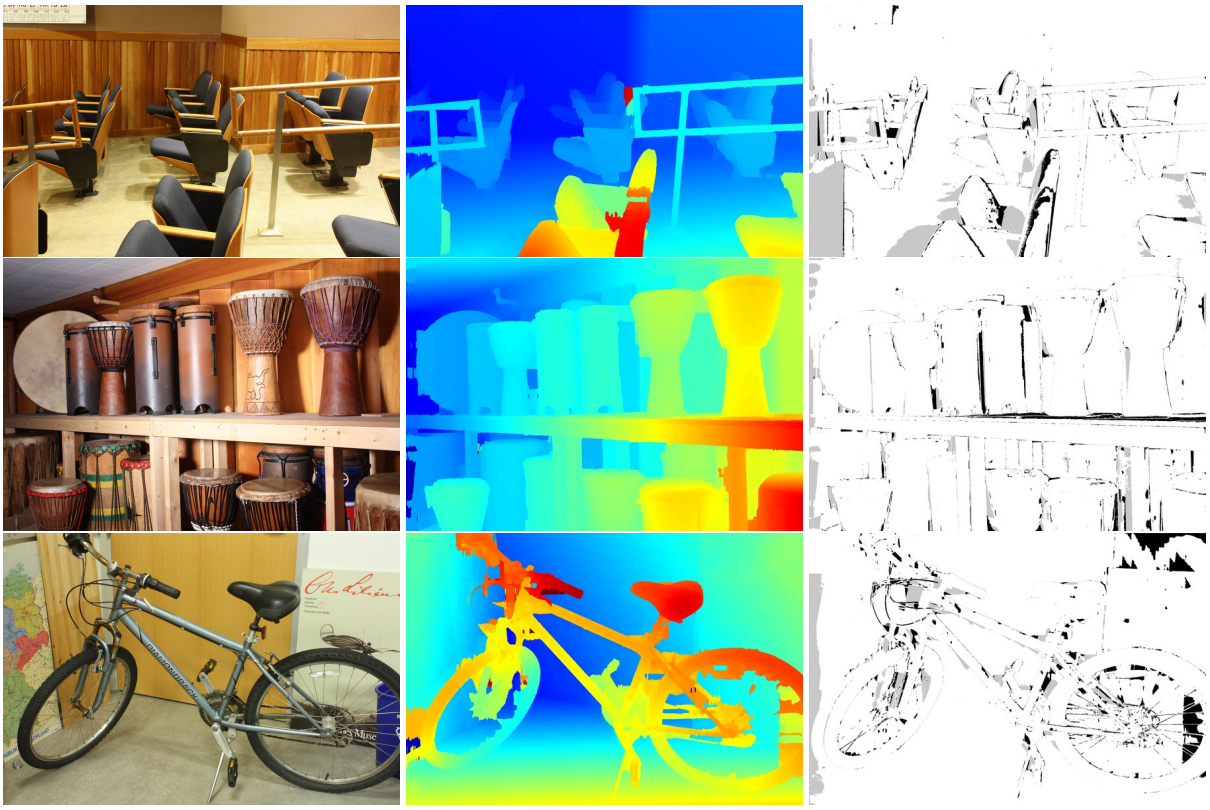

Figure C.2: Qualitative results on the Middlebury 2014 test set. Left: color coded disparity map, right error map, where white = correct disparity, black = wrong disparity and gray = occluded area.

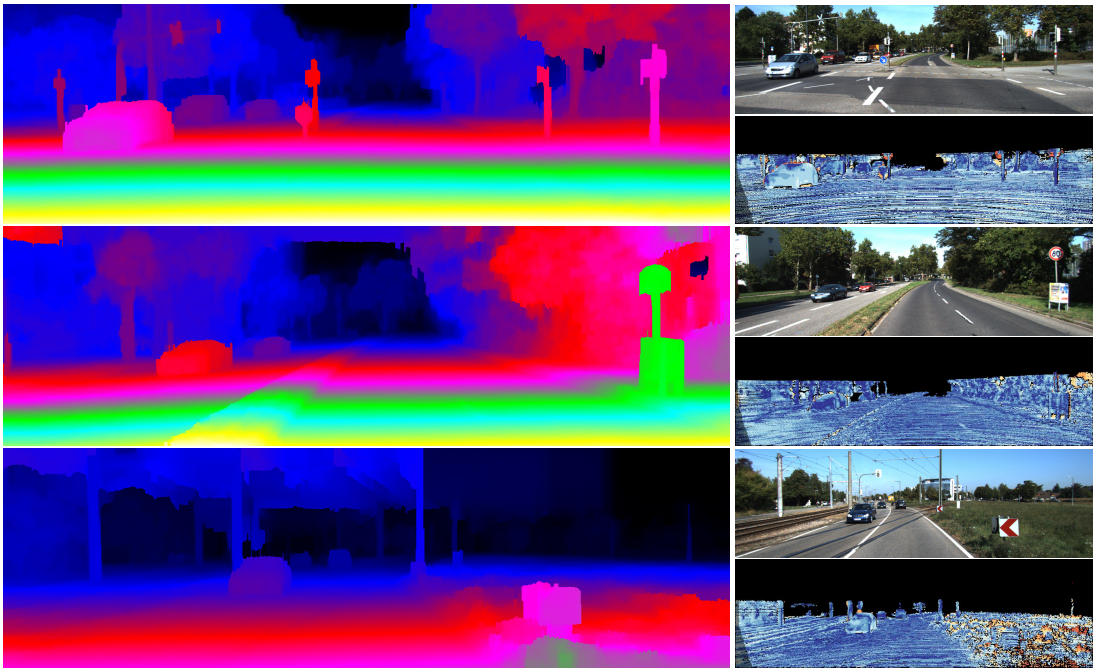

Figure C.3: Kitti test set examples. The left column shows the color-coded disparity map, the right column shows on top the input image and on the bottom the official error map on the Kitti benchmark. The blue color in the error map indicates correct predictions, orange indicate wrong predictions and black is unknown. Note how our method produces high quality results also for regions where no ground-truth is available, *i.e.* in the upper third of the images.

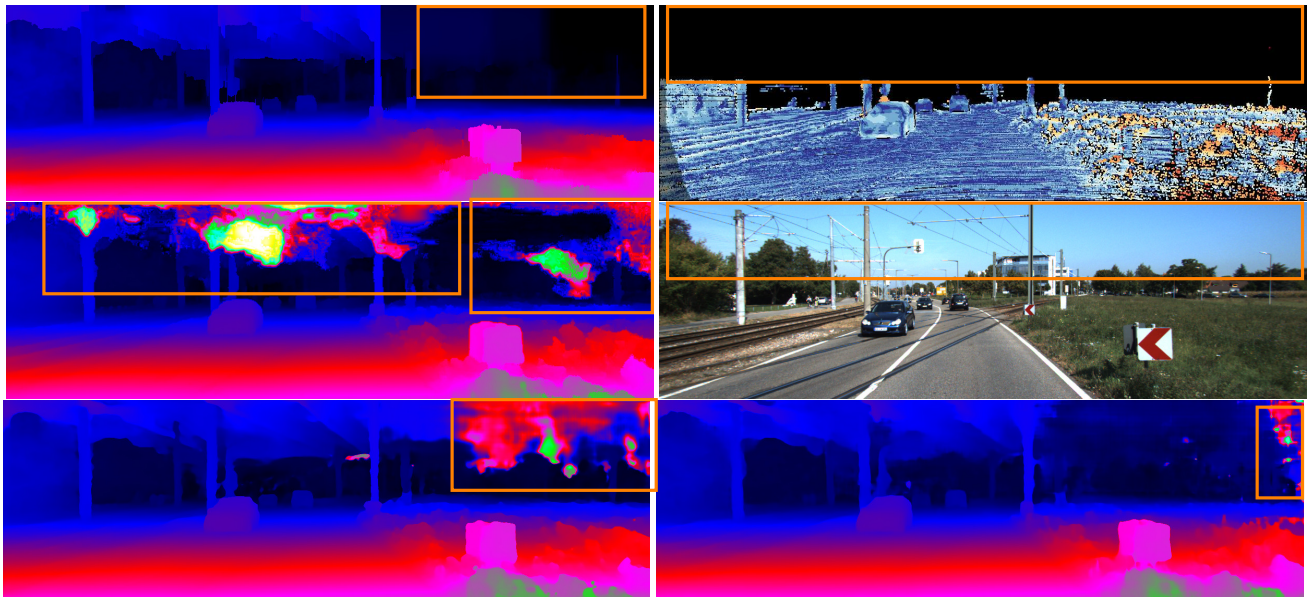

Figure C.4: Comparison with other methods on the Kitti benchmark. Top row: LBPS (ours), LBPS error visualization. Middle row: HD3 Stereo [51], input image. Bottom row: GANet [53], PSMNet [4]. One can observe that LBPS shows no artifacting in regions where no ground truth is present.

for pixels with available ground-truth, they often hallucinate incorrect disparities on the other pixels. In contrast, our method does not seem to be affected at all by this problem and thus this indicates that our model generalizes very well also to previously unseen structures. For a better comparison we highlighted these regions in Fig. C.4.

C.2. Optical Flow

We use the same network architectures for optical flow as for stereo. Thus, we have two feature nets Table B.1 and then apply hierarchically our BP-Layer on the cost-volumes.

Here we show here more examples on our validation set and highlight differences until we get our final model BP+MS+Ref (H). Therefore, Fig. C.5 shows a visual ablation study. If we compare the models we see that the quality of the results increase from top to bottom. Thus, the components we add are also beneficial for optical flow. If we add our BP-Layer and use it to regularize the WTA result we can clearly see that most of the noise, mainly coming from occlusions, is gone. The Huber loss function and the refinement successfully predict then contiguous solutions. Although our approach is very simplistic in comparison with current state-of-the-art models we are still able to compute high quality optical flow.

C.3. Semantic Segmentation

We show here additional evaluation metrics provided by the Cityscapes benchmark. In Table C.1, we show the category mIOU score for each invidual category. It can be observed, that the BP-Layer improves this metric for every category and thus the average score for all categories is also improved. The BP-layer also improves the average class mIOU, as seen in Table C.2. For this metric, the BP-layer improves the results for most classes. However, the mIOU is slightly decreased for the classes truck, train and motorcycle. This is due to the fact that a confusion between these classes in the result from ESPNet [32] can be propagated by the BP-Layer leading to larger patches of incorrect semantic labels. Figure C.7 shows a visual ablation study of the different models for semantic segmentation. It can be seen that all of the models utilizing the BP-Layer are able to regularize over inconsistencies in the original result from ESPNet [32]. Furthermore, the pixel wise models are able to better preserve fine structures like traffic lights. If we use the BP-Layer without jointly training the ESPNet, we get some line artifacts in the global and pixel results. These artefacts are easily removed by jointly training both networks as seen in the pixel joint result.

In Figure C.6, we show qualitative results from the LBPSS pixel joint model on the test set of Cityscapes [7]. It can be seen that the detail on the boundaries of the segmentation masks for scene elements such as cars and pedestrians is preserved, as transition scores are predicted from the input image. We can also show the full vertical transition score matrix for all classes, which we do in Figure C.8. As described in the paper, the matrix is not symmetric which allows for different scores when transitioning upwards and downwards. If we investigate this matrix in more detail, we are actually able to interpret the learned results. An interesting observation can *e.g.* be seen when looking at the column for the

Figure C.5: Qualitative ablation study for optical flow. The WTA result clearly shows occluded regions (the noisy regions), while our model is able to successfully inpaint these regions. Note that the details increase from top to bottom.

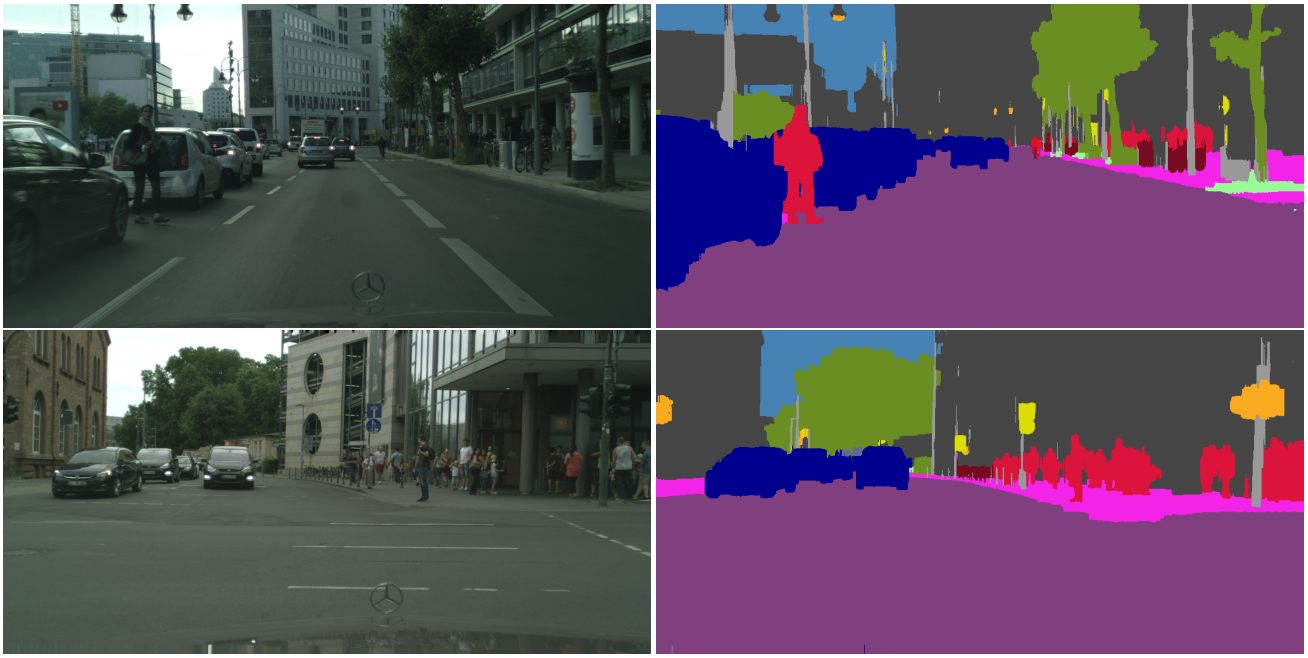

Figure C.6: Qualitative results for semantic segmentation on the Cityscapes [7] test set. Our model is able to precisely capture object boundaries around *e.g.* pedestrians and cars.

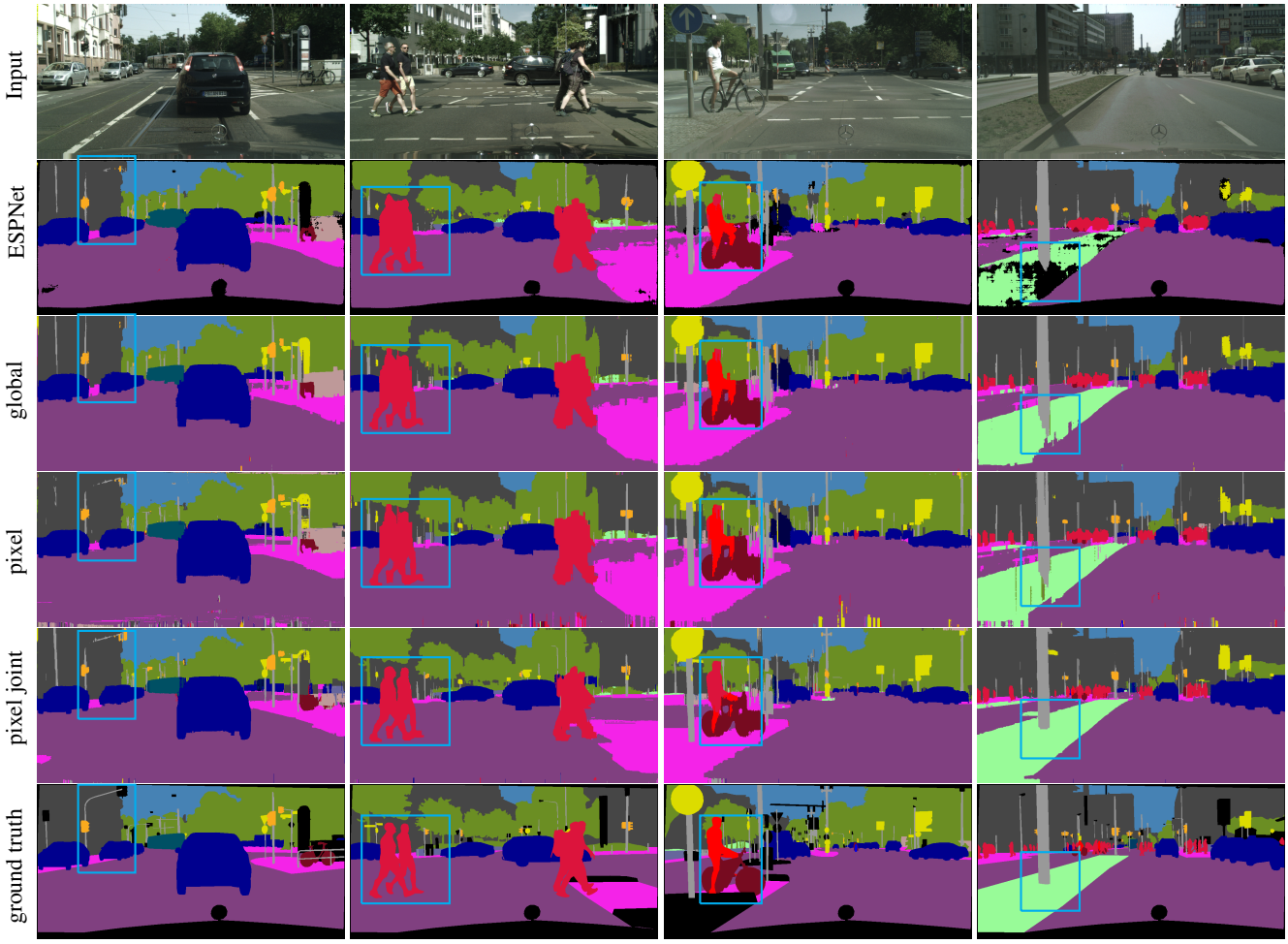

Figure C.7: Visual ablation study for semantic segmentation on the Cityscapes [7] validation set. The results in the first column show that the BP-Layer can recover fine details such as the thin structures of the traffic light. In the second column one can observe that the legs and heads of the pedestrians are recovered and do not appear as a single blob-like structure. This can also be seen when looking at the bike in the third column. The fourt column shows that the BP-Layer can regularize over inconsistencies in the initial estimation from ESPNet [32] as seen on the sidewalk.

Method	**avg**	**flat**	**nature**	**object**	**sky**	**construction**	**human**	**vehicle**
ESPNet [32]	82.18	95.49	89.46	52.94	92.47	86.67	69.76	88.45
LBPSS pixel-wise joint	**84.31**	**97.90**	**90.01**	**58.89**	**93.10**	**88.08**	**72.79**	**89.43**

Table C.1: Benchmark results for categories on the Cityscapes [7] test set

Method	**avg**	road	side.	build.	wall	fen.	pole	tr. light	tr. sign	veg.	terr.	sky	person	rider	car	truck	bus	train	motorc.	bic.
ESPNet [32]	60.34	95.68	73.29	86.60	**32.79**	36.43	47.06	46.92	55.41	89.83	**65.96**	92.47	68.48	**45.84**	89.90	**40.00**	47.73	**40.70**	**36.40**	54.89
LBPSS pw joint	**61.00**	**97.00**	**76.88**	**87.38**	31.29	**37.99**	**53.60**	**53.84**	**60.85**	**90.41**	65.85	**93.10**	**70.34**	43.27	**90.93**	31.59	**50.32**	33.93	31.77	**58.67**

Table C.2: Benchmark results with respect to the mIOU metric for each class on the Cityscapes [7] test set.

sky class. It encodes that downward label transitions from car, truck or train to sky are very expensive and upwards transitions from *e.g.* car to sky are comparably cheap. This is very intuitive and encodes that the sky is always above the car and not below. Another example is that traffic lights and vegetation are often surrounded by sky and thus these scores are higher. Also the scores for the unknown class very intuitive. The very similar scores to all other classes can be interpreted as a uniform distribution. This makes totally sense, because the class "unknown" has interactions with all other classes.

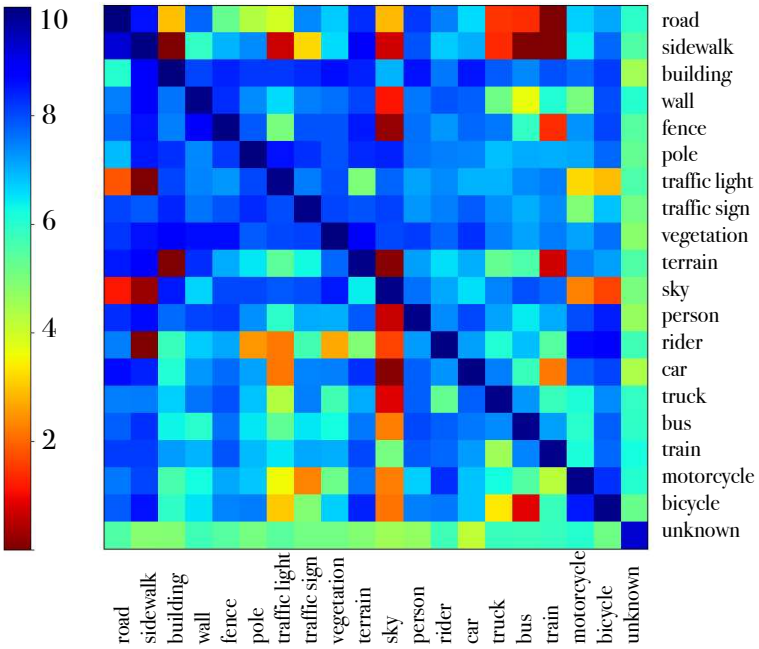

Figure C.8: Vertical transition score matrix for all classes, where the upper triangular matrix encodes upwards transitions and the lower triangular matrix encodes downwards transitions.