Belief Propagation Reloaded: Learning BP-Layers for Labeling Problems

Patrick Knöbelreiter[1] knoebelreiter@icg.tugraz.at
Christian Sormann[1] christian.sormann@icg.tugraz.at
Alexander Shekhovtsov[2] shekhovtsov@gmail.com
Friedrich Fraundorfer[1] fraundorfer@icg.tugraz.at
Thomas Pock[1] pock@icg.tugraz.at
[1] Graz University of Technology
[2] Czech Technical University in Prague

Abstract

It has been proposed by many researchers that combining deep neural networks with graphical models can create more efficient and better regularized composite models. The main difficulties in implementing this in practice are associated with a discrepancy in suitable learning objectives as well as with the necessity of approximations for the inference. In this work we take one of the simplest inference methods, a truncated max-product Belief Propagation, and add what is necessary to make it a proper component of a deep learning model: We connect it to learning formulations with losses on marginals and compute the backprop operation. This BP-Layer can be used as the final or an intermediate block in convolutional neural networks (CNNs), allowing us to design a hierarchical model composing BP inference and CNNs at different scale levels. The model is applicable to a range of dense prediction problems, is well-trainable and provides parameter-efficient and robust solutions in stereo, optical flow and semantic segmentation.

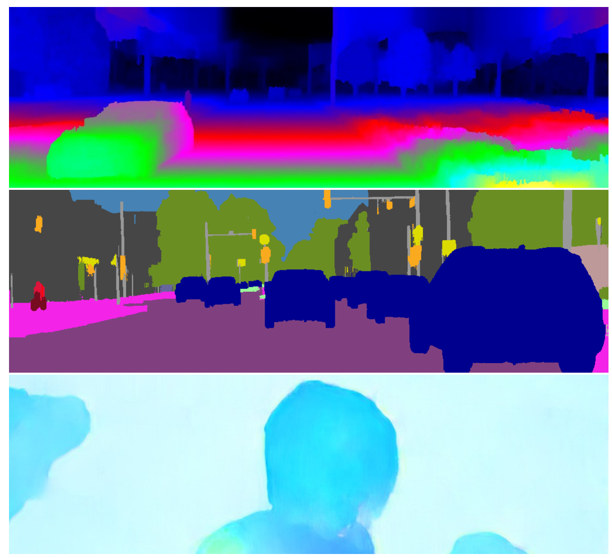

Figure 1: BP-Layer in action. The BP-Layer can be used for dense prediction problems such as stereo (top) semantic segmentation (middle) or optical flow (bottom). Note the sharp and precise edges for all three tasks. Input images are from Kitti, Cityscapes and Sintel benchmarks.

1. Introduction

We consider dense prediction tasks in computer vision that can be formulated as assigning a categorical or real value to every pixel. Of particular interest are the problems of semantic segmentation, stereo depth reconstruction and optical flow. The importance of these applications is indicated by the active development of new methods and intense competition on common benchmarks.

Convolutional Neural Networks (CNNs) have significantly pushed the limits in dense prediction tasks. However, composing only CNN blocks, though a general solution, becomes inefficient if we want to increase robustness and accuracy: with the increase of the number of blocks and respectively parameters the computational complexity and the training data required grow significantly. The limitations are in particular in handling long-range spatial interactions and structural constraints, for which Conditional Random Fields (CRFs) are much more suitable. Previous work has shown that a combination of CNN+CRF models can offer an increased performance, but incorporating inference in the stochastic gradient training poses some difficulties.

In this work we consider several simple inference methods for CRFs: A variant of Belief Propagation (BP) [43], tree-structured dynamic programming [2] and semi-global matching [13]. We introduce a general framework, where we view all these methods as specific schedules of max-product BP updates and propose how to use such BP inference as a layer in neural networks fully compatible with deep learning. The layer takes categorical probabilities on the input and produces refined categorical probabilities on the output, associated with marginals of the CRF. This allows for direct training of the truncated inference method by propagating gradients through the layer. The proposed BP-Layer can have an associated loss function on its output probabilities, which we argue to be more practical than other variants of

CRF training. Importantly, it can be also used as an inner layer of the network. We propose a multi-resolution model in which BP-Layers are combined in a hierarchical fashion and feature both, associated loss functions as well as dependent further processing blocks.

We demonstrate the effectiveness of our BP-Layer on three dense prediction tasks. The BP-Layer performs a global spatial integration of the information on the pixel-level and is able to accurately preserve object boundaries as highlighted in Fig. 1. Deep models with this layer have the following beneficial properties: (i) they contain much fewer parameters, (ii) have a smaller computation cost than the SoTA fully CNN alternatives, (iii) they are better interpretable (for example we can visualize and interpret CRF pairwise interaction costs) and (iv) lead to robust accuracy rates. In particular, in the high-resolution stereo Middlebury benchmark, amongst the models that run in less than 10 seconds, our model achieves the second best accuracy. The CRF for stereo is particularly efficient in handling occlusions, explicitly favoring slanted surfaces and in modelling a variable disparity range. In contrast, many CNN techniques have the disparity range hard-coded in the architecture.

Related Work

We discuss the related work from the points of view of the learning formulation, gradient computation and application in dense prediction tasks.

CRF Learning CRFs can be learned by the maximum margin approach (*e.g.*, [17, 22]) or the maximum likelihood approach and its variants (*e.g.*, [1, 20, 27, 35]). In the former, the loss depends on the optimal (discrete) solution and is hard to optimize. In the latter, the gradient of the likelihood is expressed via marginals and approximate marginals can be used. However, it must be ensured that during learning enough iterations are performed, close to convergence of the approximation scheme [8], which is prohibitive in large-scale learning settings. Instead, several works advocate truncated inference and a loss function directly formulated on the approximate marginals [8, 9, 15]. This gives a tighter connection between learning and inference, is better corresponding to the empirical loss minimization with the Hamming loss and is easy to apply with incomplete ground truth labelings. Experimental comparison of multiple learning approaches for CRFs [9] suggest that marginalization-based learning performs better than likelihood-based approximations on difficult problems where the model being fit is approximate in nature. Our framework follows this approach.

Differentiable CRF Inference For learning with losses on marginals Domke [9] introduced Back-Mean Field and Back-TRW algorithms allowing back-propagation in the respective inference methods. Back-Belief Propagation [11] is an efficient method applicable at a fixed point of BP, originally applied in order to improve the quality of inference, and not suitable for truncated inference. While the methods [8, 9, 11] consider the sum-product algorithms and back-propagate their elementary message passing updates, our method back-propagates the sequence of max-product BP updates on a chain at once. Max-product BP is closely related with the Viterbi algorithm and Dynamic Programming (DP). However, DP is primarily concerned with finding the optimal configuration. The smoothing technique [33] addresses differentiating the optimal solution itself and its cost. In difference, we show the back propagation of max-marginals.

The *mean field inference* in fully connected CRFs for semantic segmentation [5, 54] like our method maps label probabilities to label probabilities, is well-trainable and gives improvements in semantic segmentation. However, the model does not capture accurate boundaries [30] and cannot express constraints needed for stereo/flow such as non-symmetric and anisotropic context dependent potentials.

Gaussian CRFs (GCRFs) use quadratic costs, which is restrictive and not robust if the solution is represented by one variable per pixel. If K variables are used per pixel [46], a solution of a linear system of size $K \times K$ is needed per each pairwise update and the propagation range is only proportional to the number of iterations.

Semi-Global Matching (SGM) [13] is a very popular technique adopted by many works on stereo due to its simplicity and effectiveness. However, its training has been limited either to learning only a few global parameters [33] or to indirect training via auxiliary loss functions [40] avoiding backpropagating SGM. Although we focus on a different inference method, our framework allows for a simple implementation of SGM and its end-to-end learning.

Non-CRF Propagation Many methods train continuous optimization algorithms used inside neural networks by unrolling their iterations [21, 39, 47]. Spatial propagation networks [28], their convolutional variant [6] and guided propagation [53] apply linear spatial propagation models in particular in stereo reconstruction. In difference, we train an inference algorithm that applies non-linear spatial propagation. From this point of view it becomes related to recurrent non-linear processing methods such PixelCNN [45].

2. Belief Propagation

In this section we give an overview of sum-product and max-product belief propagation (BP) algorithms and argue that max-marginals can be viewed as approximation to marginals. This allows to connect learning with losses on marginals [9] and the max-product inference in a non-standard way, where the output is not simply the approximate MAP solution, but the whole volume of max-marginals.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph and $\mathcal{L}$ a discrete set of labels. A pairwise Markov Random Field (MRF) [25] over $\mathcal{G}$ with state space $\mathcal{V}^{\mathcal{L}}$ is a probabilistic graphical model

$p\colon \mathcal{V}^{\mathcal{L}} \to \mathbb{R}_+$ that can be written in the form

$$p(x) = \frac{1}{Z} \exp\Big(\sum_{i\in\mathcal{V}} g_i(x_i) + \sum_{(i,j)\in\mathcal{E}} f_{ij}(x_i, x_j)\Big), \quad (1)$$

where Z is the normalization constant, functions $g_i\colon \mathcal{L} \to \mathbb{R}$ are the *unary scores*[1], typically containing data evidence; and functions $f_{ij}\colon \mathcal{L}^2 \to \mathbb{R}$ are *pairwise scores* measuring the compatibility of labels at nodes i and j. A CRF $p(x|y)$ is a MRF model (1) with scores depending on the inputs y.

Belief Propagation [37] was proposed to compute marginal probabilities of a MRF (1) when the graph $\mathcal{G}$ is a tree. BP iteratively sends *messages* $M_{ij} \in \mathbb{R}^L_+$ from node i to node j with the update:

$$M^{k+1}_{ij}(t) \propto \sum_s e^{g_i(s)} e^{f_{ij}(s,t)} \prod_{n\in\mathcal{N}(i)\setminus j} M^k_{ni}(s), \quad (2)$$

where $\mathcal{N}(i)$ is the set of neighboring nodes of a node i and k is the iteration number. In a tree graph a message M_{ij} is proportional to the marginal probability that a configuration of a tree branch ending with (i, j) selects label t at j. Updates of all messages are iterated until the messages have converged. Then the marginals, or in a general graph *beliefs*, are defined as

$$B_i(x_i) \propto e^{g_i(x_i)} \prod_{n\in\mathcal{N}(i)} M_{ni}(x_i), \quad (3)$$

where the proportionality constant ensures $\sum_s B_i(s) = 1$.

The above *sum-product* variant of BP can be restated in the log domain, where the connection to max-product BP becomes apparent. We denote $\widetilde{\max}$ the operation $\mathbb{R}^n \to \mathbb{R}$ that maps $(a_1, \dots a_n)$ to $\log \sum_i e^{a_i}$, known as log-sum-exp or *smooth maximum*. The update of the sum-product BP (2) can be expressed as

$$m^{k+1}_{ij}(t) := \widetilde{\max_s} \Big(g_i(s) + f_{ij}(s, t) + \sum_{n\in\mathcal{N}(i)\setminus j} m^k_{ni}(s)\Big), \quad (4)$$

where m are the log domain messages, defined up to an additive constant. The *log-beliefs* are respectively

$$b_i(x_i) = g_i(x_i) + \sum_{n\in\mathcal{N}(i)} m_{ni}(x_i). \quad (5)$$

The *max-product BP* in the log domain takes the same form as (4) but with the hard $\max$ operation. Max-product solves the problem of finding the configuration x of the maximum probability (MAP solution) and computes *max-marginals* via (5). It can be viewed as an approximation to the marginals problem since there holds

$$\max_i a_i \le \widetilde{\max_i} a_i \le \max_i a_i + \log n \quad (6)$$

[1] The negative scores are called *costs* in the context of minimization.

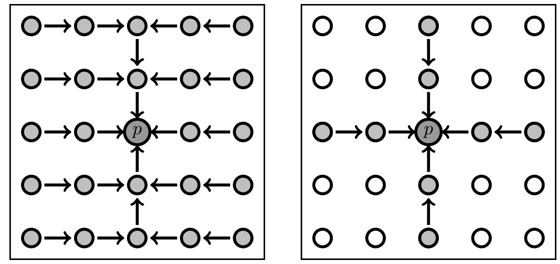

Figure 2: Max-marginal computation for node p on the highlighted trees. *Left*: Left-right-up-down BP [43] or equivalent tree DP [2]. *Right*: SGM [13] on a 4-connected graph. Note that SGM prediction for node p uses much smaller trees, ignoring the evidence from out of tree nodes.

for any tuple $(a_1 \dots a_n)$. Preceding work has noticed that max-marginals can in practice be used to assess uncertainty [23], *i.e.*, they can be viewed as approximation to marginals. The perturb and MAP technique [36] makes the relation even more precise. In this work we apply max-marginal approximation to marginals as a practical and fast inference method for both, prediction time and learning. We rely on deep learning to make up for the approximation. In particular the learning can tighten (6) by scaling up all the inputs.

To summarize, the approximation to marginals that we construct is obtained by running the updates (4) with hard $\max$ and then computing beliefs from log-beliefs (5) as

$$B_i(x_i{=}s) = \operatorname*{softmax}_s b_i(s), \quad (7)$$

where $\operatorname{softmax}_s b_i(s) = e^{b_i(s)} / \sum_s e^{b_i(s)}$. Beliefs constructed in this way may be used in the loss functions on the marginal or as an input to subsequent layers, similarly to how simple logistic regression models are composed to form a sigmoid neural network. This approach is akin to previous work that used the regularized cost volume in a subsequent refinement step [18], but is better interpretable and learnable with our methods.

3. Sweep BP-Layer

When BP is applied in general graphs, the schedule of updates becomes important. We find that the parallel synchronous update schedule [38] requires too many iterations to propagate information over the image and rarely converges. For application in deep learning, we found that the schedule which makes sequential sweeps in different directions as proposed by [43] is more suitable. For a given sweep direction, we can compute the result of all sequential updates and backpropagate the gradient in a very efficient and parallel way. This allows to propagate information arbitrarily far in the sweep direction, while working on a pixel level, which makes this schedule very powerful.

Before detailing the sweep variant of BP [43], let us make clear what is needed in order to make an operation a part of an end-to-end learning framework. Let us denote the gradient of a loss function L in variables y as $\text{đ}y := \frac{\mathrm{d}L}{\mathrm{d}y}$. If a layer computed $y = f(x)$ in the forward pass, the gradient in x is obtained as

$$\text{đ}x_j = \sum_i \frac{\partial f_i}{\partial x_j} \text{đ}y_i, \tag{8}$$

called the *backprop* of layer f. For the BP-Layer the input probabilities x and output beliefs y are big arrays containing all pixels and all labels. It is therefore crucial to be able to compute the backprop in linear time.

3.1. Sweep BP as Dynamic Programming

The BP variant of [43] (called left-right-up-down BP there and BP-M in [42]) performs sweeps in directions left→right, right→left, up→down, down→up. For each direction only messages in that direction are updated sequentially, and the rest is kept unmodified. We observe the following properties of this sweep BP: (i) Left and right messages do not depend on each other and neither on the up and down messages. Therefore, their calculation can run independently in all horizontal chains. (ii) When left-right messages are fixed, they can be combined into unary scores, which makes it possible to compute the up and down messages independently in all vertical chains in a similar manner. These properties allow us to express left-right-up-down BP as shown in Algorithm 1 and illustrated in Fig. 2 (left). In Algorithm 1, the notation $a_{\mathcal{V}'}$ means the restriction of a to the nodes in $\mathcal{V}'$, *i.e*. to a chain. It is composed of dynamic programming subroutines computing max-marginals. Since individual chains in each of the loops do not interact, they can be processed in parallel (denoted as par. for). The max-marginals a of a horizontal chain are computed as

$$a_i(s) = g_i(s) + m_i^L(s) + m_i^R(s), \tag{9}$$

where $m_i^L(s)$ denotes the message to i from its left neighbour and $m_i^R(s)$ from its right. The max-marginals (9) are indeed the beliefs after the left-right pass. The max-marginals b for vertical chains are, respectively,

$$b_i(s) = a_i(s) + m_i^U(s) + m_i^D(s). \tag{10}$$

It remains to define how the messages m are computed and back-propagated. Given a chain and the processing direction (*i.e*., L-R for left messages m^{L}), we order the nodes ascending in this direction and apply dynamic programming in Algorithm 2. The Jacobian of Algorithm 2 is well defined if the maximizer in each step is unique[2]. In this case we have a linear recurrent dependence in the vicinity of the input:

$$m_{i+1}(t) = g_i(s) + m_i(s) + f_{i,i+1}(s,t), \tag{11}$$

[2]Otherwise we take any maximizer resulting in a conditional derivative like with ReLU at 0.

Algorithm 1: Sweep Belief Propagation

Input: CRF scores $g \in \mathbb{R}^{\mathcal{V}\times\mathcal{L}}$, $f \in \mathbb{R}^{\mathcal{E}\times\mathcal{L}^2}$;
Output: Beliefs $B \in \mathbb{R}^{\mathcal{V}\times\mathcal{L}}$;
par. for *each horizontal chain subgraph* $(\mathcal{V}', \mathcal{E}')$ **do**
  $a_{\mathcal{V}'} := \texttt{max\_marginals}(g_{\mathcal{V}'}, f_{\mathcal{E}'})$;
par. for *each vertical chain subgraph* $(\mathcal{V}', \mathcal{E}')$ **do**
  $b_{\mathcal{V}'} := \texttt{max\_marginals}(a_{\mathcal{V}'}, f_{\mathcal{E}'})$;
return beliefs $B_i(s) := \mathrm{softmax}_s(b_i(s))$;

Algorithm 2: Dynamic Programming (DP)

Input: Directed chain $(\mathcal{V}, \mathcal{E})$, nodes $\mathcal{V}$ enumerated in chain direction from 0 to $n{=}|\mathcal{V}|{-}1$, scores $g \in \mathbb{R}^{\mathcal{V}\times\mathcal{L}}$, $f \in \mathbb{R}^{\mathcal{E}\times\mathcal{L}^2}$;
Output: Messages $m \in \mathbb{R}^{\mathcal{V}\times\mathcal{L}}$ in chain direction;
Init: Set: $m_0(s) := 0$; /* first node */
for $i = 0 \dots n-2$ **do**
  /* Compute message: */
  $m_{i+1}(t) := \max_s \big(g_i(s) + m_i(s) + f_{i,i+1}(s,t)\big)$;
  /* Save argmax for backward: */
  $o_{i+1}(t) := \operatorname{argmax}_s \big(g_i(s) + m_i(s) + f_{i,i+1}(s,t)\big)$;
return m;

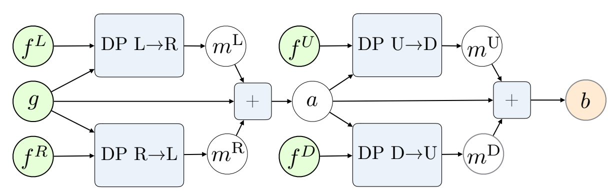

Figure 3: Computation graph of BP-Layer with Sweep BP in Algorithm 1 down to log-beliefs b. Dynamic Programming computational nodes (DP) are made differentiable with the backprop in Algorithm 3. The pairwise terms f^{L}, f^{R}, f^{U}, f^{D} illustrate the case when pairwise scores f_{ij} are different for all four directions.

Algorithm 3: Backprop DP

Input: $\text{đ}m \in \mathbb{R}^{\mathcal{V}\times\mathcal{L}}$, gradient of the loss in the messages m returned by DP on chain $(\mathcal{V}, \mathcal{E})$;
Output: $\text{đ}g \in \mathbb{R}^{\mathcal{V}\times\mathcal{L}}$, $\text{đ}f \in \mathbb{R}^{\mathcal{E}\times\mathcal{L}^2}$, gradients of the loss in the DP inputs g, f;
Init: $\text{đ}g := 0$; $\text{đ}f := 0$;
for $i = n-2 \dots 0$ **do**
  for $t \in \mathcal{L}$ **do**
    $s := o_{i+1}(t)$;
    $z := \text{đ}m_{i+1}(t) + \text{đ}g_{i+1}(t)$;
    $\text{đ}g_i(s) \mathrel{+}= z$;
    $\text{đ}f_{i,i+1}(s,t) \mathrel{+}= z$;
return $\text{đ}g$, $\text{đ}f$;

Algorithm 4: Semi-Global Matching

Input: CRF scores $g \in \mathbb{R}^{\mathcal{V}\times\mathcal{L}}$, $f \in \mathbb{R}^{\mathcal{E}\times\mathcal{L}^2}$;
Output: Beliefs $b \in \mathbb{R}^{\mathcal{V}\times\mathcal{L}}$;
par. for *each direction* k *in* $\{\mathrm{L}, \mathrm{R}, \mathrm{U}, \mathrm{D}\}$ **do**
  par. for *each chain* $(\mathcal{V}', \mathcal{E}')$ *in direction to* k **do**
    $m^k_{\mathcal{V}'} := DP(g_{\mathcal{V}'}, f_{\mathcal{E}'})$;
return $b = g + \sum_k m^k$;

where $s = o_{i+1}(t)$, *i.e.* the label maximizing the message, as defined in Algorithm 2. Back-propagating this linear dependence is similar to multiplying by the transposed matrix, *e.g.*, for the gradient in $g_i(s)$ we need to accumulate over all elements to which $g_i(s)$ is contributing. This can be efficiently done as proposed in Algorithm 3.

Thus we have completely defined sweep BP, further on referred to as *BP-Layer*, as a composition of differential operations. The computation graph of the BP-Layer shown in Fig. 3 can be back-propagated using standard rules and our Backprop DP in order to compute the gradients in all inputs very efficiently.

3.2. Other Inference Methods

We show the generality of the proposed framework by mapping several other inference techniques to the same simple DP operations. This allows to make them automatically differentiable and suitable for learning with marginal losses.

SGM We can implement SGM using the same DP function we needed for BP (Algorithm 4), where for brevity we considered a 4-connected grid graph. As discussed in the related work, the possibility to backpropagate SGM was previously missing and may be useful.

Tree-structured DP Bleyer and Gelautz [2] proposed an improvement to SGM by extending the local tree as shown in Fig. 2 (left), later used *e.g.* in a very accurate stereo matching method [50]. It seems it has not been noticed before that sweep BP [43] is exactly equivalent to the tree-structured DP of [2], as clearly seen from our presentation.

TRW and TBCA With minor modifications of the already defined DP subroutines, it is possible to implement and back-propagate several inference algorithms addressing the dual of the LP relaxation of the CRF: the Tree-Reweighted (TRW) algorithm by Wainwright et al. [48] and Tree Block Coordinate Ascent (TBCA) by Sontag and Jaakkola [41], which we show in Appendix A. These algorithms are parallel, incorporate long-range interactions and avoid the evidence over-counting problems associated with loopy BP [48]. In addition, the TBCA algorithm is monotone and has convergence guarantees. These methods are therefore good candidates for end-to-end learning, however they may require more iterations due to cautious monotone updates, which is undesirable in the applications we consider.

4. Models

We demonstrate the effectiveness of the BP-Layer on the three labeling problems: Stereo, Optical Flow and Semantic Segmentation. We have two CNNs (Table B.1) which are used to compute i) score-volumes and ii) pairwise jump-scores, at three resolution levels used hierarchically. Fig. 4 shows processing of one resolution level with the BP-Layer. The label probabilities from these predictions are considered as weak classifiers and the inference block combines them to output a stronger finer-resolution classification. Accordingly, the unary scores $g_i(s)$, called the *score volume*, are set from the CNN prediction probabilities $q_i(s)$ as

$$g_i(s) = T q_i(s), \tag{12}$$

where T is a learnable parameter. Note that g_i is itself a linear parameter of the exponential model (1). The preceding work more commonly used the model $g_i(s) = \log q_i(x)$, which, in the absence of interactions, recovers back the input probabilities. In contrast, the model (12) has the following interpretation and properties: i) it can be viewed as just another non-linearity in the network, increasing flexibility; ii) in case of stereo and flow it corresponds to a robust metric in the feature space (see below), in particular it is robust to CNN predictive probabilities being poorly calibrated.

To combine the up-sampled beliefs B^{up} from the coarser-resolution BP-Layer with a finer-resolution evidence q, we trilinearly upsample the beliefs from the lower level and add it to the score-volume of the current level, *i.e.*

$$g_i(s) = T\big(q_i(s) + B^{\mathrm{up}}_i(s)\big). \tag{13}$$

On the output we have an optional refinement block, which is useful for predicting continuous values for stereo and flow. The simplest refinement takes the average in a window around the maximum:

$$y = \sum_{d:|d-\hat{d}_i|\leq\tau} d\, B_i(d) \Big(\sum_{d:|d-\hat{d}_i|\leq\tau} B_i(d) \Big)^{-1}, \tag{14}$$

where $\hat{d}_i = \operatorname{argmax} B_i(d)$ and we use the threshold $\tau = 3$. Such averaging is not affected by a multi-modal distribution, unlike the full average used in [16]. As a more advanced refinement block we use a variant of the refinement [18] with one up-sampling step using also the confidence of our prediction as an additional input.

4.1. Stereo

For the rectified stereo problem we use two instances of a variant of the UNet detailed in Appendix B. This network is relatively shallow and contains significantly fewer parameters than SoTA. It is applied to the two input images I^0, I^1 and produces two dense feature maps f^0, f^1. The initial prediction of disparity k at pixel i is formed by the distribution

$$q_i(k) = \operatorname*{softmax}_{k \in \{0,1,\dots,D\}} \big(- \|f^0(i) - f^1(i-k)\|_1 \big), \quad (15)$$

where $i - k$ denotes the pixel location in image I^1 corresponding to location i in the reference image I^0 and disparity k and D is the maximum disparity. This model is related to robust costs [24]. The pairwise terms f_{ij} are parametric like in the SGM model [13] but with context-dependent parameters. Specifically, f_{ij} scores difference of disparity labels in the neighbouring pixels. Disparity differences of up to 3 pixels have individual scores, all larger disparity jumps have the same score. All these scores are made context dependent by regressing them with our second UNet from the reference image I^0.

4.2. Optical Flow

The optical flow problem is very similar to stereo. Instead of two rectified images, we consider now two consecutive frames in a video, I^0 and I^1. We use the same UNets to compute the per-pixel features and the jump scores as in the stereo setting. The difference lies in the computation of the initial prediction of flow $u = (u_1, u_2)$. The flow for a pixel i is formed by the two distributions

$$q_i^1(u_1) = \operatorname*{softmax}_{u_1} \max_{u_2} \big(-\|f^0(i) - f^1(i{+}u)\|_1\big), \quad (16)$$

$$q_i^2(u_2) = \operatorname*{softmax}_{u_2} \max_{u_1} \big(-\|f^0(i) - f^1(i{+}u)\|_1\big), \quad (17)$$

which follows the scalable model of Munda et al. [34], avoiding the storage of all matching scores that for an $M{\times}N$ image have the size $M{\times}N{\times}D^2$. The inner maximization steps correspond to the first iteration of an approximate MAP inference [34]. They form an "optimistic" estimate of the score volume for each component of the optical flow, which we process then independently. This scheme may be sub-optimal in that u^1 and u^2 components are inferred independently until the refinement layer, but it scales well to high resolutions (the search window size D needs to grow with the resolution as well) and allows us to readily apply the same BP-Layer model as for the stereo to q^1 and q^2 input probabilities.

4.3. Semantic Segmentation

The task in semantic segmentation is to assign a semantic class label from a number of classes to each pixel. In our model, the initial prediction probabilities are obtained with the ESPNet [32], a lightweight solution for pixel-wise semantic segmentation. This initial prediction is followed up directly with the BP-Layer, which can work with two different types of pairwise scores f_{ij}. The inhomogeneous anisotropic pairwise terms depend on each pixel and on the edge direction, while the homogeneous anisotropic scores

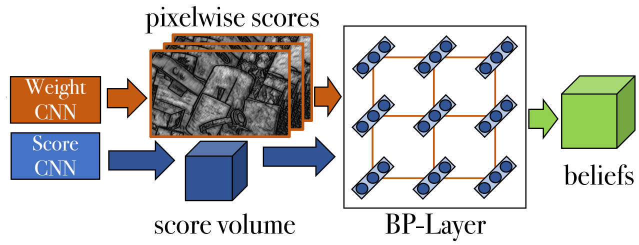

Figure 4: BP-Layer overview. The weight and score CNNs compute pixelwise weights and a score volume from the input image. This is used as an input for the BP-Layer which returns beliefs as an output.

depend only on the edge direction. We implement the homogeneous pairwise terms as parameters within the model and constrain them to be non-negative. The pixel-wise pairwise-terms are computed from the input image using the same UNet as in stereo. We follow the training scheme of [32].

5. Learning

We use the same training procedure for all three tasks. Only the loss function is adapted for the respective task. The loss function is applied to the output of each BP-Layer in the coarse-to-fine scheme and also to the final output after the refinement layer. Such a training scheme is known as deep supervision [26]. For BP output beliefs B^l at level l of the coarse-to-fine scheme, we apply at each pixel i the negative log-likelihood loss $\ell_{\text{NLL}}(B_i^l, d_i^{*l}) = -\log B_i^l(d_i^{*l})$, where d_i^{*l} is the ground truth disparity at scale l.

For the stereo and flow models that have a refinement block targeting real-valued predictions, we add a loss penalizing at each pixel the distance from the target value according to the Huber function:

$$\ell_H(y_i, y_i^*) = \begin{cases} \frac{r^2}{2\delta} & \text{if } |r| \le \delta, \\ |r| - \frac{\delta}{2} & \text{otherwise,} \end{cases} \quad (18)$$

where y_i is the continuous prediction of the model, y_i^* is the ground-truth and $r = y_i - y_i^*$.

Losses at all levels and the losses on the continuous-valued outputs are combined with equal weights[3].

6. Experiments

We implemented the BP-Layer and hierarchical model in PyTorch and used CUDA extensions for time and memory-critical functions (forward and backward for DP, score volume min-projections).[4] Appendices B and C contain the implementation details and additional qualitative results.

[3] the relative weights could be considered as hyper-parameters, but we did not tune them.

[4] https://github.com/VLOGroup/bp-layers

Model	#P	time	bad1		bad3		MAE	
WTA (NLL)	0.13	0.07	10.3	(18.0)	5.27	(13.2)	3.82	(15.1)
BP (NLL)	0.27	0.10	12.6	(17.9)	4.97	(8.12)	1.23	(3.36)
BP+MS (NLL)	0.33	0.11	10.0	(16.5)	3.66	(7.86)	1.13	(2.84)
BP+MS (H)	0.33	0.11	8.15	(15.1)	3.07	(8.00)	0.96	(3.42)
BP+MS+Ref (H)	0.56	0.15	**7.73**	**(13.8)**	**2.67**	(6.46)	**0.74**	(1.67)
GC-Net [16]	3.5	0.95	-	(16.9)	-	(9.34)	-	(2.51)
GA-Net-1 [53]	0.5	0.17	-	(16.5)	-	(-)	-	(1.82)
PDS-Net [44]	2.2	-	-	(-)	-	**(3.38)**	-	**(1.12)**

Table 1: Ablation Study on the Scene flow validation set. We report for all metrics the result on non-occluded and (all pixels). #P in millions. bold = best, underline = second best.

Method	#P[M]	Middlebury 2014		Kitti 2015	
		bad2	time[s]	bad3	time[s]
PSMNet [4]	5.2	42.1 (47.2)	2.62	2.14 (2.32)	0.41
PDS [44]	**2.2**	14.2 (21.0)	12.5	2.36 (2.58)	0.50
HSM [49]	3.2	**10.2 (16.5)**	**0.51**	**1.92 (2.14)**	**0.14**
MC-CNN [52]	**0.2**	**9.47** (20.6)	1.26	3.33 (3.89)	67.0
CNN-CRF [22]	0.3	12.5 (21.9)	3.53	4.84 (5.50)	1.30
ContentCNN [29]	0.7	-	-	4.00 (4.54)	1.00
LBPS **(ours)**	0.3	9.68 **(17.5)**	**1.05**	**3.13 (3.44)**	**0.39**

Table 2: Evaluation on the Test set of the Middlebury and Kitti Stereo Benchmark using the default metrics of the respective benchmarks. *Top group*: Large models with > 1M parameters. *Bottom group*: Light-weight models. Bold indicates the best result in the group.

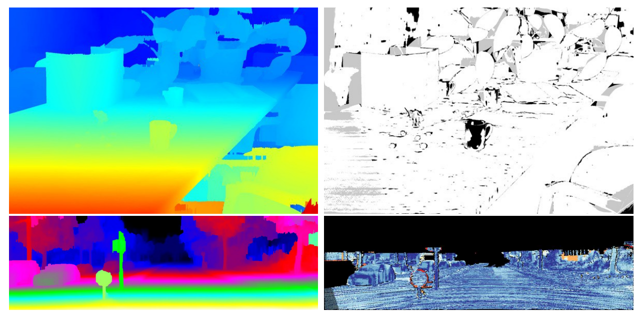

Figure 5: Qualitative results on the test sets of Middlebury 2014 (top) and Kitti 2015 (bottom) datasets. Left: Color coded disparity map, right error map, where white/blue = correct, gray = occluded, black/orange = incorrect. Note how our method produces sharp edges in all results.

6.1. Improvements brought by the BP-Layer

We investigate the importance of different architectural choices in our general model on the stereo task with the synthetic stereo data from the Scene Flow dataset [31]. The standard error metric in stereo is the $\mathrm{bad}X$ error measuring the percentage of disparities having a distance larger than X to the ground-truth. This metric is used to assess the robustness of a stereo algorithm. The second metric is the mean-absolute-error (MAE) which is more sensitive to the (sub-pixel) precision of a stereo algorithm.

Table 1 shows an overview of all variants of our model. We start from the winner-takes-all **(WTA)** model, add the proposed BP-Layer or the multi-scale model **(MS)**, then add the basic refinement (14) trained with Huber loss **(H)**, then add the refinement [18] (**Ref (H)**). The column #P in Table 1 shows the number of parameters of our model, which is significantly smaller than SoTA methods applicable to this dataset. Each of the parts of our model increase the final performance. Our algorithm performs outstandingly well in the robustness metric $\mathrm{bad}X$. The ablation study shows also the impact of the used loss function. It turns out that Huber loss function is beneficial to all the metrics but the MAE in occluded pixels. The optional refinement yielded an additional improvement, especially in occluded pixels on this data, but we could not obtain a similar improvement when training and validating on Middlebury or Kitti datasets. We therefore selected BP+MS (H) model, as the more robust variant, for evaluation in these real-data benchmarks.

6.2. Stereo Benchmark Performance

We use the model BP+MS (H) to participate on the public benchmarks of Middlebury 2014 and Kitti 2015. Both benchmarks have real-world scences, Middlebury focusing on high-resolution indoor scenes and Kitti focusing on low-resolution autonomous driving outdoor scenes. Qualitative test-set results are shown in Fig. 5.

The Middlebury benchmark is very challenging due to huge images, large maximum disparities, large untextured regions and difficult illumination. These properties make it hard or even impossible for most of the best-performing methods from Kitti to be used on the Middlebury benchmark. Due to our light-weight architecture we can easily apply our model on the challenging Middlebury images. The test-set evaluation (Table 2) shows that we are among the best performing methods with a runtime of up to 10 seconds, and thus convincingly shows the effectiveness of our light-weight model. The challenges on the Kitti dataset are regions with over- and under-saturation, reflections and complex geometry. We significantly outperform competitors with a similar number of parameters such as MC-CNN, CNN-CRF and Content CNN, which demonstrates the effectiveness of the learnable BP-Layer. Methods achieving a better performance on Kitti come with the high price of having many more parameters.

6.3. Optical Flow

Here we show the applicability of our BP-Layer to the optical flow problem. We use the FlyingChairs2 dataset [10, 14] for pre-training our model and fine-tune then with the Sintel dataset [3]. In the optical flow setting we set the search-window-size to 109×109 in the finest resolution.

Model	#P[M]	time	bad2	EPE
WTA	**0.13**	**0.27**	4.46 (5.67)	1.25 (1.65)
BP+MS (CE)	0.34	0.44	2.56 (3.46)	0.83 (0.94)
BP+MS (H)	0.34	0.44	2.24 (3.19)	0.66 (0.79)
BP+MS+Ref (H)	0.56	0.49	**2.06 (2.64)**	**0.63 (0.72)**

Table 3: Ablation Study on the Sintel Validation set.

Figure 6: Left: Qualitative optical flow results on the Sintel validation set. Right: Visualization of the endpoint error, where white=correct and darker pixels are erroneous.

We compute the 109^2 similarities per pixel without storing them and compute the two cost-volumes q^1 and q^2 using Eq. (17) on the fly. Fig. 6 shows qualitative results and Table 3 shows the ablation study on the validation set of the Sintel dataset. We use only scenes where the flow is not larger than our search-window in this study. We compare the endpoint-error (EPE) and the bad2 error on the EPE. The results show that our BP-Layer can be directly used for optical flow computation and that the BP-Layer is an important building block to boost performance.

6.4. Semantic Segmentation

We apply the BP-Layer also to semantic segmentation to demonstrate its general applicability. In Table 4 we show results with our model variants described in Section 4.3 using the same CNN block as ESPNet [32], evaluated on the Cityscapes [7] dataset. All model variants using the BP-Layer improve on ESPNet [32] in both the class mean intersection over union (mIOU) and the category mIOU. The best model is, as expected, the jointly trained pixel-wise model referred to as *LBPSS joint*. We have submitted this model to the Cityscapes benchmark. Table 5 shows the results on the test set and we can see that we outperform the baseline. Figure 7 shows that the BP-Layer refines the prediction by aligning the semantic boundaries to actual object boundaries in the image. Due to the long range interaction, the BP-Layer is also able to correct large incorrect regions such as on *e.g.* the road. One of the advantages of our model is that the learned parameters can be interpreted. Fig. 7 shows the learned non-symmetric score matrix, which allows to learn different scores for *e.g.* person $\rightarrow$ car and car $\rightarrow$ person. The upper and lower triangular matrix represent pairwise scores when jumping upwards and downwards in the image, respectively. We can read from the matrix that, *e.g.*, an upward jump from sky to road is not allowed. This confirms the intuition, since the road never occurs above the sky. Our model has thus automatically learned appropriate semantic relations which have been hand-crafted in prior work such as *e.g.* [12].

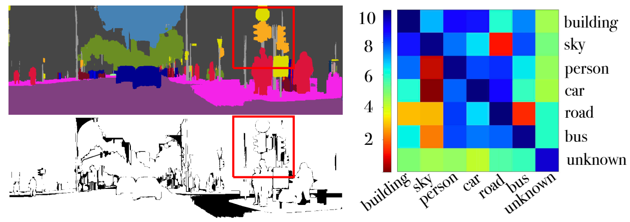

Figure 7: *Top Left*: Semantic segmentation result with the BP-Layer. *Bottom Left*: Corresponding error where black = incorrect, white = correct. The red square highlights the region where fine details were accurately reconstructed. *Right*: Visualization of learned vertical pairwise scores.

Method	pw	mIOU	CatmIOU	#P	time
ESPNet [32]	-	61.4	82.2	**0.36**	**0.01**
LBPSS	-	62.8	83.0	0.37	0.11
LBPSS	✓	63.6	83.7	0.73	0.90
LBPSS joint	✓	**65.2**	**84.7**	0.73	0.90

Table 4: Ablation study on the Cityscapes validation set. *"pw"* = pixel-wise, inhomogeneous scores.

Method	pw	mIOU	CatmIOU	#P	time
ESPNet [32]	-	60.34	82.18	**0.36**	**0.01**
LBPSS joint	✓	**61.00**	**84.31**	0.73	0.90

Table 5: Benchmark results on the Cityscapes [7] test set.

7. Conclusion

We have proposed a novel combination of CNN and CRF techniques, aiming to resolve practical challenges. We took one of the simplest inference schemes, showed how to compute its backprop and connected it with the marginal losses. The following design choices were important for achieving a high practical utility: using max-product for fast computation and backprop of approximate marginals, propagating the information over a long range with sequential subproblems; training end-to-end without approximations; coarse-to-fine processing at several resolution levels; context-dependent learnable unary and pairwise costs. We demonstrated the model can be applied to three dense prediction problems and gives robust solutions with more efficient parameter complexity and time budget than comparable CNNs. In particular in stereo and flow, the model performs strong regularization in occluded regions and this regularization mechanism is interpretable in terms of robust fitting with jump scores.

Acknowledgements This work was supported by the ERC starting grant HOMOVIS (No. 640156), Pro^{2}Future (FFG No. 854184) and the project "International Mobility of Researchers MSCA-IF II at CTU in Prague" (CZ.02.2.69/0.0/0.0/18_070/0010457).

References

[1] Alahari, K., Russell, C., Torr, P.H.S.: Efficient piecewise learning for conditional random fields. In: Conference on Computer Vision and Pattern Recognition (2010)

[2] Bleyer, M., Gelautz, M.: Simple but effective tree structures for dynamic programming-based stereo matching. In: In VISAPP. pp. 415–422 (2008)

[3] Butler, D.J., Wulff, J., Stanley, G.B., Black, M.J.: A naturalistic open source movie for optical flow evaluation. In: A. Fitzgibbon et al. (Eds.) (ed.) European Conference on Computer Vision (ECCV). pp. 611–625 (2012)

[4] Chang, J.R., Chen, Y.S.: Pyramid stereo matching network. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5410–5418 (2018)

[5] Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs. In: ICLR (2015)

[6] Cheng, X., Wang, P., Yang, R.: Learning depth with convolutional spatial propagation network. CoRR abs/1810.02695 (2018)

[7] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)

[8] Domke, J.: Parameter learning with truncated message-passing. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2937–2943 (2011)

[9] Domke, J.: Learning graphical model parameters with approximate marginal inference. IEEE Transactions on Pattern Analysis and Machine Intelligence 35(10), 2454–2467 (October 2013)

[10] Dosovitskiy, A., Fischer, P., Ilg, E., Häusser, P., Hazırbaş, C., Golkov, V., v.d. Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: IEEE International Conference on Computer Vision (ICCV) (2015)

[11] Eaton, F., Ghahramani, Z.: Choosing a variable to clamp: Approximate inference using conditioned belief propagation. In: International Conference on Artificial Intelligence and Statistics. vol. 5, pp. 145–152 (April 2009)

[12] Felzenszwalb, P.F., Veksler, O.: Tiered scene labeling with dynamic programming. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 3097–3104 (2010)

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

[14] Ilg, E., Saikia, T., Keuper, M., Brox, T.: Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In: European Conference on Computer Vision (ECCV) (2018)

[15] Kakade, S., Teh, Y.W., Roweis, S.T.: An alternate objective function for markovian fields (2002)

[16] Kendall, A., Martirosyan, H., Dasgupta, S., Henry, P., Kennedy, R., Bachrach, A., Bry, A.: End-to-end learning of geometry and context for deep stereo regression. In: IEEE International Conference on Computer Vision (ICCV) (2017)

[17] Keshet, J.: Optimizing the Measure of Performance (2014)

[18] Khamis, S., Fanello, S., Rhemann, C., Kowdle, A., Valentin, J., Izadi, S.: Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In: European Conference on Computer Vision (ECCV). pp. 573–590 (2018)

[19] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[20] Kirillov, A., Schlesinger, D., Forkel, W., Zelenin, A., Zheng, S., Torr, P.H.S., Rother, C.: Efficient likelihood learning of a generic CNN-CRF model for semantic segmentation. CoRR abs/1511.05067 (2015)

[21] Knöbelreiter, P., Pock, T.: Learned collaborative stereo refinement. In: German Conference on Pattern Recognition (GCPR) (2019)

[22] Knöbelreiter, P., Reinbacher, C., Shekhovtsov, A., Pock, T.: End-to-end training of hybrid cnn-crf models for stereo. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2339–2348 (2017)

[23] Kohli, P., Torr, P.H.S.: Measuring uncertainty in graph cut solutions – efficiently computing min-marginal energies using dynamic graph cuts. In: European Conference on Computer Vision (ECCV). pp. 30–43. Springer-Verlag (2006)

[24] Kolmogorov, V., Monasse, P., Tan, P.: Kolmogorov and Zabih's Graph Cuts Stereo Matching Algorithm. Image Processing On Line 4, 220–251 (2014)

[25] Lauritzen, S.L.: Graphical Models. No. 17 in Oxford Statistical Science Series, Oxford Science Publications (1998)

[26] Lee, C.Y., Xie, S., Gallagher, P., Zhang, Z., Tu, Z.: Deeply-supervised nets. In: Artificial intelligence and statistics. pp. 562–570 (2015)

[27] Lin, G., Shen, C., Reid, I.D., van den Hengel, A.: Efficient piecewise training of deep structured models for semantic segmentation. CoRR abs/1504.01013 (2015)

[28] Liu, S., De Mello, S., Gu, J., Zhong, G., Yang, M.H., Kautz, J.: Learning affinity via spatial propagation networks. In: Proceedings of Advances in Neural Information Processing Systems, pp. 1520–1530 (2017)

[29] Luo, W., Schwing, A.G., Urtasun, R.: Efficient deep learning for stereo matching. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5695–5703 (2016)

[30] Marin, D., Tang, M., Ayed, I.B., Boykov, Y.: Beyond gradient descent for regularized segmentation losses. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)

[31] Mayer, N., Ilg, E., Häusser, P., Fischer, P., Cremers, D., Dosovitskiy, A., Brox, T.: A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016)

[32] Mehta, S., Rastegari, M., Caspi, A., Shapiro, L., Hajishirzi, H.: Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. In: European Conference on Computer Vision (ECCV) (2018)

[33] Mensch, A., Blondel, M.: Differentiable dynamic programming for structured prediction and attention (2018)

[34] Munda, G., Shekhovtsov, A., Knöbelreiter, P., Pock, T.: Scalable full flow with learned binary descriptors. In: German Conference on Pattern Recognition (GCPR). pp. 321–332 (2017)

[35] Pal, C.J., Weinman, J.J., Tran, L.C., Scharstein, D.: On learning conditional random fields for stereo - exploring model structures and approximate inference. International Journal of Computer Vision 99(3), 319–337 (2012)

[36] Papandreou, G., Yuille, A.: Perturb-and-map random fields: Reducing random sampling to optimization, with applications in computer vision. In: Nowozin, S., Gehler, P., Jancsary, J., Lampert, C. (eds.) Advanced Structured Prediction. MIT-Press (2014)

[37] Pearl, J.: Reverend Bayes on inference engines: A distributed hierarchical approach. In: AAAI. pp. 133–136 (1982)

[38] Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1988)

[39] Riegler, G., Rüther, M., Bischof, H.: Atgv-net: Accurate depth super-resolution. In: European Conference on Computer Vision (ECCV). pp. 268–284 (2016)

[40] Seki, A., Pollefeys, M.: Sgm-nets: Semi-global matching with neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)

[41] Sontag, D., Jaakkola, T.: Tree block coordinate descent for MAP in graphical models. In: Artificial Intelligence and Statistics. pp. 544–551 (2009)

[42] Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A comparative study of energy minimization methods for markov random fields with smoothness-based priors. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(6), 1068–1080 (June 2008)

[43] Tappen, M., Freeman, W.T.: Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In: IEEE International Conference on Computer Vision (ICCV). pp. 900–906 (2003)

[44] Tulyakov, S., Ivanov, A., Fleuret, F.: Practical deep stereo (pds): Toward applications-friendly deep stereo matching. In: Proceedings of Advances in Neural Information Processing Systems. pp. 5871–5881 (2018)

[45] Van Den Oord, A., Kalchbrenner, N., Kavukcuoglu, K.: Pixel recurrent neural networks. In: ICML. pp. 1747–1756 (2016)

[46] Vemulapalli, R., Tuzel, O., Liu, M.Y., Chellapa, R.: Gaussian conditional random field network for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3224–3233 (2016)

[47] Vogel, C., Knöbelreiter, P., Pock, T.: Learning energy based inpainting for optical flow. In: Asian Conference on Computer Vision (ACCV) (2018)

[48] Wainwright, M.J., Jaakkola, T.S., Willsky, A.S.: Tree-reweighted belief propagation algorithms and approximate ml estimation by pseudo-moment matching. In: International Conference on Artificial Intelligence and Statistics (2003)

[49] Yang, G., Manela, J., Happold, M., Ramanan, D.: Hierarchical deep stereo matching on high-resolution images. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5515–5524 (2019)

[50] Yang, Q., Ji, P., Li, D., Yao, S., Zhang, M.: Fast stereo matching using adaptive guided filtering. Image and Vision Computing 32(3), 202 – 211 (2014)

[51] Yin, Z., Darrell, T., Yu, F.: Hierarchical discrete distribution decomposition for match density estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)

[52] Žbontar, J., LeCun, Y.: Stereo matching by training a convolutional neural network to compare image patches. Journal of Machine Learning Research (2016)

[53] Zhang, F., Prisacariu, V., Yang, R., Torr, P.H.: Ga-net: Guided aggregation net for end-to-end stereo matching. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 185–194 (2019)

[54] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H.: Conditional random fields as recurrent neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1529–1537 (2015)