

# Learning Sparse High Dimensional Filters: Image Filtering, Dense CRFs and Bilateral Neural Networks

Varun Jampani<sup>1</sup>, Martin Kiefel<sup>1,2</sup> and Peter V. Gehler<sup>1,2</sup>

<sup>1</sup>Max Planck Institute for Intelligent Systems, Tübingen, 72076, Germany

<sup>2</sup>Bernstein Center for Computational Neuroscience, Tübingen, 72076, Germany

{varun.jampani, martin.kiefel, peter.gehler}@tuebingen.mpg.de

## Abstract

*Bilateral filters have wide spread use due to their edge-preserving properties. The common use case is to manually choose a parametric filter type, usually a Gaussian filter. In this paper, we will generalize the parametrization and in particular derive a gradient descent algorithm so the filter parameters can be learned from data. This derivation allows to learn high dimensional linear filters that operate in sparsely populated feature spaces. We build on the permutohedral lattice construction for efficient filtering. The ability to learn more general forms of high-dimensional filters can be used in several diverse applications. First, we demonstrate the use in applications where single filter applications are desired for runtime reasons. Further, we show how this algorithm can be used to learn the pairwise potentials in densely connected conditional random fields and apply these to different image segmentation tasks. Finally, we introduce layers of bilateral filters in CNNs and propose bilateral neural networks for the use of high-dimensional sparse data. This view provides new ways to encode model structure into network architectures. A diverse set of experiments empirically validates the usage of general forms of filters.*

## 1. Introduction

Image convolutions are basic operations for many image processing and computer vision applications. In this paper we will study the class of bilateral filter convolutions and propose a general image adaptive convolution that can be learned from data. The bilateral filter [4, 42, 45] was originally introduced for the task of image denoising as an edge preserving filter. Since the bilateral filter contains the spatial convolution as a special case, we will in the following directly state the general case. Given an image  $\mathbf{v} = (\mathbf{v}_1, \dots, \mathbf{v}_n)$ ,  $\mathbf{v}_i \in \mathbb{R}^c$  with  $n$  pixels and  $c$  channels, and for every pixel  $i$ , a  $d$  dimensional feature vector  $\mathbf{f}_i \in \mathbb{R}^d$  (e.g., the  $(x, y)$  position in the image  $\mathbf{f}_i = (x_i, y_i)^\top$ ). The

bilateral filter then computes

$$\mathbf{v}'_i = \sum_{j=1}^n \mathbf{w}_{\mathbf{f}_i, \mathbf{f}_j} \mathbf{v}_j. \quad (1)$$

for all  $i$ . Almost the entire literature refers to the bilateral filter as a synonym of the Gaussian parametric form  $\mathbf{w}_{\mathbf{f}_i, \mathbf{f}_j} = \exp(-\frac{1}{2}(\mathbf{f}_i - \mathbf{f}_j)^\top \Sigma^{-1}(\mathbf{f}_i - \mathbf{f}_j))$ . The features  $\mathbf{f}_i$  are most commonly chosen to be position  $(x_i, y_i)$  and color  $(r_i, g_i, b_i)$  or pixel intensity. To appreciate the edge-preserving effect of the bilateral filter, consider the five-dimensional feature  $\mathbf{f} = (x, y, r, g, b)^\top$ . Two pixels  $i, j$  have a strong influence  $\mathbf{w}_{\mathbf{f}_i, \mathbf{f}_j}$  on each other only if they are close in position *and* color. At edges the color changes, therefore pixels lying on opposite sides have low influence and thus this filter does not blur across edges. This behaviour is sometimes referred to as “image adaptive”, since the filter has a different shape when evaluated at different locations in the image. More precisely, it is the projection of the filter to the two-dimensional image plane that changes, the filter values  $\mathbf{w}_{\mathbf{f}, \mathbf{f}'}$  do not change. The filter itself can be of  $c$  dimensions  $\mathbf{w}_{\mathbf{f}, \mathbf{f}_j} \in \mathbb{R}^c$ , in which case the multiplication in Eq. (1) becomes an inner product. For the Gaussian case the filter can be applied independently per channel. For an excellent review of image filtering we refer to [35].

The filter operation of Eq. (1) is a sparse high-dimensional convolution, a view advocated in [7, 36]. An image  $v$  is not sparse in the spatial domain, we observe pixels values for all locations  $(x, y)$ . However, when pixels are understood in a higher dimensional feature space, e.g.,  $(x, y, r, g, b)$ , the image becomes a sparse signal, since the  $r, g, b$  values lie scattered in this five-dimensional space. This view on filtering is the key difference of the bilateral filter compared to the common spatial convolution. An image edge is not “visible” for a filter in the spatial domain alone, whereas in the 5D space it is. The edge-preserving behaviour is possible due to the higher dimensional operation. Other data can naturally be understood as sparse signals, e.g., 3D surface points.

The contribution of this paper is to propose a general and learnable sparse high dimensional convolution. Our technique builds on efficient algorithms that have been developed to approximate the Gaussian bilateral filter and re-uses them for more general high-dimensional filter operations. Due to its practical importance (see related work in Sec. 2) several efficient algorithms for computing Eq. (1) have been developed, including the bilateral grid [36], Gaussian KD-trees [3], and the permutohedral lattice [2]. The design goal for these algorithms was to provide a) fast runtimes and b) small approximation errors for the Gaussian filter case. The key insight of this paper is to use the permutohedral lattice and use it not as an approximation of a predefined kernel but to freely parametrize its values. We relax the separable Gaussian filter case from [2] and show how to compute gradients of the convolution (Sec. 3) in lattice space. This enables learning the filter from data.

This insight has several useful consequences. We discuss applications where the bilateral filter has been used before: image filtering (Sec. 4) and CRF inference (Sec. 5). Further we will demonstrate how the free parametrization of the filters enables us to use them in deep convolutional neural networks (CNN) and allow convolutions that go beyond the regular spatially connected receptive fields (Sec. 6). For all domains, we present various empirical evaluations with a wide range of applications.

## 2. Related Work

We categorize the related work according to the three different generalizations of this work.

**Image Adaptive Filtering:** The literature in this area is rich and we can only provide a brief overview. Important classes of image adaptive filters include the bilateral filters [4, 45, 42], non-local means [13, 5], locally adaptive regressive kernels [44], guided image filters [24] and propagation filters [38]. The kernel least-squares regression problem can serve as a unified view of many of them [35]. In contrast to the present work that learns the filter kernel using supervised learning, all these filtering schemes use a predefined kernel. Because of the importance of the bilateral filtering to many applications in image processing, much effort has been devoted to derive fast algorithms; most notably [36, 2, 3, 21]. Surprisingly, the only attempt to learn the bilateral filter we found is [25] that casts the learning problem in the spatial domain by rearranging pixels. However, the learned filter does not necessarily obey the full region of influence of a pixel as in the case of a bilateral filter. The bilateral filter also has been proposed to regularize a large set of applications in [9, 8] and the respective optimization problems are parametrized in a bilateral space. In these works the filters are part of a learning system but unlike this work restricted to be Gaussian.

**Dense CRF:** The key observation of [31] is that mean-field inference update steps in densely connected CRFs with Gaussian edge potentials require Gaussian bilateral filtering operations. This enables tractable inference through the application of a fast filter implementation from [2]. This quickly found wide-spread use, *e.g.*, the combination of CNNs with a dense CRF is among the best performing segmentation models [15, 49, 11]. These works combine structured prediction frameworks on top of CNNs, to model the relationship between the desired output variables thereby significantly improving upon the CNN result. Bilateral neural networks, that are presented in this work, provide a principled framework for encoding the output relationship, using the feature transformation inside the network itself thereby alleviating some of the need for later processing. Several works [32, 17, 29, 49, 39] demonstrate how to learn free parameters of the dense CRF model. However, the parametric form of the pairwise term always remains a Gaussian. Campbell *et al.* [14] embed complex pixel dependencies into an Euclidean space and use a Gaussian filter for pairwise connections. This embedding is a pre-processing step and can not directly be learned. In Sec. 5 we will discuss how to learn the pairwise potentials, while retaining the efficient inference strategy of [31].

**Neural Networks:** In recent years, the use of CNNs enabled tremendous progress in a wide range of computer vision applications. Most CNN architectures use spatial convolution layers, which have fixed local receptive fields. This work suggests to replace these layers with bilateral filters, which have a varying spatial receptive field depending on the image content. The equivalent representation of the filter in a higher dimensional space leads to sparse samples that are handled by a permutohedral lattice data structure. Similarly, Bruna *et al.* [12] propose convolutions on irregularly sampled data. Their graph construction is closely related to the high-dimensional convolution that we propose and defines weights on local neighborhoods of nodes. However, the structure of the graph is bound to be fixed and it is not straightforward to add new samples. Furthermore, re-using the same filter among neighborhoods is only possible with their costly spectral construction. Both cases are handled naturally by our sparse convolution. Jaderberg *et al.* [27] propose a spatial transformation of signals within the neural network to learn invariances for a given task. The work of [26] propose matrix backpropagation techniques which can be used to build specialized structural layers such as normalized-cuts. Graham *et al.* [23] propose extensions from 2D CNNs to 3D sparse signals. Our work enables sparse 3D filtering as a special case, since we use an algorithm that allows for even higher dimensional data.

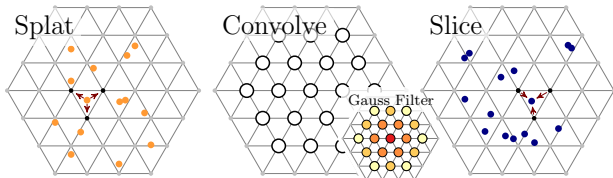


Figure 1. **Schematic of the permutohedral convolution.** Left: splatting the input points (orange) onto the lattice corners (black); Middle: The extent of a filter on the lattice with a  $s = 2$  neighborhood (white circles), for reference we show a Gaussian filter, with its values color coded. The general case has a free scalar/vector parameter per circle. Right: The result of the convolution at the lattice corners (black) is projected back to the output points (blue). Note that in general the output and input points may be different.

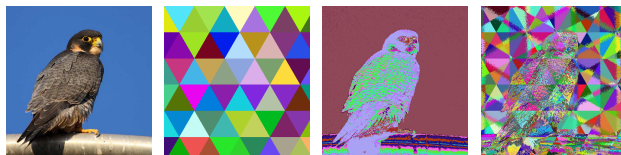
### 3. Learning Sparse High Dimensional Filters

In this section, we describe the main technical contribution of this work, we generalize the permutohedral convolution [2] and show how the filter can be learned from data.

Recall the form of the bilateral convolution from Eq. (1). A naive implementation would compute for every pixel  $i$  all associated filter values  $w_{\mathbf{f}_i, \mathbf{f}_j}$  and perform the summation independently. The view of  $w$  as a linear filter in a higher dimensional space, as proposed by [36], opened the way for new algorithms. Here, we will build on the permutohedral lattice convolution developed in Adams *et al.* [2] for approximate Gaussian filtering. The most common application of bilateral filters use photometric features (XYRGB). We chose the permutohedral lattice as it is particularly designed for this dimensionality, see Fig. 7 in [2] for a speed comparison.

#### 3.1. Permutohedral Lattice Convolutions

We first review the permutohedral lattice convolution for Gaussian bilateral filters from Adams *et al.* [2] and describe its most general case. As before, we assume that every image pixel  $i$  is associated with a  $d$ -dimensional feature vector  $\mathbf{f}_i$ . Gaussian bilateral filtering using a permutohedral lattice approximation involves 3 steps. We begin with an overview of the algorithm, then discuss each step in more detail in the next paragraphs. Fig. 1 schematically shows the three operations for 2D features. First, interpolate the image signal on the  $d$ -dimensional grid plane of the permutohedral lattice, which is called *splatting*. A permutohedral lattice is the tessellation of space into permutohedral simplices. We refer to [2] for details of the lattice construction and its properties. In Fig. 2, we visualize the permutohedral lattice in the image plane, where every simplex cell receives a different color. All pixels of the same lattice cell have the same color. Second, *convolve* the signal on the lattice. And third, retrieve the result by interpolating the signal at the original  $d$ -dimensional feature locations, called *slicing*. For example, if the features used are a combination of position and color  $\mathbf{f}_i = (x_i, y_i, r_i, g_i, b_i)^\top$ , the input signal is mapped



(a) Sample Image (b) Position (c) Color (d) Position, Color

Figure 2. **Visualization of the Permutohedral Lattice.** (a) Input image; Lattice visualizations for different feature spaces: (b) 2D position features:  $0.01(x, y)$ , (c) color features:  $0.01(r, g, b)$  and (d) position and color features:  $0.01(x, y, r, g, b)$ . All pixels falling in the same simplex cell are shown with the same color.

into the 5D cross product space of position and color and then convolved with a 5D tensor. The result is then mapped back to the original space. In practice we use a feature scaling  $Df$  with a diagonal matrix  $D$  and use separate scales for position and color features. The scale determines the distance of points and thus the size of the lattice cells. More formally, the computation is written by  $\mathbf{v}' = S_{\text{slice}} B S_{\text{splat}} \mathbf{v}$  and all involved matrices are defined below. For notational convenience we will assume scalar input signals  $v_i$ , the vector valued case is analogous, the lattice convolution changes from scalar multiplications to inner products.

**Splat:** The splat operation (cf. left-most image in Fig. 1) finds the enclosing simplex in  $\mathcal{O}(d^2)$  on the lattice of a given pixel feature  $\mathbf{f}_i$  and distributes its value  $v_i$  onto the corners of the simplex. How strong a pixel contributes to a corner  $j$  is defined by its barycentric coordinate  $t_{i,j} \in \mathbb{R}$  inside the simplex. Thus, the value  $l_j \in \mathbb{R}$  at a lattice point  $j$  is computed by summing over all enclosed input points; more precisely, we define an index set  $J_i$  for a pixel  $i$ , which contains all the lattice points  $j$  of the enclosing simplex

$$l = S_{\text{splat}} \mathbf{v}; (S_{\text{splat}})_{j,i} = t_{i,j}, \text{ if } j \in J_i, \text{ otherwise } 0. \quad (2)$$

**Convolve:** The permutohedral convolution is defined on the lattice neighborhood  $N_s(j)$  of lattice point  $j$ , *e.g.*, only  $s$  grid hops away. More formally

$$l' = B l; (B)_{j',j} = w_{j,j'}, \text{ if } j' \in N_s(j), \text{ otherwise } 0. \quad (3)$$

An illustration of a two-dimensional permutohedral filter is shown in Fig. 1 (middle). Note that we already presented the convolution in the general form that we will make use of. The work of [2] chooses the filter weights such that the resulting operation approximates a Gaussian blur, which is illustrated in Fig. 1. Further, the algorithm of [2] takes advantage of the separability of the Gaussian kernel. Since we are interested in the most general case, we extended the convolution to include non-separable filters  $B$ .

**Slice:** The slice operation (cf. right-most image in Fig. 1) computes an output value  $v'_i$  for an output pixel  $i'$  again based on its barycentric coordinates  $t_{i,j}$  and sums over the corner points  $j$  of its lattice simplex

$$\mathbf{v}' = S_{\text{slice}} l'; (S_{\text{slice}})_{i,j} = t_{i,j}, \text{ if } j \in J_i, \text{ otherwise } 0 \quad (4)$$

The splat and slice operations take a role of an interpolation between the different signal representations: the irregular and sparse distribution of pixels with their associated feature vectors and the regular structure of the permutohedral lattice points. Since high-dimensional spaces are usually sparse, performing the convolution densely on all lattice points is inefficient. So, for speed reasons, we keep track of the populated lattice points using a hash table and only convolve at those locations.

### 3.2. Learning Permutohedral Filters

The *fixed* set of filter weights  $w$  from [2] in Eq. 3 are designed to approximate a Gaussian filter. However, the convolution kernel  $w$  can naturally be understood as a general filtering operation in the permutohedral lattice space with free parameters. In the exposition above we already presented this general case. As we will show in more detail later, this modification has non-trivial consequences for bilateral filters, CNNs and probabilistic graphical models.

The size of the neighborhood  $N_s(k)$  for the blur in Eq. 3 compares to the filter size of a spatial convolution that considers  $s$  points to either side in all dimensions has  $(2s + 1)^d \in \mathcal{O}(s^d)$  parameters. A comparable filter on the permutohedral lattice with an  $s$  neighborhood is specified by  $(s + 1)^{d+1} - s^{d+1} \in \mathcal{O}(s^d)$  elements (cf. Supplementary material). Thus, both share the same asymptotic size.

By computing the gradients of the filter elements we enable the use of gradient based optimizers, *e.g.*, backpropagation for CNN in the same way that spatial filters in a CNN are learned. The gradients with respect to  $\mathbf{v}$  and the filter weights in  $B$  of a scalar loss  $L$  are:

$$\frac{\partial L}{\partial \mathbf{v}} = S'_{\text{splat}} B' S'_{\text{slice}} \frac{\partial L}{\partial \mathbf{v}'}, \quad (5)$$

$$\frac{\partial L}{\partial (B)_{i,j}} = \left( S'_{\text{slice}} \frac{\partial L}{\partial \mathbf{v}'} \right)_i (S_{\text{splat}} \mathbf{v})_j. \quad (6)$$

Both gradients are needed during backpropagation and in experiments, we use stochastic backpropagation for learning the filter kernel. The permutohedral lattice convolution is parallelizable, and scales linearly with the filter size. Specialized implementations run at interactive speeds in image processing applications [2]. Our implementation in caffe deep learning framework [28] allows arbitrary filter parameters and the computation of the gradients on both CPU and GPU. The code is available online<sup>1</sup>.

## 4. Single Bilateral Filter Applications

In this section we will consider the problem of joint bilateral upsampling [30] as a prominent instance of a single bilateral filter application. See [37] for a recent overview

Upsampling factor	Bicubic	Gaussian	Learned
Color Upsampling (PSNR)			
2x	24.19 / 30.59	33.46 / 37.93	<b>34.05 / 38.74</b>
4x	20.34 / 25.28	31.87 / 35.66	<b>32.28 / 36.38</b>
8x	17.99 / 22.12	30.51 / 33.92	<b>30.81 / 34.41</b>
16x	16.10 / 19.80	29.19 / 32.24	<b>29.52 / 32.75</b>
Depth Upsampling (RMSE)			
8x	0.753	0.753	<b>0.748</b>

Table 1. **Joint bilateral upsampling.** (top) PSNR values corresponding to various upsampling factors and upsampling strategies on the test images of the Pascal VOC12 segmentation / high-resolution 2MP dataset; (bottom) RMSE error values corresponding to upsampling depth images estimated using [18] computed on the test images from the NYU depth dataset [40].

of other bilateral filter applications. Further experiments on image denoising and 3D body mesh denoising are included in the supplementary material, together with details about exact experimental protocols and more visualizations.

### 4.1. Joint Bilateral Upsampling

A typical technique to speed up computer vision algorithms is to compute results on a lower scale and upsample the result to the full resolution. This upsampling step may use the original resolution image as a guidance image. A joint bilateral upsampling approach for this problem setting was developed in [30]. We describe the procedure for the example of upsampling a color image. Given a high resolution gray scale image (the guidance image) and the same image on a lower resolution but in colors, the task is to upsample the color image to the same resolution as the guidance image. Using the permutohedral lattice, joint bilateral upsampling proceeds by splatting the color image into the lattice, using 2D position and 1D intensity as features and the 3D RGB values as the signal. A convolution is applied in the lattice and the result is read out at the pixels of the high resolution image, that is using the 2D position and intensity of the guidance image. The possibility of reading out (slicing) points that are not necessarily the input points is an appealing feature of the permutohedral lattice convolution.

#### 4.1.1 Color Upsampling

For the task of color upsampling, we compare the Gaussian bilateral filter [30] against a learned generalized filter. We experimented with two different datasets: Pascal VOC2012 segmentation [19] using train, val and test splits, and 200 higher resolution (2MP) images from Google image search [1] with 100 train, 50 validation and 50 test images. For training we use the mean squared error (MSE) criterion and perform stochastic gradient descent with a momentum term of 0.9, and weight decay of 0.0005, found using the validation set. In Table 1 we report result in terms of PSNR for the upsampling factors 2x, 4x, 8x and 16x. We compare a standard bicubic interpolation, that does not use a

<sup>1</sup><http://bilateralnn.is.tuebingen.mpg.de>

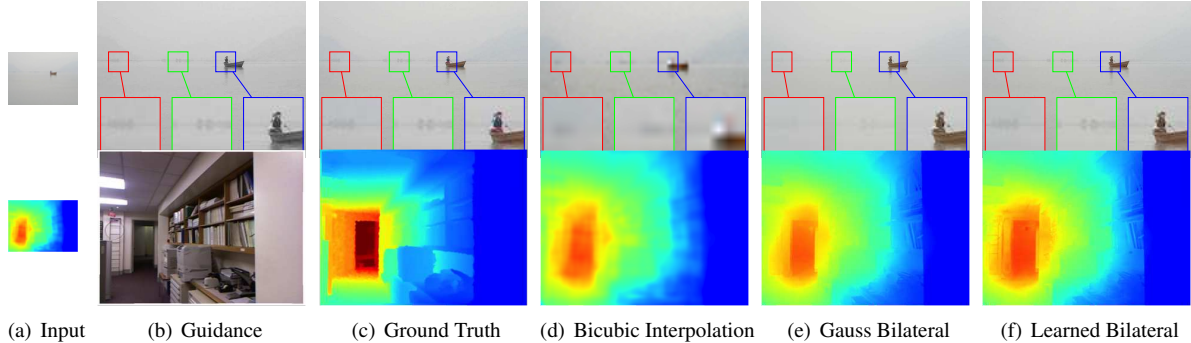


Figure 3. **Guided Upsampling.** Color (top) and depth (bottom)  $8\times$  upsampling results using different methods (best viewed on screen).

guidance image, the Gaussian bilateral filter case (with feature scales optimized on the validation set), and the learned filter. All filters have the same support. For all upsampling factors, joint bilateral Gaussian upsampling outperforms bicubic interpolation and is in turn improved using a learned filter. A result of the upsampling is shown in Fig. 3 and more results are included in the supplementary material. The learned filter recovers finer details in the images

#### 4.1.2 Depth Upsampling

The work [18] predicts depth estimates from single RGB images, we use their results as another joint upsampling task. We use The dataset of [40] that comes with predefined train, validation and test splits. The approach of [18] is a CNN model that produces a result at 1/4th of the input resolution due to down-sampling operations in max-pooling layers. Furthermore, the authors downsample the  $640 \times 480$  images to  $320 \times 240$  as a pre-processing step before CNN convolutions. The final depth result is bicubic interpolated to the original resolution. It is this interpolation that we replace with a Gaussian and learned joint bilateral upsampling. The features are five-dimensional position and color information from the high resolution input image. The filter is learned using the same protocol as for color upsampling minimizing MSE prediction error. The quantitative results are shown in Table 1, the Gaussian filter performs equal to the bicubic interpolation ( $p$ -value 0.311), the learned filter is better ( $p$ -value 0.015). Qualitative results are shown in Fig 3, both joint bilateral upsampling respect image edges in the result. For this [20] and other tasks specialized interpolation algorithms exist, *e.g.*, deconvolution networks [48]. Part of future work is to equip these approaches with bilateral filters.

## 5. Learning Pairwise Potentials in Dense CRFs

The bilateral convolution from Sec. 3 generalizes the class of dense CRF models for which the mean-field inference from [31] applies. The dense CRF models have found wide-spread use in various computer vision applications [43, 10, 49, 47, 46, 11]. Let us briefly review the dense

CRF and then discuss its generalization.

### 5.1. Dense CRF Review

Consider a fully pairwise connected CRF with discrete variables over each pixel in the image. For an image with  $n$  pixels, we have random variables  $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$  with discrete values  $\{x_1, \dots, x_n\}$  in the label space  $x_i \in \{1, \dots, \mathcal{L}\}$ . The dense CRF model has unary potentials  $\psi_u(x) \in \mathbb{R}^{\mathcal{L}}$ , *e.g.*, these can be the output of CNNs. The pairwise potentials in [31] are of the form  $\psi_p^{ij}(x_i, x_j) = \mu(x_i, x_j)k(\mathbf{f}_i, \mathbf{f}_j)$  where  $\mu$  is a label compatibility matrix,  $k$  is a Gaussian kernel  $k(\mathbf{f}_i, \mathbf{f}_j) = \exp(-(\mathbf{f}_i - \mathbf{f}_j)^\top \Sigma^{-1}(\mathbf{f}_i - \mathbf{f}_j))$  and the vectors  $\mathbf{f}_i$  are feature vectors, just alike the ones used for the bilateral filtering, *e.g.*,  $(x, y, r, g, b)$ . The Gibbs distribution for an image  $v$  thus reads  $p(x|v) \propto \exp(-\sum_i \psi_u(x_i) - \sum_{i>j} \psi_p^{ij}(x_i, x_j))$ . Because of dense connectivity, exact MAP or marginal inference is intractable. The main result of [31] is to derive the mean-field approximation of this model and to relate it to bilateral filtering which enables tractable approximate inference. Mean-field approximates the model  $p$  with a fully factorized distribution  $q = \prod_i q_i(x_i)$  and solves for  $q$  by minimizing their KL divergence  $KL(q||p)$ . This results in a fixed point equation which can be solved iteratively  $t = 0, 1, \dots$  to update the marginal distributions  $q_i$ ,

$$q_i^{t+1}(x_i) = \frac{1}{Z_i} \exp\{-\psi_u(x_i) - \underbrace{\sum_{l \in \mathcal{L}} \sum_{j \neq i} \psi_p^{ij}(x_i, l) q_j^t(l)}_{\text{bilateral filtering}}\}. \quad (7)$$

for all  $i$ . Here,  $Z_i$  ensures normalizations of  $q_i$ .

### 5.2. Learning Pairwise Potentials

The proposed bilateral convolution generalizes the class of potential functions  $\psi_p^{ij}$ , since they allow a richer class of kernels  $k(\mathbf{f}_i, \mathbf{f}_j)$  that furthermore can be learned from data. So far, all dense CRF models have used Gaussian potential functions  $k$ , we replace it with the general bilateral convolution and learn the parameters of kernel  $k$ , thus in effect learn the pairwise potentials of the dense CRF. This retains



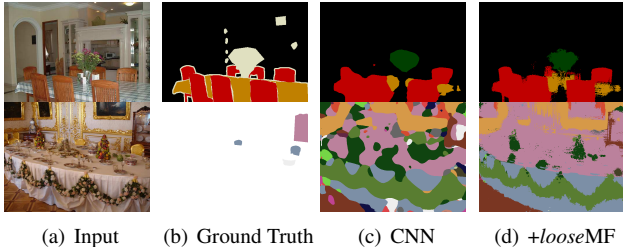


Figure 4. **Segmentation results.** An example result for semantic (top) and material (bottom) segmentation. (c) depicts the unary results before application of MF, (d) after two steps of *loose*-MF with a learned CRF. More examples with comparisons to Gaussian pairwise potentials can be found in the supplementary material.

the desirable properties of this model class – efficient inference through mean-field and the feature dependency of the pairwise potential. In order to learn the form of the pairwise potentials  $k$  we make use of the gradients for filter parameters in  $k$  and use back-propagation through the mean-field iterations [17, 34] to learn them.

The work of [32] derived gradients to learn the feature scaling  $D$  but not the form of the kernel  $k$ , which still was Gaussian. In [14], the features  $f_i$  were derived using a non-parametric embedding into a Euclidean space and again a Gaussian kernel was used. The computation of the embedding was a pre-processing step, not integrated in an end-to-end learning framework. Both aforementioned works are generalizations that are orthogonal to our development and can be used in conjunction.

### 5.3. Experimental Evaluation

We evaluate the effect of learning more general forms of potential functions on two pixel labeling tasks, semantic segmentation of VOC data [19] and material classification [11]. We use pre-trained models from the literature and compare the relative change when learning the pairwise potentials, as in the last Section. For both the experiments, we use multinomial logistic classification loss and learn the filters via back-propagation [17]. This has also been understood as recurrent neural network variants, the following experiments demonstrate the learnability of bilateral filters.

#### 5.3.1 Semantic Segmentation

Semantic segmentation is the task of assigning a semantic label to every pixel. We choose the DeepLab network [15], a variant of the VGGnet [41] for obtaining unaries. The DeepLab architecture runs a CNN model on the input image to obtain a result that is down-sampled by a factor of 8. The result is then bilinear interpolated to the desired resolution and serves as unaries  $\psi_u(x_i)$  in a dense CRF. We use the same Pott’s label compatibility function  $\mu$ , and also use two kernels  $k^1(\mathbf{f}_i, \mathbf{f}_j) + k^2(p_i, p_j)$  with the same features  $\mathbf{f}_i = (x_i, y_i, r_i, g_i, b_i)^\top$  and  $\mathbf{p}_i = (x_i, y_i)^\top$  as in [15].

	+ MF-1step	+ MF-2 step	+ <i>loose</i> MF-2 step
Semantic segmentation (IoU) - CNN [15]: 72.08 / 66.95			
Gauss CRF	+2.48	+3.38	+3.38 / +3.00
Learned CRF	+2.93	+3.71	<b>+3.85 / +3.37</b>
Material segmentation (Pixel Accuracy) - CNN [11]: 67.21 / 69.23			
Gauss CRF	+7.91 / +6.28	+9.68 / +7.35	+9.68 / +7.35
Learned CRF	+9.48 / +6.23	+11.89 / +6.93	<b>+11.91 / +6.93</b>

Table 2. **Improved mean-field inference with learned potentials.** (top) Average IoU score on Pascal VOC12 validation/test data [19] for semantic segmentation; (bottom) Accuracy for all pixels / averaged over classes on the MINC test data [11] for material segmentation.

Thus, the two filters operate in parallel on color & position, and spatial domain respectively. We also initialize the mean-field update equations with the CNN unaries. The only change in the model is the type of the pairwise potential function from Gauss to a generalized form.

We evaluate the result after 1 step and 2 steps of mean-field inference and compare the Gaussian filter versus the learned version (cf. Tab. 2). First, as in [15] we observe that one step of mean field improves the performance by 2.48% in Intersection over Union (IoU) score. However, a learned potential increases the score by 2.93%. The same behaviour is observed for 2 steps: the learned result again adds on top of the raised Gaussian mean field performance. Further, we tested a variant of the mean-field model that learns a separate kernel for the first and second step [34]. This “loose” mean-field model leads to further improvement of the performance. It is not obvious how to take advantage of a loose model in the case of Gaussian potentials.

#### 5.3.2 Material Segmentation

We adopt the method and dataset from [11] for material segmentation task. Their approach proposes the same architecture as in the previous section; a CNN to predict the material labels (*e.g.*, wool, glass, sky, etc.) followed by a densely connected CRF using Gaussian potentials and mean-field inference. We re-use the pre-trained CNN and choose the CRF parameters and Lab color/position features as in [11]. Results for pixel accuracy and class-averaged pixel accuracy are shown in Table 2. Following the CRF validation in [11], we ignored the label ‘other’ for both the training and evaluation. For this dataset, the availability of training data is small, 928 images with only sparsely segment annotations. While this is enough to cross-validate few hyperparameters, we would expect the general bilateral convolution to benefit from more training data. Visuals are shown in Fig. 4 and more are included in the supplementary material.

## 6. Bilateral Neural Networks

Probably the most promising opportunity for the generalized bilateral filter is its use in Convolutional Neural Net-

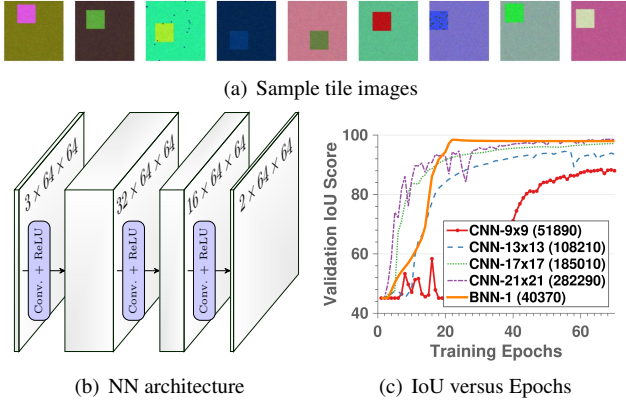


Figure 5. **Segmenting Tiles.** (a) Example tile input images; (b) the 3-layer NN architecture used in experiments. “Conv” stands for spatial convolutions, resp. bilateral convolutions; (c) Training progress in terms of validation IoU versus training epochs.

works. Since we are not restricted to the Gaussian case anymore, we can run several filters sequentially in the same way as filters are ordered in layers in typical spatial CNN architectures. Having the gradients available allows for end-to-end training with backpropagation, without the need for any change in CNN training protocols. We refer to the layers of bilateral filters as “bilateral convolution layers” (BCL). As discussed in the introduction, these can be understood as either linear filters in a high dimensional space or a filter with an image adaptive receptive field. In the remainder we will refer to CNNs that include at least one bilateral convolutional layer as a bilateral neural network (BNN).

What are the possibilities of a BCL compared to a standard spatial layer? First, we can define a feature space  $\mathbf{f}_i \in \mathbb{R}^d$  to define proximity between elements to perform the convolution. This can include color or intensity as in the previous example. We performed a runtime comparison between our current implementation of a BCL and the caffe [28] implementation of a  $d$ -dimensional convolution. For 2D positional features (first row), the standard layer is faster since the permutohedral algorithm comes with an overhead. For higher dimensions  $d > 2$ , the runtime depends on the sparsity; but ignoring the sparsity is quickly leading to intractable runtimes. The permutohedral lattice convolution is in effect a sparse matrix-vector product and thus performs favorably in this case. In the original work [2] it was presented as an approximation to the Gaussian case, here we take the viewpoint of it being the definition of the convolution itself.

Next we illustrate two use cases of BNNs and compare against spatial CNNs. The supplementary material contains further explanatory experiments with examples on MNIST digit recognition.

Dim.-Features	d-dim caffe	BCL
2D- $(x, y)$	$3.3 \pm 0.3 / 0.5 \pm 0.1$	$4.8 \pm 0.5 / 2.8 \pm 0.4$
3D- $(r, g, b)$	$364.5 \pm 43.2 / 12.1 \pm 0.4$	$5.1 \pm 0.7 / 3.2 \pm 0.4$
4D- $(x, r, g, b)$	$30741.8 \pm 9170.9 / 1446.2 \pm 304.7$	$6.2 \pm 0.7 / 3.8 \pm 0.5$
5D- $(x, y, r, g, b)$	out of memory	$7.6 \pm 0.4 / 4.5 \pm 0.4$

Table 3. **Runtime comparison: BCL vs. spatial convolution.** Average CPU/GPU runtime (in ms) of 50 1-neighborhood filters averaged over 1000 images from Pascal VOC. All scaled features  $(x, y, r, g, b) \in [0, 50)$ . BCL includes splatting and splicing operations which in layered networks can be re-used.

## 6.1. An Illustrative Example: Segmenting Tiles

In order to highlight the model possibilities of using higher dimensional sparse feature spaces for convolutions through BCLs, we constructed the following illustrative problem. A randomly colored foreground tile with size  $20 \times 20$  is placed on a random colored background of size  $64 \times 64$ . Gaussian noise with std. dev. 0.02 is added and color values normalized to  $[0, 1]$ , example images are shown in Fig. 5(a). The task is to segment out the smaller tile. Some example images. A pixel classifier can not distinguish foreground from background since the color is random. We train CNNs with three conv/relu layers and varying filters of size  $n \times n$ ,  $n \in \{9, 13, 17, 21\}$  but with The schematic of the architecture is shown in Fig 5(b) (32, 16, 2 filters).

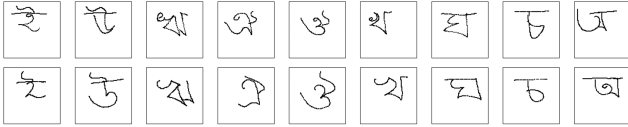
We create 10k training, 1k validation and 1k test images and, use the validation set to choose learning rates. In Fig. 5(c) we plot the validation IoU against training epochs.

Now, we replace all spatial convolutions with bilateral convolutions for a full BNN. The features are  $\mathbf{f}_i = (x_i, y_i, r_i, g_i, b_i)^T$  and the filter has a neighborhood of 1. The total number of parameters in this network is around  $40k$  compared to  $52k$  for  $9 \times 9$  up to  $282k$  for a  $21 \times 21$  CNN. With the same training protocol and optimizer, the convergence rate of BNN is much faster. In this example as in semantic segmentation discussed in the last section, color is a discriminative information for the label. The bilateral convolutions “see” the color difference, the points are already pre-grouped in the permutohedral lattice and the task remains to assign a label to the two groups.

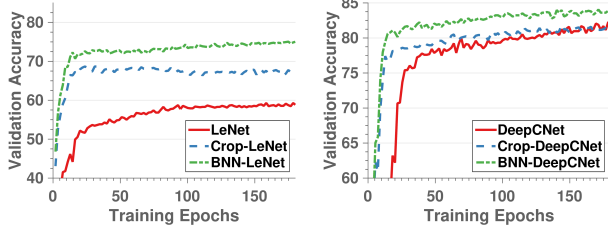
## 6.2. Character Recognition

The results for tile, semantic, and material segmentation when using general bilateral filters mainly improved because the feature space was used to encode useful prior information about the problem (similar RGB of close-by pixels have the same label). Such prior knowledge is often available when structured predictions are to be made but the input signal may also be in a sparse format to begin with. Let us consider handwritten character recognition, one of the prime cases for CNN use.

The Assamese character dataset [6] contains 183 different Indo-Aryan symbols with 45 writing samples per class.



(a) Sample Assamese character images (9 classes, 2 samples each)



(b) LeNet training

(c) DeepCNet training

Figure 6. **Character recognition.** (a) Sample Assamese character images [6]; and training progression of various models with (b) LeNet and (c) DeepCNet base networks.

Some sample character images are shown in Fig. 6(a). This dataset has been collected on a tablet PC using a pen input device and has been pre-processed to binary images of size  $96 \times 96$ . Only about 3% of the pixels contain a pen stroke, which we will denote by  $v_i = 1$ .

A CNN is a natural choice to approach this classification task. We experiment with two CNN architectures for this experiment that have been used for this task, LeNet-7 from [33] and DeepCNet [16, 22]. The LeNet is a shallower network with bigger filter sizes whereas DeepCNet is deeper with smaller convolutions. Both networks are fully specified in the supplementary. In order to simplify the task for the networks we cropped the characters by placing a tight bounding box around them and providing the bounding boxes as input to the networks. We will call these networks “Crop-”LeNet, resp. DeepCNet. For training, we randomly divided the data into 30 writers for training, 6 for validation and the remaining 9 for test. Fig.6(b) and Fig. 6(c) show the training progress for various LeNet and DeepCNet models respectively. DeepCNet is a better choice for this problem and for both cases pre-processing the data by cropping improves convergence.

The input is spatially sparse and the BCL provides a natural way to take advantage of this. For both networks we create a BNN variant (BNN-LeNet and BNN-DeepCNet) by replacing the first layer with a bilateral convolutions using the features  $\mathbf{f}_i = (x_i, y_i)^\top$  and we *only* consider the foreground points  $v_i = 1$ . The values  $(x_i, y_i)$  denote the position of the pixel with respect to the top-left corner of the bounding box around the character. In effect the lattice is very sparse which reduces runtime because the convolutions are only performed on 3% of the points that are actually observed. A bilateral filter has 7 parameters compared to a receptive field of  $3 \times 3$  for the first DeepCNet layer and  $5 \times 5$  for the first LeNet layer. Thus, a BCL with the same number of filters has fewer parameters. The result of

	LeNet	Crop-LeNet	BNN-LeNet	DeepC-Net	Crop-DeepCNet	BNN-DeepCNet
Validation	59.29	68.67	<b>75.05</b>	82.24	81.88	<b>84.15</b>
Test	55.74	69.10	<b>74.98</b>	79.78	80.02	<b>84.21</b>

Table 4. **Results on Assamese character images.** Total recognition accuracy for the different models.

the BCL convolution is then splatted at all points  $(x_i, y_i)$  and passed on to the remaining spatial layers. The convergence behaviour is shown in Fig.6 and again we find faster convergence and also better validation accuracy. The empirical results of this experiment for all tested architectures are summarized in Table 4, with BNN variants clearly outperforming their spatial counterparts.

The absolute results can be vastly improved by making use of virtual examples, *e.g.*, by affine transformations [22]. The purpose of these experiments is to compare the networks on equal grounds while we believe that additional data will be beneficial for both networks. We have no reason to believe that a particular network benefits more.

## 7. Conclusion

We proposed to learn bilateral filters from data. In hindsight, it may appear obvious that this leads to performance improvements compared to a fixed parametric form, *e.g.*, the Gaussian. To understand algorithms that facilitate fast approximate computation of Eq. (1) as a parameterized implementation of a bilateral filter with free parameters is the key insight and enables gradient descent based learning. We relaxed the non-separability in the algorithm from [2] to allow for more general filter functions. There is a wide range of possible applications for learned bilateral filters [37] and we discussed some generalizations of previous work. These include joint bilateral upsampling and inference in dense CRFs. We further demonstrated a use case of bilateral convolutions in neural networks.

The bilateral convolutional layer allows for filters whose receptive field change given the input image. The feature space view provides a canonical way to encode similarity between any kind of objects, not only pixels, but *e.g.*, bounding boxes, segmentation, surfaces. The proposed filtering operation is then a natural candidate to define a filter convolutions on these objects, it takes advantage of sparsity and scales to higher dimensions. Therefore, we believe that this view will be useful for several problems where CNNs can be applied. An open research problem is whether the sparse higher dimensional structure also allows for efficient or compact representations for intermediate layers inside CNN architectures.

**Acknowledgments** We thank Sean Bell for help with the material segmentation experiments. We thank the anonymous reviewers, J. Wulff, L. Sevilla, A. Srikantha, C. Lassner, A. Lehrmann, T. Nestmeyer, A. Geiger, D. Kappler, F. Güney and G. Pons-Moll for their feedback.



## References

- [1] Google Images. <https://images.google.com/>. [Online; accessed 1-March-2015].
- [2] A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, volume 29, pages 753–762. Wiley Online Library, 2010.
- [3] A. Adams, N. Gelfand, J. Dolson, and M. Levoy. Gaussian kd-trees for fast high-dimensional filtering. In *ACM Transactions on Graphics (TOG)*, volume 28, page 21. ACM, 2009.
- [4] V. Aurich and J. Weule. Non-linear Gaussian filters performing edge preserving diffusion. In *DAGM*, pages 538–545. Springer, 1995.
- [5] S. P. Awate and R. T. Whitaker. Higher-order image statistics for unsupervised, information-theoretic, adaptive, image filtering. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 44–51, 2005.
- [6] K. Bache and M. Lichman. UCI machine learning repository, 2013.
- [7] D. Barash. Fundamental relationship between bilateral filtering, adaptive smoothing, and the nonlinear diffusion equation. *Pattern Analysis and Machine Intelligence (T-PAMI)*, 24(6):844–847, 2002.
- [8] J. T. Barron, A. Adams, Y. Shih, and C. Hernández. Fast bilateral-space stereo for synthetic defocus. *CVPR*, 2015.
- [9] J. T. Barron and B. Poole. The Fast Bilateral Solver. *ArXiv e-prints*, Nov. 2015.
- [10] S. Bell, K. Bala, and N. Snavely. Intrinsic images in the wild. *ACM Transactions on Graphics (TOG)*, 33(4):159, 2014.
- [11] S. Bell, P. Upchurch, N. Snavely, and K. Bala. Material recognition in the wild with the materials in context database. *Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [12] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.
- [13] A. Buades, B. Coll, and J.-M. Morel. A non-local algorithm for image denoising. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 60–65, 2005.
- [14] N. D. Campbell, K. Subr, and J. Kautz. Fully-connected CRFs with non-parametric pairwise potential. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1658–1665, 2013.
- [15] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *International Conference on Learning Representations (ICLR)*, 2015.
- [16] D. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3642–3649, 2012.
- [17] J. Domke. Learning graphical model parameters with approximate marginal inference. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(10):2454–2467, 2013.
- [18] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Neural Information Processing Systems (NIPS)*, pages 2366–2374, 2014.
- [19] M. Everingham, L. V. Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL VOC2012 challenge results. 2012.
- [20] D. Ferstl, M. Ruether, and H. Bischof. Variational depth superresolution using example-based edge representations. In *International Conference on Computer Vision (ICCV)*, 2015.
- [21] E. S. Gastal and M. M. Oliveira. Adaptive manifolds for real-time high-dimensional filtering. *ACM Transactions on Graphics (TOG)*, 31(4):33, 2012.
- [22] B. Graham. Spatially-sparse convolutional neural networks. *arXiv preprint arXiv:1409.6070*, 2014.
- [23] B. Graham. Sparse 3D convolutional neural networks. In *British Machine Vision Conference (BMVC)*, 2015.
- [24] K. He, J. Sun, and X. Tang. Guided image filtering. *Pattern Analysis and Machine Intelligence (T-PAMI)*, 35(6):1397–1409, 2013.
- [25] H. Hu and G. de Haan. Trained bilateral filters and applications to coding artifacts reduction. In *Image Processing, 2007. IEEE International Conference on*, volume 1, pages 1–325.
- [26] C. Ionescu, O. Vantzos, and C. Sminchisescu. Matrix back-propagation for deep networks with structured layers. In *International Conference on Computer Vision (ICCV)*, 2015.
- [27] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *Neural Information Processing Systems (NIPS)*, 2015.
- [28] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.
- [29] M. Kiefel and P. V. Gehler. Human pose estimation with fields of parts. In *European Conference on Computer Vision (ECCV)*, pages 331–346. Springer, 2014.
- [30] J. Kopf, M. F. Cohen, D. Lischinski, and M. Uyttendaele. Joint bilateral upsampling. *ACM Transactions on Graphics (TOG)*, 26(3):96, 2007.
- [31] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with Gaussian edge potentials. *Neural Information and Processing Systems (NIPS)*, 2011.
- [32] P. Krähenbühl and V. Koltun. Parameter learning and convergent inference for dense random fields. In *International Conference on Machine Learning (ICML)*, pages 513–521, 2013.
- [33] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [34] Y. Li and R. Zemel. Mean-field networks. In *ICML Workshop on Learning Tractable Probabilistic Models*, 2014.
- [35] P. Milanfar. A tour of modern image filtering. *IEEE Signal Processing Magazine*, 2, 2011.
- [36] S. Paris and F. Durand. A fast approximation of the bilateral filter using a signal processing approach. In *European Conference on Computer Vision (ECCV)*, pages 568–580. Springer, 2006.

- [37] S. Paris, P. Kornprobst, J. Tumblin, and F. Durand. *Bilateral filtering: Theory and applications*. Now Publishers Inc, 2009.
- [38] J.-H. Rick Chang and Y.-C. Frank Wang. Propagated image filtering. In *Computer Vision and Pattern Recognition (CVPR)*, pages 10–18, 2015.
- [39] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *arXiv preprint arXiv:1503.02351*, 2015.
- [40] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgb-d images. In *European Conference on Computer Vision (ECCV)*, pages 746–760. Springer, 2012.
- [41] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.
- [42] S. M. Smith and J. M. Brady. SUSAN – a new approach to low level image processing. *Int. J. Comput. Vision*, 23(1):45–78, May 1997.
- [43] D. Sun, J. Wulff, E. B. Sudderth, H. Pfister, and M. J. Black. A fully-connected layered model of foreground and background flow. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2451–2458, 2013.
- [44] H. Takeda, S. Farsiu, and P. Milanfar. Kernel regression for image processing and reconstruction. *Image Processing, IEEE Transactions on*, 16(2):349–366, 2007.
- [45] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Computer Vision, 1998. Sixth International Conference on*, pages 839–846. IEEE, 1998.
- [46] V. Vineet, G. Sheasby, J. Warrell, and P. H. Torr. Posefield: An efficient mean-field based method for joint estimation of human pose, segmentation and depth. In *EMMCVPR*, 2013.
- [47] V. Vineet, J. Warrell, and P. H. Torr. Filter-based mean-field inference for random fields with higher order terms and product label-spaces. In *European Conference on Computer Vision (ECCV)*, 2012.
- [48] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2528–2535, 2010.
- [49] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. *International Conference on Computer Vision (ICCV)*, 2015.