The Fast Bilateral Solver

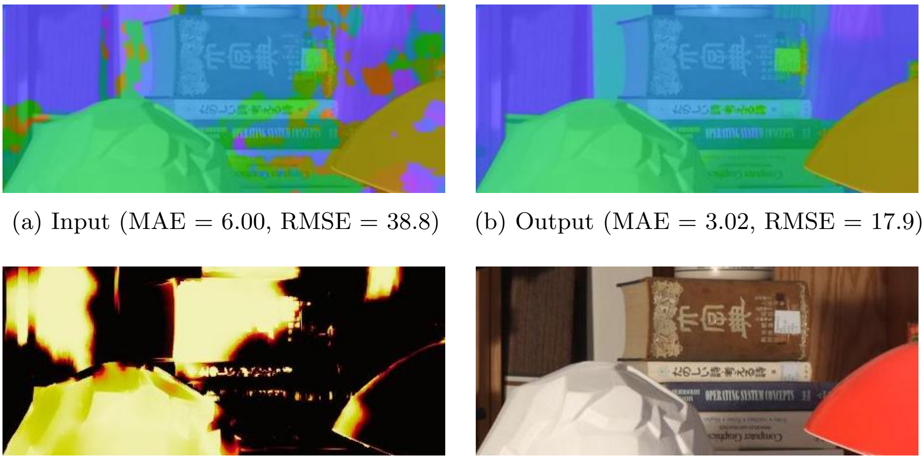
Jonathan T. Barron
barron@google.com

Ben Poole
poole@cs.stanford.edu

Abstract. We present the bilateral solver, a novel algorithm for edge-aware smoothing that combines the flexibility and speed of simple filtering approaches with the accuracy of domain-specific optimization algorithms. Our technique is capable of matching or improving upon state-of-the-art results on several different computer vision tasks (stereo, depth superresolution, colorization, and semantic segmentation) while being 10-1000× faster than baseline techniques with comparable accuracy, and producing lower-error output than techniques with comparable runtimes. The bilateral solver is fast, robust, straightforward to generalize to new domains, and simple to integrate into deep learning pipelines.

1 Introduction

Images of the natural world exhibit a useful prior – many scene properties (depth, color, object category, etc.) are correlated within smooth regions of an image, while differing across discontinuities in the image. Edge-aware smoothing techniques exploit this relationship to propagate signals of interest within, but not



(a) Input (MAE = 6.00, RMSE = 38.8) (b) Output (MAE = 3.02, RMSE = 17.9)

(c) Input Confidence (d) Input Reference

Fig. 1: The bilateral solver can be used to improve depth maps. A depth map (a) from a state-of-the-art stereo method [43] is processed with our robust bilateral solver using a reference RGB image (d). Our output (b) is smooth with respect to the reference image, resulting in a 50% reduction in error.

across edges present in an image. Traditional approaches to edge-aware smoothing apply an image-dependent filter to a signal of interest. Examples of this include joint bilateral filtering [37, 40] and upsampling [20], adaptive manifolds [12], the domain transform [11], the guided filter [17, 16], MST-based filtering [41], and weighted median filtering [30, 44]. These techniques are flexible and computationally efficient, but often insufficient for solving more challenging computer vision tasks. Difficult tasks often necessitate complex iterative inference or optimization procedures that encourage smoothness while maintaining fidelity with respect to some observation. Optimization algorithms of this nature have been used in global stereo [34], depth superresolution [10, 19, 24, 26, 29, 32], colorization [25], and semantic segmentation [6, 22, 28, 45]. These approaches are tailored to their specific task, and are generally computationally expensive. In this work we present an optimization algorithm that is 10-1000$\times$ faster than existing domain-specific approaches with comparable accuracy, and produces higher-quality output than lightweight filtering techniques with comparable runtimes.

Our algorithm is based on the work of Barron *et al.*[2], who presented the idea of using fast bilateral filtering techniques to solve optimization problems in "bilateral-space". This allows for some optimization problems with bilateral affinity terms to be solved quickly, and also guarantees that the solutions to those problems are "bilateral-smooth" — smooth within objects, but not smooth across edges. In this paper we present a new form of bilateral-space optimization which we call *the bilateral solver*, which efficiently solves a regularized least-squares optimization problem to produce an output that is bilateral-smooth and close to the input. This approach has a number of benefits:

General The bilateral solver is a single intuitive abstraction that can be applied to many different problems, while matching or beating the specialized state-of-the-art algorithms for each of these problems. It can be generalized to a variety of loss functions using standard techniques from M-estimation [14].

Differentiable Unlike other approaches for edge-aware smoothness which require a complicated and expensive "unrolling" to perform backpropagation [45], the backward pass through our solver is as simple and fast as the forward pass, allowing it to be easily incorporated into deep learning architectures.

Fast The bilateral solver is expressible as a linear least-squares optimization problem, unlike the non-linear optimization problem used in [2]. This enables a number of optimization improvements including a hierarchical preconditioner and initialization technique that hasten convergence, as well as efficient methods for solving multiple problems at once.

2 Problem Formulation

We begin by presenting the objective and optimization techniques that make up our bilateral solver. Let us assume that we have some per-pixel input quantities $\mathbf{t}$ (the "target" value, see Figure 1a) and some per-pixel confidence of those quantities $\mathbf{c}$ (Figure 1c), both represented as vectorized images. Let us also assume that we have some "reference" image (Figure 1d), which is a normal RGB image. Our goal is to recover an "output" vector $\mathbf{x}$ (Figure 1b), which will resemble the input target where the confidence is large while being smooth and tightly aligned to edges in the reference image. We will accomplish this by constructing an optimization problem consisting of an image-dependent smoothness term that encourages $\mathbf{x}$ to be bilateral-smooth, and a data-fidelity term that minimizes the squared residual between $\mathbf{x}$ and the target $\mathbf{t}$ weighted by our confidence $\mathbf{c}$:

$$\underset{\mathbf{x}}{\text{minimize}} \ \frac{\lambda}{2} \sum_{i,j} \hat{W}_{i,j} \left(x_i - x_j\right)^2 + \sum_i c_i (x_i - t_i)^2 \tag{1}$$

The smoothness term in this optimization problem is built around an affinity matrix $\hat{W}$, which is a bistochastized version of a bilateral affinity matrix W. Each element of the bilateral affinity matrix $W_{i,j}$ reflects the affinity between pixels i and j in the reference image in the YUV colorspace:

$$W_{i,j} = \exp\left(-\frac{\|[[p_i^x, p_i^y] - [p_j^x, p_j^y]\|^2}{2\sigma_{xy}^2} - \frac{(p_i^l - p_j^l)^2}{2\sigma_l^2} - \frac{\|[p_i^u, p_i^v] - [p_j^u, p_j^v]\|^2}{2\sigma_{uv}^2} \right) \tag{2}$$

Where p_i is a pixel in our reference image with a spatial position (p_i^x, p_i^y) and color (p_i^l, p_i^u, p_i^v)[1]. The σ_{xy}, σ_l, and σ_{uv} parameters control the extent of the spatial, luma, and chroma support of the filter, respectively.

This W matrix is commonly used in the bilateral filter [40], an edge-preserving filter that blurs within regions but not across edges by locally adapting the filter to the image content. There are techniques for speeding up bilateral filtering [1, 5] which treat the filter as a "splat/blur/slice" procedure: pixel values are "splatted" onto a small set of vertices in a grid [2, 5] or lattice [1] (a soft histogramming operation), then those vertex values are blurred, and then the filtered pixel values are produced via a "slice" (an interpolation) of the blurred vertex values. These splat/blur/slice filtering approaches all correspond to a compact and efficient factorization of W:

$$W = S^{\mathrm{T}} \bar{B} S \tag{3}$$

Barron *et al.*[2] built on this idea to allow for optimization problems to be "splatted" and solved in bilateral-space. They use a "simplified" bilateral grid and a technique for producing bistochastization matrices $D_{\mathbf{n}}$, $D_{\mathbf{m}}$ that together give the the following equivalences:

$$\hat{W} = S^{\mathrm{T}} D_{\mathbf{m}}^{-1} D_{\mathbf{n}} \bar{B} D_{\mathbf{n}} D_{\mathbf{m}}^{-1} S \qquad S S^{\mathrm{T}} = D_{\mathbf{m}} \tag{4}$$

They also perform a variable substitution, which reformulates a high-dimensional pixel-space optimization problem in terms of the lower-dimensional bilateral-space vertices:

$$\mathbf{x} = S^{\mathrm{T}} \mathbf{y} \tag{5}$$

Where $\mathbf{y}$ is a small vector of values for each bilateral-space vertex, while $\mathbf{x}$ is a large vector of values for each pixel. With these tools we can not only reformulate

[1] To reduce confusion between the Y's in "YUV" and "XY" we refer to luma as "l"

our pixel-space loss function in Eq 1 in bilateral-space, but we can rewrite that bilateral-space loss function in a quadratic form:

$$\underset{\mathbf{y}}{\text{minimize}} \quad \frac{1}{2}\mathbf{y}^{\mathrm{T}}A\mathbf{y} - \mathbf{b}^{\mathrm{T}}\mathbf{y} + c \tag{6}$$

$$A = \lambda(D_{\mathbf{m}} - D_{\mathbf{n}}\bar{B}D_{\mathbf{n}}) + \operatorname{diag}(S\mathbf{c}) \qquad \mathbf{b} = S(\mathbf{c} \circ \mathbf{t}) \qquad c = \frac{1}{2}(\mathbf{c} \circ \mathbf{t})^{\mathrm{T}}\mathbf{t}$$

where $\circ$ is the Hadamard product. A derivation of this reformulation can be found in the supplement. While the optimization problem in Equation 1 is intractably expensive to solve naively, in this bilateral-space formulation optimization can be performed quickly. Minimizing that quadratic form is equivalent to solving a sparse linear system:

$$A\mathbf{y} = \mathbf{b} \tag{7}$$

We can produce a pixel-space solution $\hat{\mathbf{x}}$ by simply slicing the solution to that linear system:

$$\hat{\mathbf{x}} = S^{\mathrm{T}}(A^{-1}\mathbf{b}) \tag{8}$$

With this we can describe our algorithm, which we will refer to as the “bilateral solver.” The input to the solver is a reference RGB image, a target image that contains noisy observed quantities which we wish to improve, and a confidence image. We construct a simplified bilateral grid from the reference image, which is bistochastized as in [2] (see the supplement for details), and with that we construct the A matrix and $\mathbf{b}$ vector described in Equation 6 which are used to solve the linear system in Equation 8 to produce an output image. If we have multiple target images (with the same reference and confidence images) then we can construct a larger linear system in which $\mathbf{b}$ has many columns, and solve for each channel simultaneously using the same A matrix. In this many-target case, if $\mathbf{b}$ is low rank then that property can be exploited to accelerate optimization, as we show in the supplement.

Our pixel-space loss (Eq 1) resembles that of weighted least squares filtering [8, 9, 31], with one critical difference being our use of bilateral-space optimization which allows for efficient optimization even when using a large spatial support in the bilateral affinity, thereby improving the quality of our output and the speed of our algorithm. Our algorithm is similar to the optimization problem that underlies the stereo technique of [2], but with several advantages: Our approach reduces to a simple least-squares problem, which allows us to optimize using standard techniques (we use the preconditioned conjugate gradient algorithm of [36], see the supplement for details). This simple least-squares formulation also allows us to efficiently backpropagate through the solver (Section 3), allowing it to be integrated into deep learning pipelines. This formulation also improves the rate of convergence during optimization, provides guarantees on correctness, allows us to use advanced techniques for preconditioning and initialization (Section 4), and enables robust and multivariate generalizations of our solver (see the supplement).

3 Backpropagation

Integrating any operation into a deep learning framework requires that it is possible to backpropagate through that operation. Backpropagating through global operators such as our bilateral solver is generally understood to be difficult, and is an active research area [18]. Unlike most global smoothing operators, our model is easy to backpropagate through by construction. Note that we do not mean backpropagating *through* a multiplication of a matrix inverse A^{-1}, which would simply be another multiplication by A^{-1}. Instead, we will backpropagate *onto* the A matrix used in the least-squares solve that underpins the bilateral solver, thereby allowing us to backpropagate through the bilateral solver itself.

Consider the general problem of solving a linear system:

$$A\mathbf{y} = \mathbf{b} \tag{9}$$

Where A is an invertible square matrix, and $\mathbf{y}$ and $\mathbf{b}$ are vectors. We can solve for $\hat{\mathbf{y}}$ as a simple least squares problem:

$$\hat{\mathbf{y}} = A^{-1}\mathbf{b} \tag{10}$$

Let us assume that A is symmetric in addition to being positive definite, which is true in our case. Now let us compute some loss with respect to our estimated vector $g(\hat{\mathbf{y}})$, whose gradient will be $\partial g/\partial \hat{\mathbf{y}}$. We would like to backpropagate that quantity onto A and $\mathbf{b}$:

$$\frac{\partial_g}{\partial_{\mathbf{b}}} = A^{-1}\frac{\partial_g}{\partial_{\hat{\mathbf{y}}}} \qquad \frac{\partial_g}{\partial_A} = \left(-A^{-1}\frac{\partial_g}{\partial_{\hat{\mathbf{y}}}}\right)\hat{\mathbf{y}}^{\mathrm{T}} = -\frac{\partial_g}{\partial_{\mathbf{b}}}\hat{\mathbf{y}}^{\mathrm{T}} \tag{11}$$

This can be derived using the implicit function theorem. We see that backpropagating a gradient through a linear system only requires a single least-squares solve. The gradient of the loss with respect to the diagonal of A can be computed more efficiently:

$$\frac{\partial_g}{\partial_{\mathrm{diag}(A)}} = -\frac{\partial_g}{\partial_{\mathbf{b}}} \circ \hat{\mathbf{y}} \tag{12}$$

We will use these observations to backpropagate through the bilateral solver. The bilateral solver takes some input target $\mathbf{t}$ and some input confidence $\mathbf{c}$, and then constructs a linear system that gives us a bilateral-space solution $\hat{\mathbf{y}}$, from which we can "slice" out a pixel-space solution $\hat{\mathbf{x}}$.

$$\hat{\mathbf{y}} = A^{-1}\mathbf{b} \qquad \hat{\mathbf{x}} = S^{\mathrm{T}}\hat{\mathbf{y}} \tag{13}$$

Note that A and $\mathbf{b}$ are both functions of $\mathbf{t}$ and $\mathbf{c}$, though they are not written as such. Let us assume that we have computed some loss $f(\hat{x})$ and its gradient $\partial_f/\partial_{\hat{\mathbf{x}}}$. Remember that the A matrix and $\mathbf{b}$ vector in our linear system are functions of some input signal $\mathbf{t}$ and some input confidence $\mathbf{c}$. Using (11) we can compute the gradient of the loss with respect to the parameters of the linear system within the bilateral solver:

$$\frac{\partial_f}{\partial_{\mathbf{b}}} = A^{-1}\left(S\frac{\partial_f}{\partial_{\hat{\mathbf{x}}}}\right) \qquad \frac{\partial_f}{\partial_{\mathrm{diag}(A)}} = -\frac{\partial_f}{\partial_{\mathbf{b}}} \circ \hat{\mathbf{y}} \tag{14}$$

We need only compute the gradient of the loss with respect to the diagonal of A as opposed to the entirety of A, because the off-diagonal elements of A do not depend on the input signal or confidence. We can now backpropagate the gradient of the loss $f(\hat{\mathbf{x}})$ onto the inputs of the bilateral solver:

$$\frac{\partial_f}{\partial_{\mathbf{t}}} = \mathbf{c} \circ \left(S^{\mathrm{T}}\frac{\partial_f}{\partial_{\mathbf{b}}}\right) \qquad \frac{\partial_f}{\partial_{\mathbf{c}}} = \left(S^{\mathrm{T}}\frac{\partial_f}{\partial_{\mathrm{diag}(A)}}\right) + \left(S^{\mathrm{T}}\frac{\partial_f}{\partial_{\mathbf{b}}}\right) \circ \mathbf{t} \tag{15}$$

To review, the bilateral solver can be viewed as a function which takes in a reference image, some input signal and a per-pixel confidence in that input signal, and produces some smoothed output:

$$\mathrm{output} \leftarrow \mathrm{solver}_{\mathrm{reference}}(\mathrm{target}, \mathrm{confidence}) \tag{16}$$

And we have shown how to backpropagate through the solver:

$$(\nabla\mathrm{target}, \nabla\mathrm{confidence}) \leftarrow \mathrm{backprop}_{\mathrm{reference}}(\nabla\mathrm{output}) \tag{17}$$

Because the computational cost of the backwards pass is dominated by the least squares solve necessary to compute $\partial_f/\partial_{\mathbf{b}}$, computing the backward pass through the solver is no more costly than computing the forward pass. Contrast this with past approaches for using iterative optimization algorithms in deep learning architectures, which create a sequence of layers, one for each iteration in optimization [45]. The backward pass in these networks is a fixed function of the forward pass and so cannot adapt like the bilateral solver to the structure of the error gradient at the output. Furthermore, in these "unrolled" architectures, the output at each iteration (layer) must be stored during training, causing the memory requirement to grow linearly with the number of iterations. In the bilateral solver, the memory requirements are small and independent of the number of iterations, as we only need to store the bilateral-space output of the solver $\hat{\mathbf{y}}$ during training. These properties make the bilateral solver an attractive option for deep learning architectures where speed and memory usage are important.

4 Preconditioning & Initialization

Optimization of the quadratic objective of the bilateral solver can be sped up with improved initialization and preconditioning. In the previous work of [2], the non-linear optimization used a hierarchical technique which lifted optimization into a pyramid space, using a bilateral variant of the image pyramid optimization approach of [3]. This approach cannot be used by our solver, as most linear solvers require a preconditioner where the input is of the same dimensionality as the output. Regardless, the approach of [2] is also suboptimal for our use case, as the simple linear structure of our system allows us to construct more accurate and effective preconditioning and initialization techniques.

To best explain our preconditioning and initialization techniques we must first present baselines techniques for both. We can extract the diagonal of our A matrix to construct a Jacobi preconditioner:

$$\mathrm{diag}(A) = \lambda \left(\mathrm{diag}\left(D_{\mathbf{m}}\right) - \mathrm{diag}(D_{\mathbf{n}})\, \bar{B}_{\mathrm{diag}} \mathrm{diag}(D_{\mathbf{n}}) \right) + S\mathbf{c}$$

This is straightforward to compute, as $D_{\mathbf{m}}$ and $D_{\mathbf{n}}$ are diagonal matrices and $\bar{B}$ has a constant value along the diagonal denoted here as $\bar{B}_{\mathrm{diag}}$. The Jacobi preconditioner is simply the inverse of the diagonal of A:

$$M_{jacobi}^{-1}(\mathbf{y}) = \mathrm{diag}(A)^{-1}\mathbf{y} \tag{18}$$

We can also initialize the state vector $\mathbf{y}$ in our optimization to the value which minimizes the data term in our loss, which has a closed form:

$$\mathbf{y}_{\mathit{flat}} = S(\mathbf{c} \circ \mathbf{t})/S(\mathbf{c}) \tag{19}$$

This preconditioner and initialization technique perform well, as can be seen in Figure 2. But we can improve upon these baseline techniques by constructing hierarchical generalizations of each.

Hierarchical preconditioners have been studied extensively for image interpolation and optimization tasks. Unfortunately, techniques based on image pyramids [38] are not applicable to our task as our optimization occurs in a sparse 5-dimensional bilateral-space. More sophisticated image-dependent or graph based techniques [21, 23, 39] are effective preconditioners, but in our experiments the cost of constructing the preconditioner greatly outweighs the savings provided by the improved conditioning. We will present a novel preconditioner which is similar in spirit to hierarchical basis functions [38] or push-pull interpolation [13], but adapted to our task using the bilateral pyramid techniques presented in [2]. Because of its bilateral nature, our preconditioner is inherently locally adapted and so resembles image-adapted preconditioners [23, 39].

We will use the multiscale representation of bilateral-space presented in [2] to implement our hierarchical preconditioner. This gives us $P(\mathbf{y})$ and $P^{\mathrm{T}}(\mathbf{z})$, which construct a pyramid-space vector $\mathbf{z}$ from a bilateral-space vector $\mathbf{y}$, and collapse $\mathbf{z}$ down to $\mathbf{y}$ respectively (see the supplement for details). To evaluate our preconditioner, we lift our bilateral-space vector into pyramid-space, apply an element-wise scaling of each pyramid coefficient, and then project back onto bilateral-space:

$$M_{hier}^{-1}(\mathrm{y}) = P^{\mathrm{T}} \left(\frac{\mathbf{z}_{weight} \circ P(\mathbf{1}) \circ P(\mathbf{y})}{P(\mathrm{diag}(A))} \right) \tag{20}$$

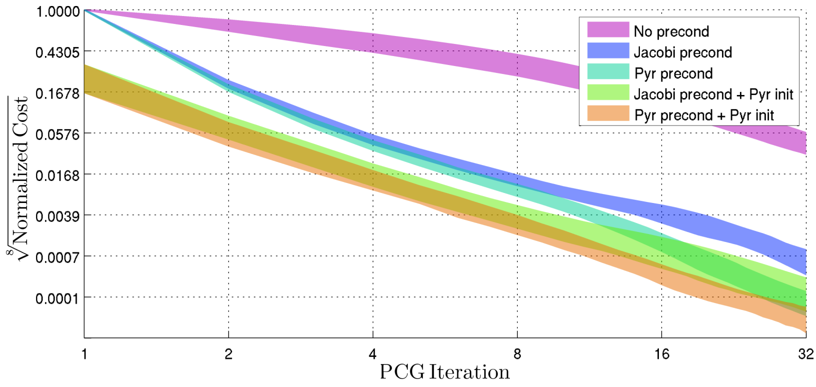


Fig. 2: Our loss during PCG for 20 4-megapixel images, with the loss for each image normalized to $[0, 1]$ and with the 25th-75th percentiles plotted. We see that preconditioning is critical, and that our hierarchical ("Pyr") preconditioning and initialization techniques significantly improve performance over the naive Jacobi preconditioner and "flat" initialization. Note the non-linear y-axis and logarithmic x-axis.

where the division is element-wise. $M_{hier}^{-1}(\cdot)$ includes an ad-hoc element-wise scaling:

$$\mathbf{z}_{weight} = \begin{cases} 1 & \text{if } k = 0 \\ \alpha^{-(\beta+k)} & \text{otherwise} \end{cases} \tag{21}$$

The pyramid-space scaling we use in Equation 20 is proportional to: 1) the number of bilateral-space vertices assigned to each pyramid-space coefficient (computed by lifting a vector of ones), 2) the inverse of the diagonal of the A matrix, computed by lifting and inverting the diagonal of the A matrix, and 3) an exponential weighting of each pyramid-space coefficient according to its level in the pyramid. This per-level scaling $\mathbf{z}_{weight}$ is computed as a function of the level k of each coefficient, which allows us to prescribe the influence that each scale of the pyramid should have in the preconditioner. Note that as the coarser levels are weighed less (ie, as α or β increases) our preconditioner degenerates naturally to the Jacobi preconditioner. In all experiments we use ($\alpha = 2$, $\beta = 5$) for the preconditioner.

This same bilateral pyramid approach can be used to effectively initialize the state before optimization. Rather than simply taking the input target and using it as our initial state as was done in Equation 19, we perform a push-pull filter of that initial state with the pyramid according to the input confidence:

$$\mathbf{y}_{hier} = P^{\mathrm{T}} \left(\frac{\mathbf{z}_{weight} \circ P(S(\mathbf{c} \circ \mathbf{t}))}{P(\mathbf{1})} \right) / P^{\mathrm{T}} \left(\frac{\mathbf{z}_{weight} \circ P(S(\mathbf{c}))}{P(\mathbf{1})} \right) \tag{22}$$

Table 1: Our approach's runtime has a lower mean and variance than that of [2]. Runtimes are from the same workstation, averaged over the 20 4-megapixel images used in [2] for profiling.

	Time (ms)	
Algorithm Component	Barron *et al.*[2]	This Work
Problem Construction	190 ± 167	35 ± 7
Optimization	460 ± 207	152 ± 36
Total	650 ± 266	187 ± 37

Like our hierarchical preconditioner, this initialization degrades naturally to our non-hierarchical initialization in Eq. 19 as α and β increase. In all experiments we use ($\alpha = 4$, $\beta = 0$) for initialization.

See Figure 2 for a visualization of how our hierarchical preconditioning and initialization improve convergence during optimization, compared to the "flat" baseline algorithms. See Table 1 for a comparison of our runtime compared to [2], where we observe a substantial speedup with respect to the solver of [2]. Though the techniques presented here for efficient optimization and initialization are framed in terms of the forward pass through the solver, they all apply directly to the backward pass through the solver described in Section 3, and produce equivalent improvements in speed.

5 Applications

We evaluate our solver on a variety of applications: stereo, depth superresolution, image colorization, and semantic segmentation. Each of these tasks has been the focus of significant research, with specialized techniques having been developed for each problem. For some of these applications (semantic segmentation and stereo) our solver serves as a building block in a larger algorithm, while for others (colorization and depth superresolution) our solver is a complete algorithm. We will demonstrate that our bilateral solver produces results that are comparable to or better than the state-of-the-art for each problem, while being either 1-3 orders of magnitude faster. For those techniques with comparable runtimes, we will demonstrate that the bilateral solver produces higher quality output. Unless otherwise noted, all runtimes were benchmarked on a 2012 HP Z420 workstation (Intel Xeon CPU E5-1650, 3.20GHz, 32 GB RAM), and our algorithm is implemented in standard, single-threaded C++. As was done in [2], the output of our bilateral solver is post-processed by the domain transform [11] to smooth out the blocky artifacts introduced by the simplified bilateral grid, and the domain transform is included in all runtimes. For all results of each application we use the same implementation of the same algorithm with different parameters, which are noted in each sub-section. Parameters are: the spatial bandwidths of the bilateral grid (σ_{xy}, σ_l, σ_{uv}), the smoothness multiplier (λ), the spatial and range bandwidths of the domain transform (σ'_{xy}, σ'_{rgb}). Unless otherwise stated, the bilateral solver is run for 25 iterations of PCG.

5.1 Stereo

We first demonstrate the utility of the bilateral solver as a post-processing procedure for stereo algorithms. Because depth maps produced by stereo algorithms tend to have heavy-tailed noise distributions, we use a variant of our technique called the robust bilateral solver (RBS) with the Geman-McClure loss (described in the supplement). We applied the RBS to the output of the top-performing MC-CNN [43] algorithm on the Middlebury Stereo Benchmark V3 [43]. For comparison, we also evaluated against four other techniques which can or have been used to post-process the output of stereo algorithms. In Table 2 we see that the RBS cuts test- and training-set absolute and RMS errors in half while having little negative effect on the "bad 1%" error metric (the percent of pixels which whose disparities are wrong by more than 1). This improvement is smaller when we only consider non-occluded (NoOcc) as most state-of-the-art stereo algorithms already perform well in the absence of occlusions. The improvement provided by the RBS is more dramatic when the depth maps are visualized, as can be seen in Figure 1 and in the supplement. At submission time our technique achieved a lower test-set MAE and RMSE on the Middlebury benchmark than any published technique[2].

See the supplement for a discussion of how our baseline comparison results were produced, an evaluation of our RBS and our baseline techniques on three additional contemporary stereo algorithms, the parameters settings used in this experiment, how we compute the initial confidence **c** for the RBS, and many visualizations.

[2] `http://vision.middlebury.edu/stereo/eval3/`

Table 2: Our robust bilateral solver significantly improves depth map quality the state-of-the-art MC-CNN[43] stereo algorithm on the Middlebury dataset V3 [33].

Method	All			NoOcc		
	bad 1%	MAE	RMSE	bad 1%	MAE	RMSE
Test Set						
MC − CNN[43]	28.1	17.9	55.0	18.0	3.82	21.3
MC − CNN[43]+RBS	28.2	8.19	29.9	18.9	2.67	15.0
Training Set						
MC-CNN[43]	20.07	5.93	18.36	10.42	1.94	9.07
MC-CNN[43] + TF[41]	29.15	5.67	16.18	20.15	2.17	7.71
MC-CNN[43] + FGF[16]	32.29	5.91	16.32	23.62	2.42	7.98
MC-CNN[43] + WMF[30]	33.37	5.30	15.62	26.29	2.32	8.22
MC-CNN[43] + DT[11]	25.17	5.69	16.53	15.53	2.01	7.72
MC-CNN[43] + RBS (Ours)	19.49	2.81	8.44	11.33	1.40	5.23

5.2 Depth Superresolution

With the advent of consumer depth sensors, techniques have been proposed for upsampling noisy depth maps produced by these sensors using a high-resolution RGB reference image [10, 19, 24, 26, 29, 32, 4, 27, 31]. Other techniques have been developed for post-processing depth maps in other contexts [41, 30, 35], and many general edge-aware upsampling or filtering techniques can be used for this task[20, 11, 17, 16, 44]. We present an extensive evaluation of the bilateral solver against these approaches for the depth superresolution task. Given a noisy input depth map and an RGB reference image, we resize the depth map to be the size of the reference image with bicubic interpolation and then apply the bilateral solver or one of our baseline techniques. The hyperparameters used by the solver for all experiments are: $\sigma_{xy} = 8$, $\sigma_l = 4$, $\sigma_{uv} = 3$, $\sigma'_{xy} = \sigma'_{rgb} = 16$, $\lambda = 4^{f-1/2}$ (where f is the upsampling factor) and 15 iterations of PCG. Our confidence $\mathbf{c}$ is a Gaussian bump ($\sigma = f/4$) modeling the support of each low-resolution pixel in the upsampled image. To evaluate our model, we use the depth superresolution benchmark of [10] which is based on the Middlebury stereo dataset [34]. Our performance can be see in Table 3 and Figure 3, with more detailed results in the supplement. The bilateral solver produces the third-lowest error rate for this task, though the two better-performing tasks [24, 26] use large amounts of external training data and so have an advantage over our technique, which uses no learning for this experiment. Our approach is 600×, 1200×, and 3000× faster than the three most accurate techniques. The techniques with speeds comparable to or better than the bilateral solver [11, 41, 16, 31] produce error rates that are 25-40% greater than our approach. The bilateral solver represents a effective combination of speed and accuracy, while requiring no training or learning. See the supplement for a more detailed table, a discussion of baselines and runtimes, and many visualizations.

Table 3: Performance on the depth superresolution task of [10]. Runtimes in gray were not computed on our reference hardware, and algorithms which use external training data are indicated with a dagger.

Method	Err	Time (sec)
Nearest Neighbor	7.26	0.003
Bicubic	5.91	0.007
†Kiechle *et al.*[19]	5.86	450
Bilinear	5.16	0.004
Liu *et al.* [27]	5.10	16.60
Shen *et al.* [35]	4.24	31.48
Diebel & Thrun [7]	3.98	–
Chan *et al.*[4]	3.83	3.02
GuidedFilter[17, 10]	3.76	23.89
Min *et al.* [31]	3.74	0.383
†Lu & Forsyth[29]	3.69	20
Park *et al.*[32]	3.61	24.05
⋮		
Domain Transform [11]	3.56	0.021
Ma *et al.* [30]	3.49	18
GuidedFilter(Matlab)[17]	3.47	0.434
Zhang *et al.* [44]	3.45	1.346
FastGuidedFilter[16]	3.41	0.225
Yang 2015 [41]	3.41	0.304
Yang *et al.* 2007 [42]	3.25	–
Farbman *et al.* [9]	3.19	6.11
JBU [1, 20]	3.14	1.98
Ferstl *et al.*[10]	2.93	140
†Li *et al.*[26]	2.56	700
†Kwon *et al.*[24]	1.21	300
BS (Ours)	2.70	0.234

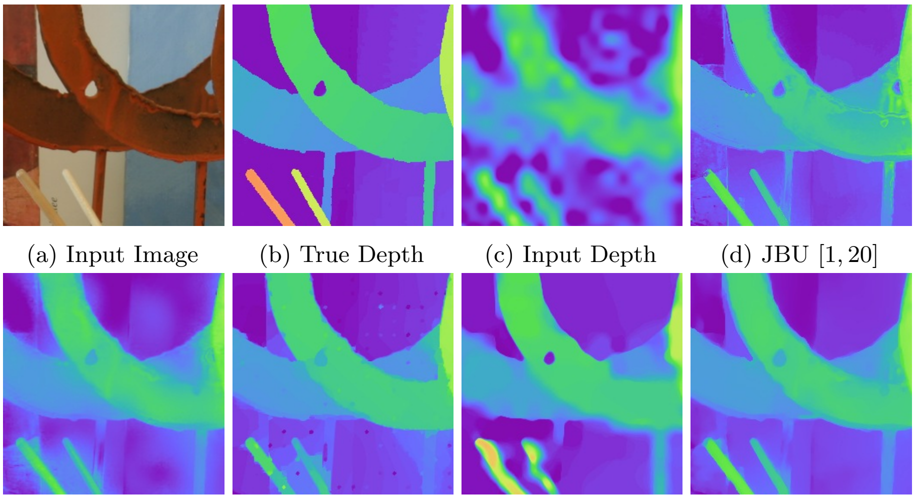


Fig. 3: Partial results for the depth superresolution task of [10], see the supplement for exhaustive visualizations.

5.3 Colorization

Colorization is the problem of introducing color to a grayscale image with a small amount of user input or outside information, for the purpose of improving black-and-white films or photographs. Levin *et al.*[25] presented an effective technique for this task by formulating and solving a specialized optimization problem. We can solve the same task using our bilateral solver: we use the grayscale image as the input reference image and the UV channels of the user-annotated scribbles as the input target images, with a confidence image that is 1 where the user has scribbled and 0 everywhere else. We then construct our final output by combining the grayscale image with our output UV images, and converting from YUV to

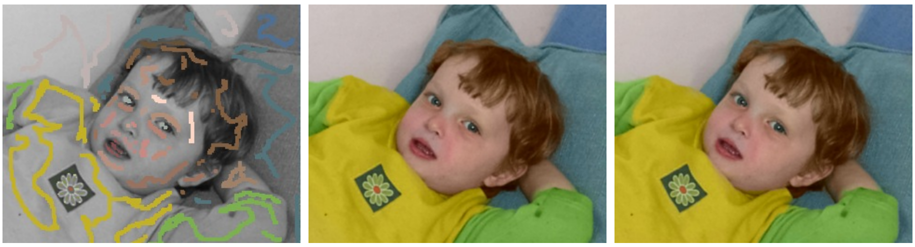


Fig. 4: Results for the user-assisted colorization task. Our bilateral solver produces comparable results to the technique of Levin *et al.*[25] while being 95× faster.

RGB. Our results can be seen in Figure 4, where we see that our output is nearly indistinguishable from that of [25]. The important distinction here is speed, as the approach of [25] take 80.99 seconds per megapixel while our approach takes 0.854 seconds per megapixel — a 95× speedup. For all results our parameters are $\sigma_{xy} = \sigma_l = \sigma_{uv} = 4$, $\lambda = 0.5$, $\sigma'_{xy} = 4$, $\sigma'_{rgb} = 8$. More results can be see in the supplement.

5.4 Semantic Segmentation

Semantic segmentation is the problem of assigning a category label to each pixel in an image. State-of-the-art approaches to semantic segmentation use large convolutional neural networks (CNNs) to map from pixels to labels [6, 28]. The output of these CNNs is often smoothed across image boundaries, so recent approaches refine their output with a CRF ([6, 45]). These CRF-based approaches improve per-pixel labeling accuracy, but this accuracy comes at a computational cost: inference in a fully connected CRF on a 500×500 image can take up to a second (see Table 4). To evaluate whether the bilateral solver could improve the efficiency of semantic segmentation pipelines, we use it instead of the CRF component in two state-of-the-art models: DeepLab-LargeFOV [6] and CRF-RNN [45]. The DeepLab model consists of a CNN trained on Pascal VOC12 and then augmented with a fixed dense CRF. The CRF-RNN model generalizes the CRF with a recurrent neural network, and trains this component jointly with the CNN on Pascal and MSCOCO.

As the bilateral solver operates on real-valued inputs, it is not immediately clear how to map it onto the discrete optimization problem of the dense CRF. For each class, we compute the 21-channel class probability image from the CNN

Table 4: Semantic segmentation results and runtimes on Pascal VOC 2012 validation set. The bilateral solver improves performance over the CNN output while being substantially faster than the CRF-based approaches. "Post" is the time spent post-processing the CNN output, which is the dense CRF for DeepLab, and the generalized CRF-RNN component for CRF-RNN. FCN is the convolutional neural network component of the CRF-RNN model. *DeepLab-LargeFOV model from [6] trained on Pascal VOC 2012 training data augmented with data from [15]. †CRF-RNN model from [45] trained with additional MSCOCO data and evaluated on reduced Pascal validation set of 346 images.

Method	IOU(%)	Time (ms)		
		CNN	Post	Total
DeepLab	62.25*	58	0	58
DeepLab + CRF	67.64*	58	918	976
DeepLab + BS(Ours)	66.00*	58	111	169
CNN	69.60†	715	0	715
CRF − RNN	72.96†	715	2214	2929
CNN + BS(Ours)	70.68†	715	217	913

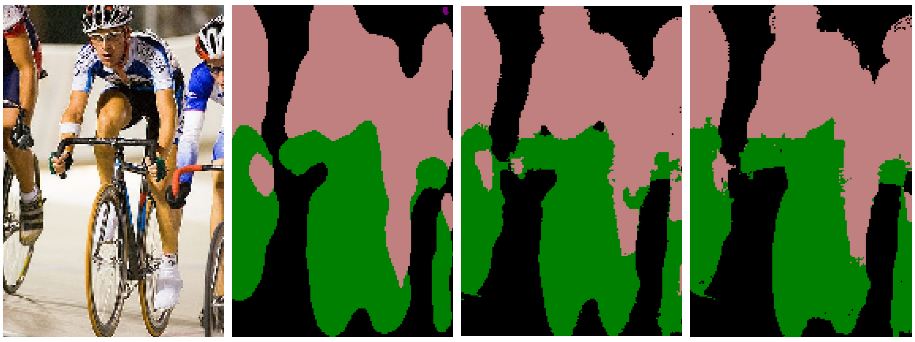


Fig. 5: Using the DeepLab CNN-based semantic segmentation algorithm [6] (5b) as input our bilateral solver can produce comparable edge-aware output (5d) to the DenseCRF [22] used in [6] (5c), while being 8× faster.

outputs of either the DeepLab or CRF-RNN model. As many class probability maps are zero across the entire image, the resulting $\mathbf{b}$ matrix in the bilateral solver is often low-rank, allowing us to solve a reduced linear system to recover the smoothed class probability maps (see the supplement). This approach produces nearly identical output, with a 5× speedup on average.

We applied our bilateral solver to the class probability maps using uniform confidence. The resulting discrete segmentations are more accurate and qualitatively smoother than the CNN outputs, despite our per-channel smoothing providing no explicit smoothness guarantees on the argmax of the filtered per-class probabilities (Table. 4, Figure 5). The bilateral solver is $8 - 10\times$ faster than the CRF and CRF-RNN approaches when applied to the same inputs (Table 4). Although the bilateral solver performs slightly worse than the CRF-based approaches, its speed suggests that it may be a useful tool in contexts such as robotics and autonomous driving, where low latency is necessary.

6 Conclusion

We have presented the bilateral solver, a flexible and fast technique for inducing edge-aware smoothness. We have demonstrated that the solver can produce or improve state-of-the-art results on a variety of different computer vision tasks, while being faster or more accurate than other approaches. Its speed and generality suggests that the bilateral solver is a useful tool in the construction of computer vision algorithms and deep learning pipelines.

References

1. Adams, A., Baek, J., Davis, M.A.: Fast high-dimensional filtering using the permutohedral lattice. Eurographics (2010)
2. Barron, J.T., Adams, A., Shih, Y., Hernández, C.: Fast bilateral-space stereo for synthetic defocus. CVPR (2015)
3. Barron, J.T., Malik, J.: Shape, illumination, and reflectance from shading. TPAMI (2015)
4. Chan, D., Buisman, H., Theobalt, C., Thrun, S.: A noise-aware filter for real-time depth upsampling. ECCV Workshops (2008)
5. Chen, J., Paris, S., Durand, F.: Real-time edge-aware image processing with the bilateral grid. SIGGRAPH (2007)
6. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Semantic image segmentation with deep convolutional nets and fully connected crfs. ICLR (2015)
7. Diebel, J., Thrun, S.: An application of markov random fields to range sensing. NIPS (2005)
8. Elad, M.: On the origin of the bilateral filter and ways to improve it. Transactions on Image Processing (2002)
9. Farbman, Z., Fattal, R., Lischinski, D., Szeliski, R.: Edge-preserving decompositions for multi-scale tone and detail manipulation. SIGGRAPH (2008)
10. Ferstl, D., Reinbacher, C., Ranftl, R., Ruether, M., Bischof, H.: Image guided depth upsampling using anisotropic total generalized variation. ICCV (2013)
11. Gastal, E.S.L., Oliveira, M.M.: Domain transform for edge-aware image and video processing. SIGGRAPH (2011)
12. Gastal, E.S.L., Oliveira, M.M.: Adaptive manifolds for real-time high-dimensional filtering. SIGGRAPH (2012)
13. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. SIGGRAPH (1996)
14. Hampel, F.R., Ronchetti, E.M., Rousseeuw, P.J., Stahel, W.A.: Robust Statistics - The Approach Based on Influence Functions. Wiley (1986)
15. Hariharan, B., Arbelaez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. ICCV (2011)
16. He, K., Sun, J.: Fast guided filter. CoRR abs/1505.00996 (2015)
17. He, K., Sun, J., Tang, X.: Guided image filtering. ECCV (2010)
18. Ionescu, C., Vantzos, O., Sminchisescu, C.: Matrix backpropagation for deep networks with structured layers. ICCV (2015)
19. Kiechle, M., Hawe, S., Kleinsteuber, M.: A joint intensity and depth co-sparse analysis model for depth map super-resolution. ICCV (2013)
20. Kopf, J., Cohen, M.F., Lischinski, D., Uyttendaele, M.: Joint bilateral upsampling. SIGGRAPH (2007)
21. Koutis, I., Miller, G.L., Tolliver, D.: Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing. CVIU (2011)
22. Krähenbühl, P., Koltun, V.: Efficient inference in fully connected crfs with gaussian edge potentials. NIPS (2011)
23. Krishnan, D., Fattal, R., Szeliski, R.: Efficient preconditioning of laplacian matrices for computer graphics. SIGGRAPH (2013)
24. Kwon, H., Tai, Y.W., Lin, S.: Data-driven depth map refinement via multi-scale sparse representation. CVPR (2015)
25. Levin, A., Lischinski, D., Weiss, Y.: Colorization using optimization. SIGGRAPH (2004)

26. Li, Y., Xue, T., Sun, L., Liu, J.: Joint example-based depth map super-resolution. ICME (2012)
27. Liu, M.Y., Tuzel, O., Taguchi, Y.: Joint geodesic upsampling of depth images. CVPR (2013)
28. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. CVPR (2015)
29. Lu, J., Forsyth, D.: Sparse depth super resolution. CVPR (2015)
30. Ma, Z., He, K., Wei, Y., Sun, J., Wu, E.: Constant time weighted median filtering for stereo matching and beyond. ICCV (2013)
31. Min, D., Choi, S., Lu, J., Ham, B., Sohn, K., Do, M.N.: Fast global image smoothing based on weighted least squares. Transactions on Image Processing (2014)
32. Park, J., Kim, H., Tai, Y.W., Brown, M.S., Kweon, I.: High quality depth map upsampling for 3d-tof cameras. ICCV (2011)
33. Scharstein, D., Hirschmüller, H., Kitajima, Y., Krathwohl, G., Nesic, N., Wang, X., Westling, P.: High-resolution stereo datasets with subpixel-accurate ground truth. GCPR (2014)
34. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. IJCV (2002)
35. Shen, X., Zhou, C., Xu, L., Jia, J.: Mutual-structure for joint filtering. ICCV (2015)
36. Shewchuk, J.R.: An introduction to the conjugate gradient method without the agonizing pain. Tech. rep., Carnegie Mellon University (1994)
37. Smith, S.M., Brady, J.M.: Susan - a new approach to low level image processing. IJCV (1997)
38. Szeliski, R.: Fast surface interpolation using hierarchical basis functions. TPAMI (1990)
39. Szeliski, R.: Locally adapted hierarchical basis preconditioning. SIGGRAPH (2006)
40. Tomasi, C., Manduchi, R.: Bilateral filtering for gray and color images. ICCV (1998)
41. Yang, Q.: Stereo matching using tree filtering. PAMI (2015)
42. Yang, Q., Yang, R., Davis, J., Nistér, D.: Spatial-depth super resolution for range images. CVPR (2007)
43. Zbontar, J., LeCun, Y.: Computing the stereo matching cost with a convolutional neural network. CVPR (2015)
44. Zhang, Q., Xu, L., Jia, J.: 100+ times faster weighted median filter (wmf). CVPR (2014)
45. Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.: Conditional random fields as recurrent neural networks. ICCV (2015)