Bilateral Guided Upsampling

Jiawen Chen* Andrew Adams* Neal Wadhwa† Samuel W. Hasinoff*

Google Research* MIT CSAIL†

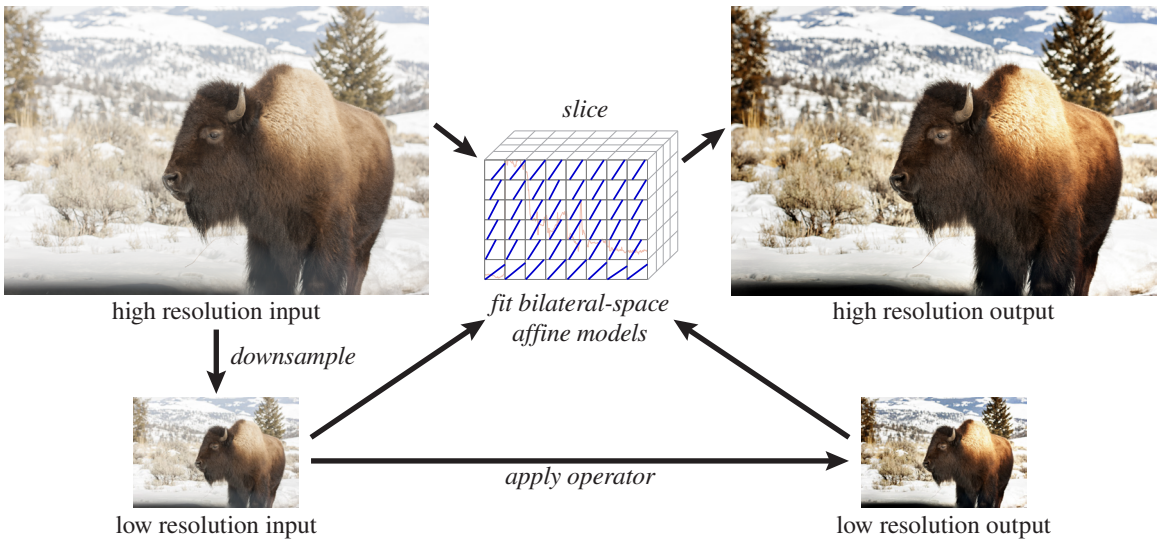

Figure 1: *Our technique accelerates black-box image processing operators by fitting bilateral-space affine models to a low-resolution input/output pair. We then produce a high-resolution output by evaluating the models on the high-resolution input. Fitting the affine models takes milliseconds, even on a mobile phone, and applying them can be done in a simple GPU shader, reducing the total cost to the cost of running the operator at greatly reduced resolution. In this example, we faithfully reproduce a complex sequence of Adobe Photoshop filters that increase local contrast, boost saturation, and remove haze.*

Abstract

We present an algorithm to accelerate a large class of image processing operators. Given a low-resolution reference input and output pair, we model the operator by fitting local curves that map the input to the output. We can then produce a full-resolution output by evaluating these low-resolution curves on the full-resolution input. We demonstrate that this faithfully models state-of-the-art operators for tone mapping, style transfer, and recoloring. The curves are computed by lifting the input into a bilateral grid and then solving for the 3D array of affine matrices that best maps input color to output color per x, y, intensity bin. We enforce a smoothness term on the matrices which prevents false edges and noise amplification. We can either globally optimize this energy, or quickly approximate a solution by locally fitting matrices and then enforcing smoothness by blurring in grid space. This latter option reduces to joint bilateral upsampling [Kopf et al. 2007] or the guided filter [He et al. 2013], depending on the choice of parameters. The cost of running the algorithm is reduced to the cost of running the original algorithm at greatly reduced resolution, as fitting the curves takes about 10 ms on mobile devices, and 1-2 ms on desktop CPUs, and evaluating the curves can be done with a simple GPU shader.

Keywords: bilateral grid, fast image processing, local curve

Concepts: •**Computing methodologies** → **Image manipulation;** *Computational photography;*

*{jiawen,abadams,hasinoff}@google.com

†nwadhwa@mit.edu

ACM Reference Format
Chen, J., Adams, A., Wadhwa, N., Hasinoff, S. 2016. Bilateral Guided Upsampling. ACM Trans. Graph. 35, 6, Article 203 (November 2016), 8 pages. DOI = 10.1145/2980179.2982423 http://doi.acm.org/10.1145/2980179.2982423.

1 Introduction

Imaging operators can improve photographs in numerous ways. They can be used to remove haze [Kim et al. 2013], compress dynamic range [Paris et al. 2011], colorize [Levin et al. 2004], stylize [Aubry et al. 2014], or enhance details [Farbman et al. 2008]. However, most photographs are captured and processed entirely on mobile phones, which have limited computational capabilities stemming from their strict power budget. This makes most algorithms from the literature too slow to deploy on mobile devices.

A simple way to accelerate an operator is to apply it at low resolution, then upsample the result, ideally using some method which reintroduces detail present in the high-resolution original lost in downsampling. Two such methods are joint bilateral upsampling [Kopf et al. 2007] and the fast guided filter [He and Sun 2015]. Either method can be fast enough to run on a mobile device, but they can only reproduce a limited range of operators (see figure 8).

SA '16 Technical Papers,, December 05 - 08, 2016, Macao
ISBN: 978-1-4503-4514-9/16/12
DOI: http://dx.doi.org/10.1145/2980179.2982423

We present *bilateral guided upsampling*, which unifies and generalizes these two methods. We first make the observation that for many imaging operators, nearby pixels that have similar colors in the input also have similar colors in the output. To do otherwise would introduce a new edge, a halo, or noise where there was no such feature in the input. This is equivalent to saying that the operator is a smooth function in *bilateral space* [Barron et al. 2015].

For a black-box image operator and a specific input image, we can learn this smooth function by running the operator at reduced resolution and then fitting locally-affine models in bilateral space. That is, for each tile of the low-resolution image and for each intensity range, we learn an affine relationship between input and output. We constrain these affine matrices to vary smoothly, either by solving for them globally with a smoothness term, or more cheaply by fitting them using overlapping windows. Trilinearly interpolating into this array of affine models as a function of position and intensity acts as a piecewise-quadratic local curve that can be applied to the full-resolution input.

If we reduce our model to a single intensity range for each spatial tile, then we fit lines rather than curves, and this is precisely the fast guided filter. If we reduce our affine matrices to their constant terms, then this is precisely joint bilateral upsampling.

We found that we can run most operators at one-eighth resolution (thus doing 64 times less work) and still produce comparable full-resolution outputs. Fitting the affine models takes on the order of milliseconds and producing the full-resolution output can then be done in a minimal OpenGL shader (included in the supplemental material). We therefore greatly accelerate the original operator.

Our approach requires that the operator be somewhat scale-invariant. Applying it at low resolution must yield useful information about how to apply it at full resolution. This means that we poorly model operators for which this assumption does not hold, such as denoising or deblurring.

This paper presents the theory behind this approach (section 3), the algorithm we propose (section 4), and its performance on a range of imaging operators (section 5). The algorithm is fast, effective, and simple. We include source code for both local and global algorithms along with full-resolution images as supplemental material.

2 Related Work

The bilateral filter is a non-linear, edge-preserving filter, proposed by Tomasi and Manduchi [1998]. Many techniques and data structures have been proposed for speeding up the bilateral filter [Paris and Durand 2006; Chen et al. 2007; Adams et al. 2010; Gastal and Oliveira 2011; Gastal and Oliveira 2012]. Chen et al. [2007] introduced the bilateral grid, a data structure that enables edge-aware manipulation of images. Chen et al. used a regular grid of samples, but this representation has been extended to other layouts such as k-d trees and high-dimensional lattices [Adams 2011]. Barron et al. [2015] introduced the concept of "bilateral-space" optimization, showing that stereo can be solved quickly by projecting the problem onto the bilateral grid. Barron and Poole [2016] generalize this to colorization, depth super-resolution and semantic segmentation.

Kopf et al. [2007] proposed *joint bilateral upsampling*. They use this technique to upsample low-resolution colorization, tone mapping, and depth maps into high-resolution ones that respect color discontinuities of the underlying scene. While their approach results in a piecewise-smooth image, we solve for a smooth *transform* between pairs of images.

Our affine model is inspired by the *guided filter* [He et al. 2013], which is a fast, edge-preserving filter. This method works by fitting a

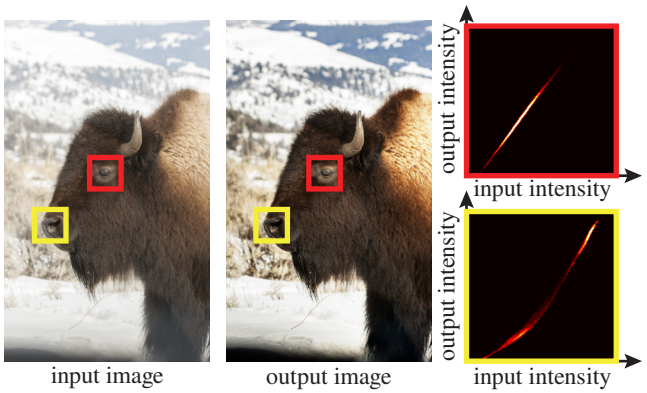

Figure 2: *Left: input image and output after a sequence of edits in Photoshop. Right: input/output scatter plots of* 128×128 *patches of the bison's eye and nose. For many operators, nearby pixels with similar color in the input also have similar color in the output. This means that locally, complex algorithms can be approximated by a smooth curve that maps input values to output values.*

2D array of affine functions between overlapping patches in a guide image and an input image. These affine functions are then applied to the input image. This smooths the input image while respecting the edges in the guide image. We instead fit a 3D array of affine functions on the image's bilateral-grid representation, meaning that within a single patch, we fit different affine functions per intensity range. In the degenerate case where the grid has a single intensity bin, our representation is equivalent to that of the guided filter. Figure 8 demonstrates the increased expressiveness of our model.

Yuan and Sun [2011] solve the related problem of enhancing a high-resolution JPEG produced on-camera using a low-resolution pseudo-raw file processed offline. Their tone-mapping step uses a locally-affine model to relate patches from the processed RAW file to the original JPEG. Like the guided filter, this can only express local lines, not curves.

Gharbi et al. [2015] introduced the concept of a *transform recipe*. Whereas we seek to avoid computing the full-resolution operator entirely, Gharbi et al. instead offload that work to a cloud server, and send back a multi-scale model similar to but richer than ours describing the effect of the operator so that it can be reproduced on device. Compared to transform recipes, we are faster, simpler, and run entirely on-device. However, we cannot model the effects of an operator on the highest frequencies as faithfully, because we never run the operator at full resolution. Despite this, our PSNR is competitive with their technique (figure 8).

3 Local Properties of Imaging Operators

In this section we analyze the relationship between an input image and the output image produced by the action of an operator. Figure 2 contains a scatter plot illustrating such a mapping for a particular operator run on a particular image.

Consider a small patch of the input, and its corresponding patch in the output. We will attempt to capture the action of the operator by fitting a model over the patch that predicts the output given the input. For a 1×1 patch, we can fit any operator, no matter how exotic, with a constant model that simply states the output intensity. Over a slightly larger patch, a constant model is only correct where the output image is featureless. To more accurately model the relationship we can consider the next term in the Taylor

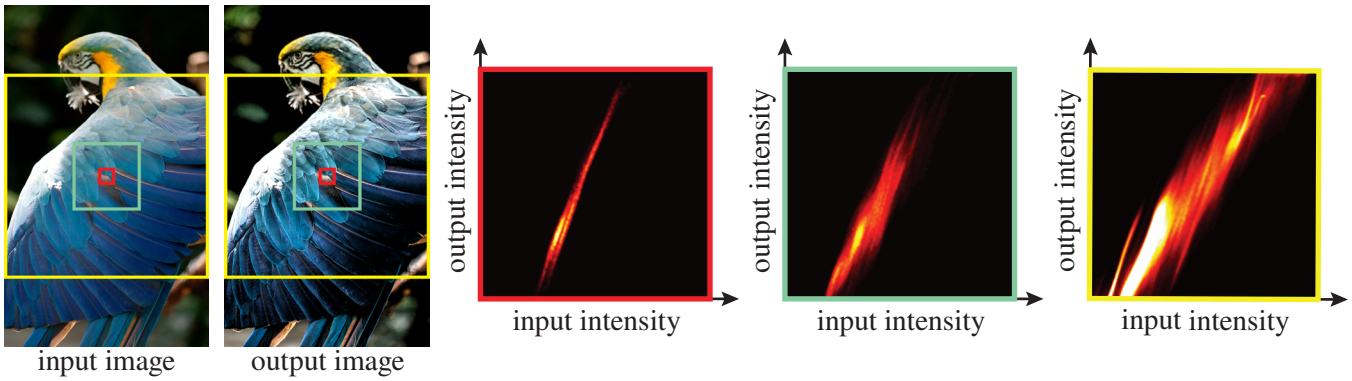

Figure 3: *On the left we show input and output for an aggressive local contrast enhancement using local Laplacian filters [Paris et al. 2011]. At small scale (red,* 128×128 *pixels), even complex algorithms are well approximated by a smooth curve (red, right), and each intensity range along such a curve is well approximated by an affine model—a line. As patch size grows, the mapping deviates from a single smooth curve. Far-away pixels with the same color in the input do not have the same color in the output.*

series and also record how the output intensity varies under small perturbations of input intensity. This is now an affine model.

For well-behaved operators, these first order terms should be small. If a small change in input intensity produces a large change in output intensity, then the operator must amplify any noise present.

The scale at which the input-output mapping is well approximated by an affine model depends on the properties of the underlying operator. We illustrate this in figure 3, where we show input-output intensity scatter plots for patches of different sizes. An affine model works well for small patches, especially ones that contain limited brightness variation, but breaks once the patch size grows beyond the natural scale of the operator.

If a patch contains a large range of intensities and the operator is non-linear, the first-order Taylor expansion no longer holds. For example, if a patch contains very bright pixels and very dark pixels separated by an edge, the relationship between input and output may be very different on either side of that edge. We can however still model the relationship with affine models if we fit a *separate* affine model to each intensity range present in a patch. Linearly interpolating within this array of affine models as a function of intensity produces a piecewise-quadratic spline describing a *local curve* mapping input to output within this patch.

If we grow the patches larger still, then for many interesting operators, equal-intensity pixels in the input may map to very different intensities in the output, and the relationship can no longer be modeled by any single function (see the rightmost scatter plot in figure 3). We therefore slice the image into tiles and fit a distinct curve within each tile. For well-behaved operators these curves must vary smoothly across space. If they did not, neighboring pixels with the same intensities in the input could have very different intensities in the output, which means that the operator has introduced a new edge that was not present in the input. This is only desirable behavior for a small class of operators, most notably super-resolution and deblurring.

We now have a 3D array of affine models, indexed by position and intensity. We require them to vary smoothly with position and intensity, so we can represent them at low resolution. In fact, fitting them at low resolution projects the original operator onto a space of operators that do not amplify noise or introduce new edges. The 3D array can be treated as a type of bilateral grid that stores affine models instead of colors. Given any input position and intensity, we can trilinearly interpolate into it to retrieve an appropriate affine model that will tell us the output intensity.

Using the input/output image data alone, this bilateral grid is sparse. We can only fit affine models in the cells where we have input data (see figure 4). If we wish to apply the model to new, previously unseen input intensities, then we need to extend our models to cover the entire grid. Our desired smoothness property and our affine representation makes this straightforward. Spatial smoothness can be enforced by minimizing finite differences in model coefficients in the x and y directions. Likewise, intensity smoothness can be enforced by minimizing finite differences in the intensity direction z. A fully populated bilateral grid tells us the action of the imaging operator on *any* image given only a single example input and output. However, is merely a first-order Taylor series of the operator about this particular input, so the quality drops off rapidly as we deviate from it. If we sample the grid at locations driven by a *higher-resolution* version of the same input then we stay close to the original manifold, and so produce an output very similar to running the original operator at a higher resolution.

Affine color model We have thus far limited our discussion to grayscale images. To handle color inputs and outputs, we could use a 5D bilateral grid, which stores a 3×4 affine matrix from input to output color at each (x, y, r, g, b) cell. However, this space is too large: even with a low-resolution grid, each cell would not have enough data to adequately fit an affine transformation. Instead, we found that a hybrid color model works well. We use a 3D grid, where the z coordinate corresponds to luminance, but within each cell, we store a 3×4 affine matrix (1×4 for operators that map color to gray). The hybrid model only respects *luminance* edges by encoding them as a Euclidean distance, but locally models the operator as an affine color transformation, as in [Bousseau et al. 2009]. We use this color model for all our results.

Connection to transform recipes While transform recipes [Gharbi et al. 2015] handles color differently, for luminance they too fit local curves that map input intensity to output intensity. Indeed, they go one step further and fit distinct curves per spatial frequency band. However, fitting the high-frequency terms in this representation requires access to the full-resolution input and output, which for us begs the question. We are attempting to produce a full-resolution output *without* running the original operator at that resolution. Despite this handicap, our technique functions as well as transform recipes in terms of PSNR (figure 8) and is substantially faster.

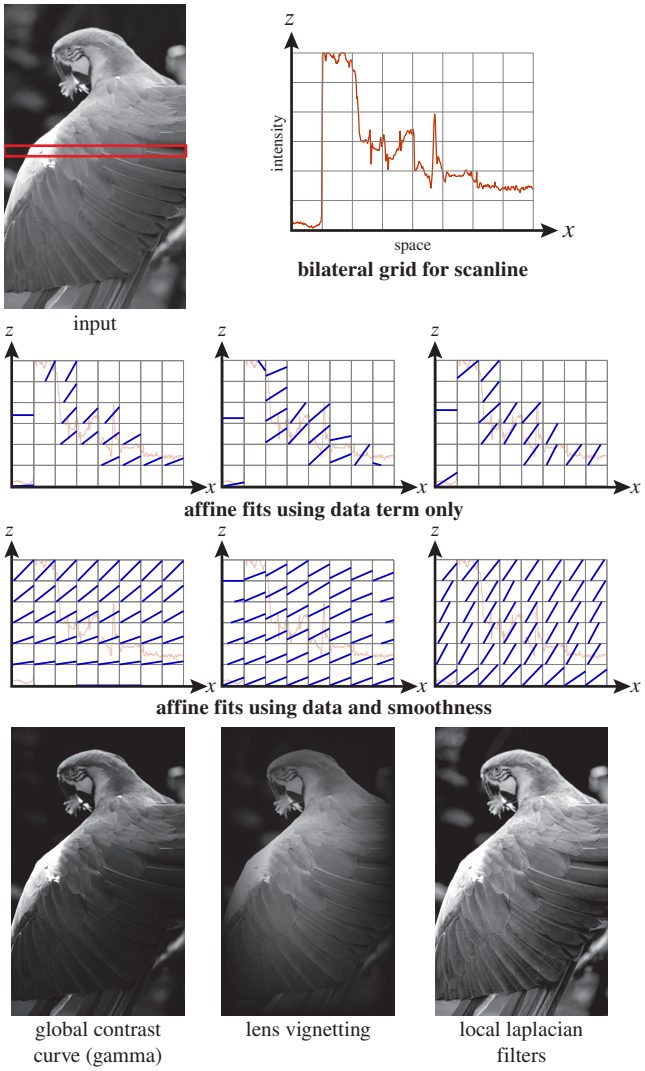

Figure 4: *Our algorithm modeling three different operators. Top: input image and bilateral grid for highlighted scanline. Second row: local affine models fit to the input/output pair with a data term only. Note that grid cells with no data are empty. Third row: affine models fit everywhere using data and smoothness. A global curve (bottom left) results in affine models that vary with intensity (z), but not spatially. A vignette can be expressed as an affine model (a scaling) that varies with position but not intensity. A more complex effect (bottom right) produces affine models that varies with both.*

4 Algorithm

We accelerate an imaging operator by applying it to a low-resolution version of the input, fitting a bilateral grid of affine models to the low-resolution input/output pair, and then applying the models to the high-resolution input. We present two methods for fitting affine models. The first, based on optimization, is slow but produces the highest-quality results. The second is a fast approximation to this that runs in real-time on a mobile device.

Once fit, applying the model to the high-resolution input involves a per-pixel trilinear interpolation to retrieve a 3×4 affine color transform, and then a matrix multiply to generate the output color. This can be done rapidly on a GPU, and an OpenGL shader that implements this is included in the supplemental material.

Preliminaries Chen et al. [2007] showed that joint bilateral filtering of an image with respect to a guide can be expressed as a splat-blur-slice procedure. We construct a bilateral grid by *splatting* the input values at locations determined by the guide image, *blurring* the values in the grid with a small Gaussian kernel, and then *slicing* the values out of the grid by sampling it at same locations determined by the guide image. Barron et al. [2015] showed that with matching reconstruction filters (e.g., trilinear), filtering may be expressed as $S^T B S \mathbf{x}$, where $\mathbf{x}$ is the vector of values to be filtered, S and S^T are the guide-image-dependent splat and slice matrices respectively, and B is a blur in grid space. Using this notation, we set up an optimization problem to solve for grid of affine models.

Optimization Our energy function is a linear least squares problem and is the sum of data and smoothness terms. The data term seeks the 3D array of 3×4 matrices such that when *sliced* using the input luminance, then *applied* to the input color, best produces the low-resolution output. Note that while slicing and application both depend on the input image, the overall reconstruction is linear in the unknowns. The smoothness term is independent for each affine matrix coefficient and is the squared magnitude of its partial derivatives in each of the spatial (x, y) and intensity (z) directions. To balance the data and smoothness terms, we weight each derivative with tuning parameters $(\lambda_x, \lambda_y, \lambda_z)$, which we fix at $(1, 1, 4 \times 10^{-6})$ in all our experiments. Our optimization problem is:

$$\underset{\boldsymbol{\gamma}}{\operatorname{argmin}} \int \left(\beta - AS^T\boldsymbol{\gamma}\right)^2 + (\lambda_x D_x \boldsymbol{\gamma})^2 + (\lambda_y D_y \boldsymbol{\gamma})^2 + (\lambda_z D_z \boldsymbol{\gamma})^2 \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z \tag{1}$$

$\boldsymbol{\gamma}$ collects all the unknowns into a $w \times h \times d \times 3 \times 4$ element vector, S^T is the slicing matrix incorporating trilinear interpolation, and A is the matrix that applies the per-pixel 3×4 matrix to each input pixel. β is the low-resolution output image we seek to match. Note that S^T depends on the low-resolution input luminance, and A is simply the low-resolution input image replicated and reshaped to left-multiply each color transform.

This optimization problem can be discretized using standard finite element analysis and the resulting system of linear equations solved using sparse QR factorization. For a 320×240 image and a $20 \times 15 \times 10$ bilateral grid, our MATLAB implementation takes about 15 seconds on a desktop workstation, which is too slow for real-time applications. Below, we derive a fast approximation suitable for deployment on a mobile device.

Fast approximation Instead of optimizing a single global energy function, the fast approximation solves for the affine matrices in the bilateral grid as a set of smaller overlapping linear least squares problems. Within each grid cell, we want the 3×4 matrix M that best maps each input pixel i with RGB values α_{1i}, α_{2i}, α_{3i} to its corresponding output RGB values β_{1i}, β_{2i}, β_{3i}. We handle the constant term in this affine relationship by setting $\alpha_{4i} = 1$. We therefore we want the least-squares solution to $M\alpha = \beta$, where M is the unknown. This is optimized by solving $M\alpha\alpha^T = \beta\alpha^T$ for M in each grid cell.

The algorithm iterates over each input pixel, identifies its corresponding bilateral grid cell using its position and intensity, and accumulates $\alpha\alpha^T$ and $\beta\alpha^T$ into that cell. Taking into account that the first term is a symmetric matrix (it is the Gram matrix), this accumulates 22 distinct values in each grid cell. Note that four of these values (the last row/column of $\alpha\alpha^T$) are precisely the values that are accumulated in each grid cell during a conventional bilateral filter using the bilateral grid: α_{1i}, α_{2i}, α_{3i}, and 1.

We then enforce smoothness of the solution by blurring these 22 terms across the x, y, and intensity axes of the grid with a 7-tap separable filter. This turns our independent least squares problems into larger overlapping weighted least squares problems with weights that diminish with distance from the cell in x, y, and intensity. For under-constrained cells, this effectively inpaints constraints from more populated cells in the grid. We found that the precise blur kernel does not significantly affect results, so long as it has a strong central lobe. We use a $1/(r+1)^3$ filter, motivated by the kernel used to interpolate smooth membranes by Farbman et al. [2011].

Some cells may still be under-constrained after this inpainting. We therefore instead solve a modified system where under-constrained cells degrade towards γ, a robust ratio between the average output luminance and the average input luminance over the pixels that contributed to that cell:

$$M(\alpha\alpha^T + \lambda I) = \beta\alpha^T + \lambda\gamma I \tag{2}$$

where $\lambda = 10^{-6}((\alpha\alpha^T)_{44} + 1)$. Recall that $\alpha_{4i} = 1$, so $(\alpha\alpha^T)_{44}$ is a count of the number of constraints influencing this cell after blurring. λ must scale with this count so that a huge number of linearly-dependent constraints (e.g., from a region of constant color) cannot reduce it to floating-point insignificance. See the code in the supplemental material for the precise formulation.

Our fast approximation is written in the Halide image processing language [Ragan-Kelley et al. 2013], and is based on the bilateral grid sample code from that project. It is parallelized, vectorized, and fused for locality in the same way. With default settings, our implementation can solve for affine models in 1–2 ms on a desktop CPU, and about 10 ms on a mobile CPU (see figure 5).

Connection to the fast guided filter The fast guided filter [He and Sun 2015] fits an array of affine matrices that map input RGB to output RGB within overlapping tiles. It is fast both because it fits at low resolution and also because it exploits the same trick we do—it blurs Gram matrices to create overlapping problems. If we restrict our fast approximation to a single intensity bin (thus reducing the dimensionality of the array from three to two), and correspondingly use bilinear interpolation instead of trilinear to slice out the affine matrices, our method becomes the fast guided filter. This technique can only express local lines rather than local curves, and therefore models a more limited set of operators (see figure 8).

Connection to joint bilateral upsampling Although Kopf et al. [2007] did not originally describe it in this way, joint bilateral upsampling can be implemented using a bilateral grid by *splatting* low-resolution output values at locations determined by low-resolution input intensities, *blurring* within the grid, then *slicing* at locations determined by the high-resolution image [Adams 2011]. If we restrict our fast method to solving for only the constant term in each grid cell instead of an affine matrix, it reduces to this implementation of joint bilateral upsampling. The linear system we solve in each grid cell reduces to 1×1, and is exactly the division by the sum of the weights necessary in joint bilateral upsampling.

Fitting a constant rather than an affine matrix tends to produce piecewise-constant outputs (again, see figure 8). Kopf et al. therefore do not use joint bilateral upsampling to produce output RGB values directly, but rather for quantities that are naturally piecewise constant, such as depth, alpha mattes, chrominance, or gain maps for local tone mapping. Utilizing joint bilateral upsampling effectively therefore requires some degree of problem-specific insight.

Note that if we degrade our fast method by taking the limit as $\lambda \to \infty$, then equation 2 becomes $M = \gamma I$. This shows that our regularizer is equivalent to Kopf's tone mapping method that joint bilateral upsamples a local gain map.

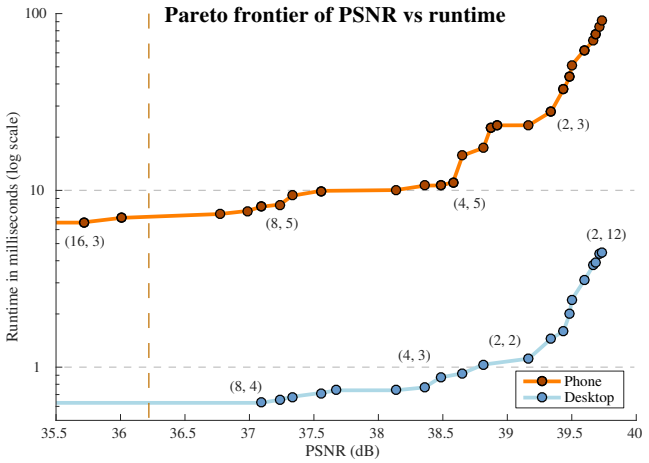

Figure 5: *We can trade-off PSNR against runtime by varying grid resolution. Here we show the Pareto frontier across various grid resolutions for the fast approximate method applied to the local Laplacian filters task from figure 7, with the vertical dashed line corresponding to the parameter settings used. We also label a few points with spatial bin size in pixels and number of intensity bins. The desktop has an Intel Xeon E5-2690 CPU running at 2.9 GHz, and the phone has a Qualcomm Snapdragon 810 running at 2 GHz.*

Our fast approximation thus unifies the fast guided filter and joint bilateral upsampling, reduces to either of them if suitably restricted, and is more expressive than both.

5 Results

We evaluate our global method and fast approximation on a variety of image processing operators, applied to 116 images from the transform recipes dataset. The operators we run are:

- **Local Laplacian filters** by Aubry et al. [2014] enhances local contrast in images by constructing the Laplacian pyramid of the desired output, coefficient-by-coefficient.
- **Style transfer** from the same paper by Aubry et al. [2014] transfers style from a model image to the input by altering the input's distribution of gradients to match that of the model.
- **Unsharp mask** sharpens an image by magnifying the difference between the input and a blurred version.
- **Colorization** by Levin et al. [2004] adds color to a grayscale image given a set of sparse scribbles.
- **Portrait style transfer** by Shih et al. [2014] transfers the style of one portrait image to another.
- $\boldsymbol{L_0}$ **smoothing** by Xu et al. [2011] removes image detail for the purposes of stylization.
- **Matting** computes an alpha matte that separates foreground from background. We use the method of Chen et al. [2013].
- **Dehazing** corrects for contrast loss due to atmospheric scattering. We use the algorithm of Kim et al. [2013].

Most images are 5–8 Mpixels (portraits are 1.3 Mpixels). We box-downsample each image by factors of 4 and 8, then apply the operator to produce a low-resolution output. We run bilateral guided upsampling on a range of grid resolutions and assess image quality by computing its PSNR and structural similarity index (SSIM) with respect to ground truth. With default parameters of 8x downsampling, 8 intensity bins, and spatial bins corresponding to 16×16 pixels in the low-resolution input, our global method achieves a

a) input **b)** dehaze then downsample **c)** bilateral guided upsampling modeling **a** → **b** **d)** downsample then dehaze **e)** bilateral guided upsampling modeling **a** → **d**

Figure 6: *Most imaging operators that manipulate tone and color approximately commute with downsampling: applying the same operator to a higher-resolution image of the same scene should not produce a significantly different output [Jeong et al. 2011]. However, not all black-box operators are commutative in this way (perhaps due to hidden internal parameters expressed in units of pixels). An example of this is the dehazing technique of Kim et al. [2013], which produces different results when run at high and low resolution (**b** and **d**, respectively). Bilateral guided upsampling can model the action of the operator at either resolution (**c** matches **b**, and **e** matches **d**). However the premise of this paper is that modeling an operator at low resolution and using that model to produce a full-resolution result is equivalent to running the operator at full resolution, and that is not the case here (**e** does not look like **b**). The image is from the transform recipes dataset [Gharbi et al. 2015].*

mean PSNR of 28.4 dB and a mean SSIM of 0.92 over the entire dataset. Our fast approximation produces slightly lower image quality (mean PSNR of 27.0 dB, mean SSIM of 0.88) but runs in under 1 ms on a desktop workstation. To measure the impact of commutativity, we box-downsample the full-resolution ground truth to represent the output of an ideal scale-invariant operator that commutes with downsampling (see figure 6). When applied to these image pairs, mean PSNR and SSIM increase to 33.3 dB and 0.94 for the global method and 29.3 dB and 0.88 for the fast approximation. Full-resolution versions of the images in figures 6 and 7 are included in the supplement along with PSNR and SSIM for the entire dataset.

For a 10 MPixel image with default parameters, our fast approximation on a desktop CPU takes 2 ms to downsample by 8x, 2 ms to fit curves, and 13 ms to apply them. Most of the cost comes from trilinearly interpolating the bilateral grid, which is slow on CPUs without texture hardware but almost free on GPUs. In practice, for most applications such as those in figure 7, performance is dominated by running the original operator at low resolution, even with the 64x acceleration we gain by computing it at 8x reduced resolution.

Discussion and Limitations Both the global optimization and its fast approximation perform well given the right parameters. While we used uniform parameter settings for figure 7, the optimal choice of downsampling factor and grid resolution is operator-dependent, reflecting the natural scale at which the operator has its effect. Most operators have semantically meaningful parameters that guide how they scale with resolution. Linear parameters such as filter radii should be scaled linearly (e.g., unsharp mask), logarithmic parameters such as the number of pyramid levels should be subtracted (e.g., local Laplacian, portrait transfer), and properly designed regularizers should have units such as inverse pixels or inverse pixels squared, which dictate how they scale. For label propagation, we conservatively use nearest-neighbor to downsample masks (colorization, matting, portrait transfer).

The quality of our method depends on whether the operator can be modeled as a local curve and whether running it at reduced resolution faithfully captures what it would do to a higher-resolution image of the same scene. This is not always the case (figure 6). For a small-support unsharp mask, which enhances the highest frequencies, our local curves model only holds true over small spatial bins, and downsampling may entirely discard the frequencies we need to learn a valid model. At the opposite extreme, a global contrast curve needs only one spatial bin and can fit the curve from a very low-resolution input/output pair. When the size of the grid bins or the downsampling factor is too large for the operator, we fail in the direction of applying a single global curve, which merely preserves input detail rather than enhancing it (e.g., the wall in style transfer), or suppressing it (e.g., L_0 smoothing and alpha matting). The guided filter and joint bilateral upsampling exhibit this same phenomenon more strongly (figure 8). Transform recipes' pyramid-based model has more representative power, but requires running the original operator at full resolution. If one could learn a transform recipe without running the original operator at full resolution, it may be possible to reap the benefits of both techniques.

6 Conclusion

We have described a simple method for accelerating a large class of image operators. For many operators, nearby pixels that have a similar color in the input also have a similar color in the output, meaning that the image-dependent function mapping input colors to output colors is smooth in bilateral space. We can learn this smooth function by running the operator at reduced resolution, and fitting a bilateral grid of affine matrices that map the inputs to the outputs. This collection of affine matrices can be thought of as tangent planes to the true function—for each input position and intensity, the constant term yields the output color and the linear term controls how a small variation in input position or color yields a small variation in the output color. Thus, they let us accurately predict the results of the operator given a higher-resolution input.

Our model can be restricted to the constant term alone in order to produce joint bilateral upsampling, or restricted to a single intensity bin to produce the fast guided filter. Thus, we relate these two methods and demonstrate they both can be interpreted as resampling by fitting and applying local models.

7 Acknowledgments

We thank Jon Barron, Rob Carroll, and Marc Levoy for discussions and feedback and Elena Adams for the bison and parrot photos.

References

ADAMS, A., BAEK, J., AND DAVIS, M. A. 2010. Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, vol. 29, 753–762.

ADAMS, A. 2011. *High-dimensional gaussian filtering for computational photography*. PhD thesis, Stanford University.

AUBRY, M., PARIS, S., HASINOFF, S. W., KAUTZ, J., AND DURAND, F. 2014. Fast local Laplacian filters: theory and applications. *ACM Trans. Graph. 33*, 5 (Sept.), 167:1–167:14.

BARRON, J. T., AND POOLE, B. 2016. The fast bilateral solver. *ECCV*.

BARRON, J. T., ADAMS, A., SHIH, Y., AND HERNANDEZ, C. 2015. Fast bilateral-space stereo for synthetic defocus. *CVPR*.

BOUSSEAU, A., PARIS, S., AND DURAND, F. 2009. User-assisted intrinsic images. *ACM Trans. Graph. 28*, 5 (Dec.), 130:1–130:10.

Figure 7: *Evaluation of bilateral guided upsampling and our fast approximation on a variety of image processing operators. Bilateral guided upsampling is able to model complex local operators such as local Laplacian filters, style transfer, and even a strong unsharp mark with large radius and strength. In these cases, the input-output relation is approximately a curve. Portrait style transfer works despite the fact that this is not true in the worst-case patch. Below the dashed line, we demonstrate more severe failures.* L_0 *smoothing flattens the image into patches of constant intensity, which appear as overlapping horizontal lines in the scatter plot. This is not a local curve and our algorithm leaves much of the input structure intact. Alpha matting is another failure case, where our model fails to capture the complex geometric dependence between color and alpha. Most images are downsampled by a factor of 8 and upsampled using a bilateral grid with 8 intensity bins and spatial bins corresponding to* 16×16 *pixels in the low-resolution input. The portrait is only 1.3 Mpixels and is instead downsampled by a factor of 4 and upsampled with* 8×8 *bins. We encourage the reader to zoom in or consult the supplemental material to view our full-resolution images. The images in rows 4-7 are from the transform recipes dataset [Gharbi et al. 2015]. Portrait photo © Colin Strain and used with permission.*

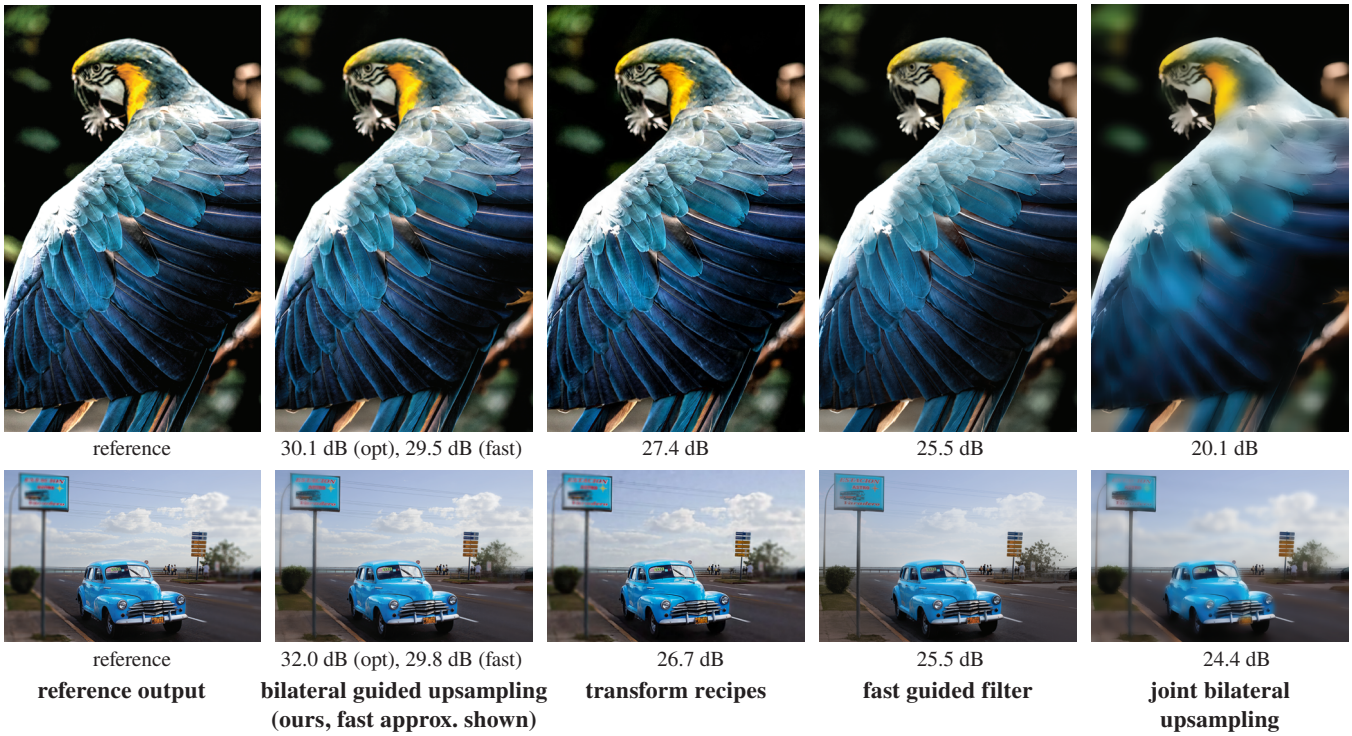

Figure 8: *Quality comparison of various algorithms on two challenging cases. Top: Aggressive detail enhancement on a highly textured subject using local Laplacian filters. Bottom: Contrast enhancement coupled with a strong blur applied to the left and right sides of the image, leaving the center intact. Bilateral guided upsampling (fast version shown) and transform recipes both manage to reproduce the operators, whereas the fast guided filter cannot capture the increase in contrast, and joint bilateral upsampling degrades toward a piecewise-constant output (it was not designed for this task). Transform recipes' low PSNR is partially due to the JPEG compression inherent in that technique, which is especially visible in the center. We ran all algorithms using equivalent parameters (8x image downsampling, 8 intensity bins, and spatial bins corresponding to* 4×4 *pixels in the low-resolution input). For transform recipes, we used the authors' implementation with default compression settings. Full-resolution images are in the supplement. The car image is from the transform recipes dataset [Gharbi et al. 2015].*

CHEN, J., PARIS, S., AND DURAND, F. 2007. Real-time edge-aware image processing with the bilateral grid. *ACM Trans. Graph. 26*, 3.

CHEN, Q., LI, D., AND TANG, C.-K. 2013. KNN matting. *IEEE TPAMI 35*, 9, 2175–2188.

FARBMAN, Z., FATTAL, R., LISCHINSKI, D., AND SZELISKI, R. 2008. Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Trans. Graph. 27*, 3, 67:1–67:10.

FARBMAN, Z., FATTAL, R., AND LISCHINSKI, D. 2011. Convolution pyramids. *ACM Trans. Graph. 30*, 6 (Dec.), 175:1–175:8.

GASTAL, E. S. L., AND OLIVEIRA, M. M. 2011. Domain transform for edge-aware image and video processing. *ACM Trans. Graph. 30*, 4 (July), 69:1–69:12.

GASTAL, E. S. L., AND OLIVEIRA, M. M. 2012. Adaptive manifolds for real-time high-dimensional filtering. *ACM Trans. Graph. 31*, 4 (July), 33:1–33:13.

GHARBI, M., SHIH, Y., CHAURASIA, G., RAGAN-KELLEY, J., PARIS, S., AND DURAND, F. 2015. Transform recipes for efficient cloud photo enhancement. *ACM Trans. Graph. 34*, 6.

HE, K., AND SUN, J. 2015. Fast guided filter. *CoRR abs/1505.00996*.

HE, K., SUN, J., AND TANG, X. 2013. Guided image filtering. *IEEE TPAMI 35*, 6, 1397–1409.

JEONG, W.-K., JOHNSON, M. K., YU, I., KAUTZ, J., PFISTER, H., AND PARIS, S. 2011. Display-aware image editing. *ICCP*.

KIM, J.-H., JANG, W.-D., SIM, J.-Y., AND KIM, C.-S. 2013. Optimized contrast enhancement for real-time image and video dehazing. *Vis. Comm. and Image Representation 24*, 3, 410–425.

KOPF, J., COHEN, M. F., LISCHINSKI, D., AND UYTTENDAELE, M. 2007. Joint bilateral upsampling. *ACM Trans. Graph. 26*, 3.

LEVIN, A., LISCHINSKI, D., AND WEISS, Y. 2004. Colorization using optimization. *ACM Trans. Graph. 23*, 3, 689–694.

PARIS, S., AND DURAND, F. 2006. A fast approximation of the bilateral filter using a signal processing approach. *ECCV*.

PARIS, S., HASINOFF, S. W., AND KAUTZ, J. 2011. Local Laplacian filters: edge-aware image processing with a Laplacian pyramid. *ACM Trans. Graph. 30*, 4 (July), 68:1–68:12.

RAGAN-KELLEY, J., BARNES, C., ADAMS, A., PARIS, S., DURAND, F., AND AMARASINGHE, S. 2013. Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines. *ACM SIGPLAN 48*, 6.

SHIH, Y., PARIS, S., BARNES, C., FREEMAN, W. T., AND DURAND, F. 2014. Style transfer for headshot portraits. *ACM Trans. Graph. 33*, 4 (July), 148:1–148:14.

TOMASI, C., AND MANDUCHI, R. 1998. Bilateral filtering for gray and color images. *IEEE ICCV*.

XU, L., LU, C., XU, Y., AND JIA, J. 2011. Image smoothing via L_0 gradient minimization. *ACM Trans. Graph. 30*, 6, 174:1–174:12.

YUAN, L., AND SUN, J. 2011. High quality image reconstruction from RAW and JPEG image pair. *IEEE ICCV*.