Fourier Analysis of the 2D Screened Poisson Equation for Gradient Domain Problems

Pravin Bhat[1] Brian Curless[1] Michael Cohen[1,2] C. Lawrence Zitnick[2]

[1]University of Washington [2]Microsoft Research

Abstract. We analyze the problem of reconstructing a 2D function that approximates a set of desired gradients and a data term. The combined data and gradient terms enable operations like modifying the gradients of an image while staying close to the original image. Starting with a variational formulation, we arrive at the "screened Poisson equation" known in physics. Analysis of this equation in the Fourier domain leads to a direct, exact, and efficient solution to the problem. Further analysis reveals the structure of the spatial filters that solve the 2D screened Poisson equation and shows gradient scaling to be a well-defined sharpen filter that generalizes Laplacian sharpening, which itself can be mapped to gradient domain filtering. Results using a DCT-based screened Poisson solver are demonstrated on several applications including image blending for panoramas, image sharpening, and de-blocking of compressed images.

1 Introduction

Accurately and efficiently recovering a depth map or an image from gradients has become a common problem in computer vision and computer graphics. In photometric stereo, for instance, one measures gradients (normals) to the surface and then "integrates" them to recover a depth map. In gradient domain compositing applications, one combines the gradients of multiple images and then solves for the underlying image most compatible with those gradients. In both cases, the problem is over-constrained; in general, no function exists whose gradients match the input gradients. The goal then is to project to the nearest function whose gradients approximate the inputs. A common approach is to employ a least squares metric and integrate over the domain, leading to a Poisson equation. This equation may be solved using, e.g., multigrid methods [1], fast marching methods [2], or so-called fast Poisson solvers [3].

In many applications the use of fast Poisson solvers based on the Fast Fourier Transform (FFT) are overlooked. This is due in part to the fact that fast Poisson solvers are restricted in the class of problems they can handle; e.g., spatially varying weights are not supported. Further, more complex approaches such as multigrid methods are perceived to be much faster.

In this paper, we expand the set of gradient domain problems that may be solved directly and exactly in the Fourier domain by including a data term that the reconstructed function must also approximate. This extra term enables operations like modifying the gradients of an image while staying close to the

original image or combining independently measured depth maps and normals. We pose the problem in a variational framework and arrive at a modification to the Poisson equation, a result that corresponds to a 2D version of the "screened Poisson equation" known in physics [4].

Additional analysis reveals the structure of the spatial filters used to solve the 2D screened Poisson equation. Further, we show that uniformly scaling the gradients in an image, which intuitively ought to sharpen it, does in fact precisely correspond to a linear sharpen filter. This filter generalizes the standard Laplacian subtraction filter, which we show can also be interpreted in the same variational framework. In that framework, we show that, unlike Laplacian subtraction, gradient scaling includes a penalty term for large gradients.

We demonstrate results using an FFT-based screened Poisson solver for a set of image operations including image blending for panoramas, image sharpening, and de-blocking of compressed images. The FFT approach is direct and exact, unlike efficient least squares solvers which are typically iterated to within some error tolerance [5, 6]. Though not quite as fast as the fastest solvers, the performance does scale well, and the implementation is very simple given commonly available libraries.

The paper is organized as follows: Section 2 describes previous work. Next in Section 3 we formulate the problem using a variational framework. In Section 4 we map our variational approach to the Fourier domain followed by a mapping to the spatial domain in Section 5; in these sections, we also analyze gradient-based sharpen filters. A description of our FFT-based screened Poisson solver is presented in Section 6 with results in Section 7. We conclude our paper with a discussion in Section 8.

2 Related work

Gradient domain problems that map to Poisson equations can arise in numerous scenarios in vision and graphics. Simchony et al. [7] describe a number of such scenarios in vision including shape-from-shading, the lightness problem, and optical flow. The problem of integrability of normals, in addition to shape-from-shading, is important in photometric stereo [8] and Helmholtz stereo [9]. In computer graphics, gradient domain methods have become an essential tool in the image processing toolbox. Examples include tone-mapping of high dynamic range images [10], Poisson image editing [11], and digital photomontage [12].

The introduction of a data function term is less common, though it is a natural extension and is getting increasing attention. An early example is given by Horn for shape-from-shading, in the form of a term that minimizes the difference between the reflectance map and image irradiance, along with the surface gradient [13]. Several papers combine depth information with normals to improve depth map reconstruction [14–16]. Lischinski et al. [17] apply strokes as data constraints for gradient-based scattered data interpolation. More recently, Bhat et al. [18] have introduced a variety of image operations based on gradients and data images to enable new image and video processing filters like saliency sharpening, suppressing block artifacts in compressed images, and non-photorealistic

rendering. All of these methods take a linear (weighted or unweighted) least squares approach to representing and solving their problems. Agrawal et al. [19] explore a wide space of formulations (some of them non-linear) for surface reconstruction that includes the possibility of a data term. Agrawal's thesis [20] is an excellent survey of surface reconstruction techniques from gradients. In this paper, we analyze the gradients plus data term using a formulation that corresponds to unweighted least squares, but analyze it in the Fourier domain and develop a corresponding fast solver.

Linear least squares techniques are among the most common approaches to solving problems described above. Szeliski [21] provides a nice summary of these approaches. A particular advantage of these approaches is their flexibility in modeling irregularly shaped domains and spatially varying weights. Agarwala [5] develops a quad-tree based solution for gradient domain compositing with uniform weights. A number of problems have a regular domain and uniform weights. Additional solvers include the fast marching method for integrating surface normals, introduced by Ho et al. [2] and the streaming multigrid method of Kazhdan and Hoppe [6]. Frankot and Chellapa [22] approached the same problem using a Fourier basis, and others have adopted cosine basis for their particular boundary conditions [22][23]. These latter approaches fall into the category of fast Poisson solvers [3]. We follow this Fourier approach to the problem of gradient domain integration that includes a data term.

We also note that Weiss [24] provides a method for computing a discrete spatial filter that minimizes squared differences between filtered versions of an image and corresponding inputs. Our analysis focuses on the continuous problem, specifically variational and Fourier analysis of the screened Poisson equation.

3 Variational formulation

In this section, we describe the standard gradient integration problem and its Poisson solution and then expand this result to include a data function term.

The problem of computing a function $f(x, y)$ whose gradient $\nabla f(x, y)$ is as close as possible to a given gradient field $\mathbf{g}(x, y)$ is commonly solved by minimizing the following objective:

$$\iint \|\nabla f - \mathbf{g}\|^2 \, dx \, dy. \tag{1}$$

Note that $\mathbf{g}$ is a vector-valued function that is generally not a gradient derived from another function. (If $\mathbf{g}$ were derived from another function, then the optimal f would be that other function, up to an unknown constant offset.)

It is well-known that, by applying the Euler-Lagrange equation, the optimal f satisfies the following Poisson equation:

$$\nabla^2 f = \nabla \cdot \mathbf{g}, \tag{2}$$

which can be expanded as $f_{xx} + f_{yy} = g^x_x + g^y_y$, where $\mathbf{g} = (g^x, g^y)$. Subscripts in x and y correspond to partial derivatives with respect to those variables. We

have superscripted g^x and g^y to denote the elements of $\mathbf{g}$ rather than subscript them, which would incorrectly suggest they are partial derivatives of the same function.

We now expand the objective beyond the standard formulation. In particular, we additionally require $f(x, y)$ to be as close as possible to some data function $u(x, y)$. The objective function to minimize now becomes:

$$\iint \lambda_d (f-u)^2 + \|\nabla f - \mathbf{g}\|^2 \, dx \, dy, \tag{3}$$

where λ_d is a constant that controls the trade-off between the fidelity of f to the data function versus the input gradient field.

To solve for the function f that minimizes this integral, we first isolate the integrand:

$$L = \lambda_d (f-u)^2 + \|\nabla f - \mathbf{g}\|^2 = \lambda_d (f-u)^2 + (f_x - g^x)^2 + (f_y - g^y)^2. \tag{4}$$

The function f that minimizes this integral satisfies the Euler-Lagrange equation:

$$\frac{\partial L}{\partial f} - \frac{\partial}{\partial x}\frac{\partial L}{\partial f_x} - \frac{\partial}{\partial y}\frac{\partial L}{\partial f_y} = 0. \tag{5}$$

Substituting and differentiating, we then have:

$$2\lambda_d (f-u) - 2(f_{xx} - g^x_x) - 2(f_{yy} - g^y_y) = 0. \tag{6}$$

Rearranging gives us:

$$\lambda_d f - (f_{xx} + f_{yy}) = \lambda_d u - (g^x_x + g^y_y) \tag{7}$$

or equivalently:

$$\lambda_d f - \nabla^2 f = \lambda_d u - \nabla \cdot \mathbf{g}. \tag{8}$$

The left-hand side of this equation is a screened Poisson equation, typically studied in three dimensions in physics [4]. Our analysis will be in 2D. As expected, setting $\lambda_d = 0$ nullifies the data term and gives us the Poisson equation.

4 Fourier solution

In this section we analyze the 2D screened Poisson equation the Fourier domain. As with fast Poisson solvers, we can solve the screened Poisson equation (Equation 8) by taking its Fourier transform. First, we adopt the (s_x, s_y) spatial frequency notation of Bracewell [25] and recall that for a given function h and its Fourier transform, $\mathcal{F}\{h\} = H$, we have $\mathcal{F}\{h_x\} = i2\pi s_x H$, $\mathcal{F}\{h_y\} = i2\pi s_y H$, $\mathcal{F}\{h_{xx}\} = -4\pi^2 s_x^2 H$, and $\mathcal{F}\{h_{yy}\} = -4\pi^2 s_y^2 H$.

Simply transforming the left and right sides of Equation 7 gives us:

$$\lambda_d F + 4\pi^2 s_x^2 F + 4\pi^2 s_y^2 F = \lambda_d U - i2\pi s_x G^x - i2\pi s_y G^y, \tag{9}$$

where F, U, G^x, and G^y are the Fourier transforms of f, u, g_x, and g_y respectively. Solving for F, we find:

$$F = \frac{\lambda_d U - i2\pi s_x G^x - i2\pi s_y G^y}{\lambda_d + 4\pi^2 \left(s_x^2 + s_y^2\right)}. \tag{10}$$

We can interpret this equation as combining, in the numerator, the data function with "sharpened" versions (after taking derivatives) of the gradient field components, and then low-pass filtering the result with the denominator, which tends to dampen high frequencies.

Note that when $\lambda_d = 0$, Equation 10 simplifies to the well-known fast Poisson solver result:

$$F = \frac{-i2\pi s_x G^x - i2\pi s_y G^y}{4\pi^2 \left(s_x^2 + s_y^2\right)}. \tag{11}$$

This solution, however, is undefined at $s_x = s_y = 0$, corresponding to an unknown DC term (constant offset) which must be supplied. Thus, there exists a null space of solutions, and the operation is not strictly invertible. This observation is evident by examining the objective in Equation 1, which is invariant to constant offsets to f. This situation does not arise, however, when a data term is present $\lambda_d > 0$, in which case we find $F(0,0) = U(0,0)$.

4.1 Image sharpening through gradient amplification

The previous section showed how gradient domain functionals that can be written in the form given by Equation 3 can be solved in the Fourier domain. It turns out that many interesting image processing filters can be written quite intuitively in this form, as will be discussed in Section 7.

In this section, we explore one intuitive example: sharpening an image by scaling up its gradients. Consider taking an image u and sharpening it by boosting its gradients ∇u by a constant factor c_s. Clearly, if one simply required an output image with scaled gradients, a simple (and optimal) solution would be to scale the image intensities by c_s. But, in addition to possibly pushing the intensities out of displayable range, this output image has drifted substantially from the intensities of the input image.

Instead, we can formulate an objective that trades off fidelity to the image against fidelity to the amplified gradients:

$$\iint \lambda_d (f-u)^2 + \|\nabla f - c_s \nabla u\|^2 \, dx \, dy. \tag{12}$$

This objective function is equivalent to the one in Equation 3, where we now have $\mathbf{g} = c_s \nabla u$. The Euler-Lagrange solution is then:

$$\lambda_d f - \nabla^2 f = \lambda_d u - c_s \nabla^2 u. \tag{13}$$

Similarly, the Fourier domain versions of gradient functions become: $G^x = i2\pi s_x U$ and $G^y = i2\pi s_y U$. Substituting into Equation 10, we obtain:

$$F = \left[\frac{\lambda_d + 4\pi^2 c_s \left(s_x^2 + s_y^2\right)}{\lambda_d + 4\pi^2 \left(s_x^2 + s_y^2\right)}\right] U = \left[\frac{1 + 4\pi^2 (c_s/\lambda_d) \left(s_x^2 + s_y^2\right)}{1 + 4\pi^2 (1/\lambda_d) \left(s_x^2 + s_y^2\right)}\right] U. \tag{14}$$

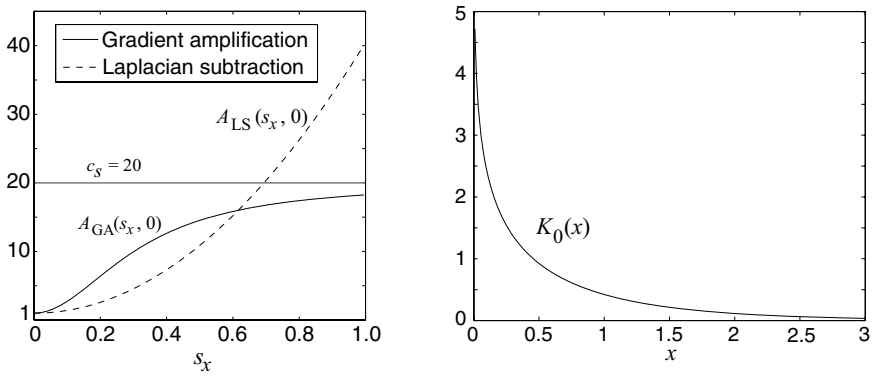


Fig. 1. Left: The frequency domain filters for gradient amplification (A_{GA}) and Laplacian subtraction (A_{LS}). Here, we set the parameters as follows: $\lambda_s = 1$, $\lambda_d = 4$, and $c_s = 20$. **Right:** The zeroth order modified Bessel function of the second kind, $K_0(x)$. In 2D, $K_0(r)$ is rotationally symmetric, and, while infinite at $r = 0$, is integrable.

For $c_s > 1$, this equation results in boosting the high frequencies of u, and thus f is a sharpened version of u. As $(s_x^2 + s_y^2) \to \infty$, we can see that the frequency amplification levels off to a constant, i.e., to c_s, as illustrated in Figure 1, in which A_{GA} corresponds to the frequency filter in brackets in Equation 14.

5 Spatial solution

Given our Fourier analysis we can map our problem back into the spatial domain. We can analyze the denominator in Equation 10 and treat it as a filter being applied to the numerator. Consider the denominator:

$$\frac{1}{\lambda_d + 4\pi^2 \left(s_x^2 + s_y^2\right)}. \tag{15}$$

Since this filter is radially symmetric, we can compute its Fourier transform using the Hankel transform, with radial frequency $\rho = \sqrt{s_x^2 + s_y^2}$ and spatial radius $r = \sqrt{x^2 + y^2}$. The following transform is known:

$$\mathcal{F}_\rho \left\{ \frac{1}{a^2 + \rho^2} \right\} = 2\pi K_0(2\pi a r), \tag{16}$$

where $\mathcal{F}_\rho\{\}$ is the Hankel transform from the frequency to the spatial domain, and K_0 is the zeroth order modified Bessel function of the second kind. This function has the approximate shape of a rotationally symmetric exponential function over the radius e^{-r}/r and can be seen in Figure 1. (In fact, in three dimensions, the exponential form e^{-r}/r is known to be the exact solution for the screened Poisson equation [4].)

With some simple algebraic manipulations, we arrive at:

$$\mathcal{F}_\rho \left\{ \frac{1}{\lambda_d + 4\pi^2 \left(s_x^2 + s_y^2\right)} \right\} = \frac{1}{2\pi} K_0(2\pi\sqrt{\lambda_d} r) = \frac{1}{2\pi} K_0(2\pi\sqrt{\lambda_d (x^2 + y^2)}). \tag{17}$$

The numerator in Equation 10 corresponds to $\lambda_d u - g_x^x - g_y^y$ in the spatial domain, and so the final result is the following convolution:

$$f = \frac{1}{2\pi} K_0(2\pi\sqrt{\lambda_d (x^2 + y^2)}) * (\lambda_d u - g_x^x - g_y^y). \tag{18}$$

Thus, we see that f can be obtained by subtracting the divergence of the gradient field from the input image u, and then blurring the result with the K_0 filter. Note that as λ_d increases, the support of this filter becomes smaller; i.e., a stronger data term shrinks the support of the blurring filter.

5.1 Spatial domain sharpening

Here we determine the spatial domain filter associated with gradient amplification. Starting with Equation 14, we can decompose the frequency filter as follows:

$$\frac{\lambda_d + 4\pi^2 c_s \left(s_x^2 + s_y^2\right)}{\lambda_d + 4\pi^2 \left(s_x^2 + s_y^2\right)} = c_s - \frac{\lambda_d (c_s - 1)}{\lambda_d + 4\pi^2 \left(s_x^2 + s_y^2\right)}. \tag{19}$$

The inverse Fourier transform of the first term is a scaled Dirac delta function, and the second term follows from Equation 17, giving us:

$$\mathcal{F}_\rho \left\{ \frac{\lambda_d + 4\pi^2 c_s \left(s_x^2 + s_y^2\right)}{\lambda_d + 4\pi^2 \left(s_x^2 + s_y^2\right)} \right\} = c_s \delta(x, y) - \frac{\lambda_d (c_s - 1)}{2\pi} K_0(2\pi\sqrt{\lambda_d (x^2 + y^2)}). \tag{20}$$

Convolving this with the data function then gives:

$$f(x, y) = c_s u(x, y) - \frac{\lambda_d (c_s - 1)}{2\pi} K_0(2\pi\sqrt{\lambda_d (x^2 + y^2)}) * u(x, y). \tag{21}$$

Note that when amplifying gradients, $c_s > 1$, so the operation amounts to blurring the image with the $K_0()$ filter and subtracting it from the original image.

5.2 Relationship to Laplacian subtraction sharpening

In this section, we relate sharpening by gradient amplification to the more conventional sharpening by Laplacian subtraction. First, we return to the Fourier domain, define another constant $\lambda_s = c_s/\lambda_d$, and re-write Equation 14 as:

$$F = \left[\frac{1 + 4\pi^2 \lambda_s \left(s_x^2 + s_y^2\right)}{1 + 4\pi^2 (1/\lambda_d) \left(s_x^2 + s_y^2\right)} \right] U = A_{\mathrm{GA}} U. \tag{22}$$

If we then let $\lambda_d \to \infty$ while holding λ_s constant, we arrive at:

$$F = \left[1 + 4\pi^2 \lambda_s \left(s_x^2 + s_y^2\right)\right] U = A_{\mathrm{LS}} U. \tag{23}$$

This equation is precisely the Fourier transform of a commonly known sharpen filter (the Laplacian subtraction filter):

$$f = u - \lambda_s \nabla^2 u. \tag{24}$$

Thus, the gradient domain sharpen filter subsumes another common sharpen filter. At the same time, unlike this common sharpen filter, our new filter has an additional parameter to control the amount of high frequency gain. As mentioned above, the parameter c_s controls the maximum frequency amplification.

It is interesting to note, as justified in Appendix A, that the Laplacian subtraction filter can also be interpreted in a variational framework. In particular, it minimizes the following:

$$\iint (f-u)^2 - 2\lambda_s \nabla f \cdot \nabla u \, dx \, dy. \tag{25}$$

In words, the desired function f must trade off being close to the input image u against maximizing the dot product between ∇f and ∇u. The second part of this objective favors f with large gradients aligned with the input image gradients.

Furthermore, one can take the objective for gradient sharpening (Equation 12) and show that minimizing it is equivalent to minimizing the following objective:

$$\iint (f-u)^2 - 2\lambda_s \nabla f \cdot \nabla u + \frac{1}{\lambda_d} \|\nabla f\|^2 \, dx \, dy. \tag{26}$$

This objective has almost the same form as the Laplacian subtraction objective, augmented with a penalty on gradient magnitudes. This observation gives some intuition for the difference between the two sharpen filters in the frequency domain; while the Laplacian subtraction filter amplifies high frequencies without bound, the gradient amplification filter rolls off to a constant due to the additional penalty on large gradients (Figure 1).

6 Discrete solution

The Fourier method provides a direct solution to the screened Poisson equation (Equation 8). In practice, we operate on sampled representations and solve the problem with discrete derivatives and the discrete Fourier transform (DFT).

In particular, we can first re-write Equation 7 as:

$$\lambda_d \hat{f} - (\hat{d}_x * \hat{d}_x * \hat{f} + \hat{d}_y * \hat{d}_y * \hat{f}) = \lambda_d \hat{u} - (\hat{d}_x * \hat{g}^x + \hat{d}_y * \hat{g}^y), \tag{27}$$

where $\hat{d}_x$ and $\hat{d}_y$ are discrete derivative filters in the x and y directions respectively and $\hat{f}$, $\hat{u}$, $\hat{g}^x$, and $\hat{g}^y$ are sampled versions of their continuous counterparts from Section 4. Typical choices for these discrete derivatives are forward, backward, or central differences. Taking the DFT of this equation gives us:

$$\lambda_d \hat{F} - \hat{D}_x^2 \hat{F} - \hat{D}_y^2 \hat{F} = \lambda_d \hat{U} - \hat{D}_x \hat{G}^x - \hat{D}_y \hat{G}^y, \tag{28}$$

where capitalization indicates having taken the DFT of a given discrete function. Rearranging, we arrive at:

$$\hat{F} = \frac{\lambda_d \hat{U} - \hat{D}_x \hat{G}^x - \hat{D}_y \hat{G}^y}{\lambda_d - \hat{D}_x^2 - \hat{D}_y^2} = \frac{\hat{H}}{\lambda_d - \hat{D}_x^2 - \hat{D}_y^2}, \tag{29}$$

where, for efficiency, $\hat{H}$ is computed as the DFT of $\hat{h} = \lambda_d \hat{u} - \hat{d}_x * \hat{g}^x - \hat{d}_y * \hat{g}^y$.

In practice, using the DFT is problematic, as it implicitly assumes the input sequence is periodic, placing the left boundary next to the right, and top next to bottom, before filtering. Instead, we employ the discrete cosine transform (DCT), which implicitly performs reflections across boundaries before tiling the plane periodically. We note that switching to a cosine basis has been explored by other researchers [22][23] and corresponds to Neumann boundary conditions.

A few implementation details follow. We compute the input gradients in $\hat{g}^x$ and $\hat{g}^y$ using backward differences, while the derivative filters $\hat{d}_x$ and $\hat{d}_y$ are implemented as forward differences when computing $\hat{h}$. Using backward then forward differences for derivatives also means that the derivative terms in the denominator of Equation 29 correspond to the DCT of the standard discrete 2D Laplacian. Because this filter has extremely local support in the spatial domain, we can compute its transform efficiently using the brute force DCT and without explicitly storing it.

We employ type-I DCTs using the FFTW library [26]. A single forward transform is needed to transform $\hat{h}$ to $\hat{H}$. A single reverse transform is needed to compute $\hat{f}$ from $\hat{F}$. For color images, this process is repeated for each color channel independently. The FFTW library can compute transforms in place in memory with single precision floating point arithmetic; therefore the maximum amount of memory required by our solver at any given time is four bytes for each pixel in the image. The FFTW library computes each transform in $O(n \log n)$ time, where n is the number of pixels in the image.

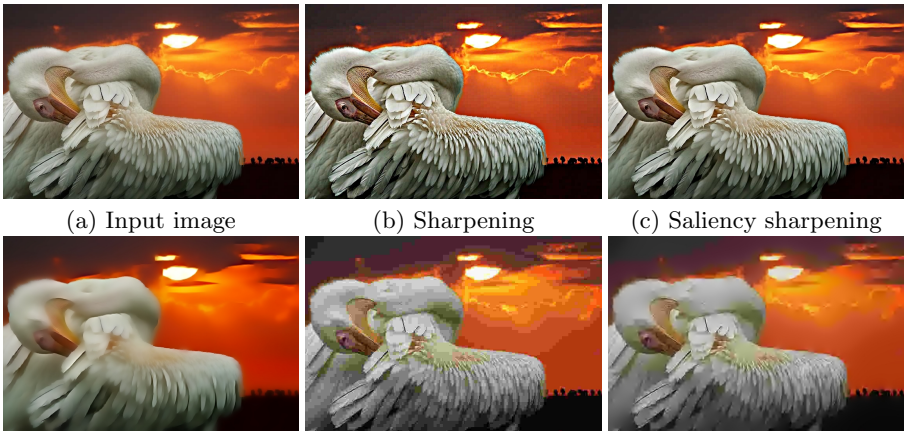
7 Applications and results

In this section, we demonstrate the practical utility of our Fourier domain analysis of gradient domain problems. We start with several examples in image processing and then compare performance to several state-of-the-art methods currently used to solve one of these applications.

7.1 Gradient domain filters and applications

The *de-blocking* filter proposed by Bhat et al. [18] suppresses blocking artifacts seen in highly compressed images by selectively attenuating gradients that lie across macro-block boundaries. The modified gradients are integrated to obtain a de-blocked image, while a data term ensures the result does not drift too far from the compressed image. See Figures 2e,2f for an example.

The *saliency sharpening filter* [18] builds on the gradient amplification sharpening filter presented in Section 4.1 in order to sharpen large scale features in an



(a) Input image (b) Sharpening (c) Saliency sharpening

(d) Image abstraction (e) Compressed image (f) De-blocking filter

Fig. 2. Image processing with data and gradient terms using the FFT solver. Sub-figures (b,c,d) show the application of various image enhancement filters on the input image (a). Note that the blockiness in (b) arises from sharpening JPEG compression artifacts present in the original image. Sub-figure (f) shows the de-blocking filter applied to a compressed image (e).

image without amplifying the noise or background clutter. This filter works by using an edge detector to detect large scale edges in the image and then selectively amplifying only those gradients that lie across large edges. Figures 2b,2c compare uniform gradient scaling to saliency sharpening.

The *image abstraction* filter defined by Bhat et al. [18] operates by suppressing small scale textures in an image. Similar to the saliency sharpening filtering, this filter uses an edge detector to determine which gradients give rise to salient features in an image. Then it selectively attenuates the non-salient gradients, thus abstracting away fine texture and noise from the input image (Figure 2d).

In *gradient domain compositing* applications [12], one combines the gradients of multiple images and then solves for the underlying image most compatible with those gradients. When no data function is used these applications can also be solved using regular Poisson solvers. Our method supports both types of gradient domain compositing applications – those with or without a data function.

7.2 Comparison to gradient-only methods

We now compare the performance of our method on a gradient domain compositing application (no data term) to several state-of-the-art methods. The application here concerns seamless image stitching to create large panoramic images. Agarwala [5] presented a method specialized for this application by relying on the solution to the Poisson equation being smooth in most regions of the domain. He compared the performance of his method (QT) to that of a preconditioned conjugate gradient solver using two different preconditioners

	Size	Time (s)					Memory (MB)				
Dataset	(MP)	FS	QT	SM	HB	LHAB	FS	QT	SM	HB	LHAB
St. Emilion	10	31	9	NA	3639	160	40	24	NA	362	1044
Beynac	12	22	8	17	3357	177	48	16	190	435	1252
Rainier	23	79	14	33	6446	268	92	27	110	620	1790
Sedona	36	85	29	NA	NA	NA	144	52	NA	NA	NA
Edinburgh	51	187	122	79	NA	NA	204	123	203	NA	NA
Crag	68	172	78	NA	NA	NA	272	96	NA	NA	NA
RedRock	88	270	118	118	NA	NA	352	112	133	NA	NA

Table 1. A comparison of memory and run-time performance of our method (FS) to other state-of-the-art methods (QT [5], out-of-core SM [6], HB [21], LHAB) [27]. We obtained the performance data for the non-Fourier methods from the QT and SM papers; the in-core speed of SM is better than shown here, but at very significant memory cost. Our method was evaluated on a different but a comparable speed processor (Intel Pentium 4 2.66GHz). Note also that the image dimensions in these datasets are not tuned for FFT, which explains some of the variation in timing with size.

- hierarchical basis (HB) preconditioning [21] and locally adaptive hierarchical basis (LAHB) preconditioning [27]. More recently (contemporaneous with our own work), Kazhdan and Hoppe [6] developed a streaming multigrid (SM) Poisson solver. We ran our Fourier solver (FS) on the evaluation dataset presented in the QT paper [5]. Table 1 compares the performance of all these methods.

As can be seen from the evaluation presented in Table 1, our method is somewhat slower than QT and SM, and substantially better than HB and LAHB. It should be noted that our method and HB have very low code complexity; each taking about 100 lines of C++ code to implement using freely available libraries (FFTW [26] in our case). In contrast, QT, SM, and LAHB are relatively complex to implement. Also, QT is tuned to the image stitching problem and SM is tuned to gradient domain integration, while our method solves a more general class of problems. Of course, the HB and LAHB preconditioners can be used with linear solvers to handle a still more general class of problems.

The FS timings include a small amount of overhead introduced by FFTW to determine or "plan" an efficient DCT algorithm given the image dimensions. We chose the lowest overhead planner. However, if the image dimensions are common or many transformations are to be applied (e.g., for interactive processing or for video), then a slower planner can be employed to give more efficient transforms, and the plan can be stored for later use. In addition, FFTW supports multiple processors and cores, and in our experiments with up to 8 processors and a good plan, we have observed roughly linear speed-ups with the number of processors. The choice of planner is simply a flag passed to the FFTW library, and executing on multiple processors/cores requires adding a single line of code.

8 Discussion

In this paper, we have shown how gradient domain variational problems that include a data term can be mapped to a 2D screened Poisson equation. This problem can be studied in both the Fourier and spatial domain. This analysis

gives insights into how gradient amplification corresponds to a linear sharpen filter and can be related to a standard Laplacian subtraction filter. Moreover, we show that this screened Poisson equation can be solved directly and efficiently using DCT's. By handling a data term, we are able to demonstrate a number of useful applications in image processing. We also note that the DCT formulation is extremely simple to implement with standard, optimized FFT libraries with multi-processor support.

The primary limitation of this approach is the inability to handle spatially varying weights on the gradient and data term constraints. Analysis shows that inserting such weighting into the variational formulation results in product terms that become convolutions in the frequency domain. In addition, we require complete, regular domains. Still, a number of applications can operate with constant weights over regular domains, or may potentially be initialized with an unweighted solution over a regular domain using our DCT solver to speed convergence of a more general solver.

Acknowledgments

We thank Sameer Agarwal and Eli Shechtman for helpful discussions that generalized our initial results on gradient domain filtering. Sameer also identified the connection to the screened Poisson equation. This work was supported by Microsoft, Adobe, the University of Washington Animation Research Labs, and an NVIDIA fellowship.

A Variational formulation of Laplacian subtraction

In Section 4.1, we showed that gradient amplification has a variational formulation that maps to a sharpen filter. Here we show that a more common sharpen filter – Laplacian subtraction (Equation 24) – can arise from another variational formulation. Consider the following objective to minimize:

$$\iint (f-u)^2 - 2\lambda_s \nabla f \cdot \nabla u \, dx \, dy. \tag{30}$$

We can isolate and expand the integrand:

$$L_{\mathrm{LS}} = (f-u)^2 - 2\lambda_s f_x u_x - 2\lambda_s f_y u_y. \tag{31}$$

Applying the Euler-Lagrange equation, the minimizing function must satisfy:

$$2(f-u) - 2\lambda_s u_{xx} - 2\lambda_s u_{yy} = 0. \tag{32}$$

Rearranging gives us:

$$f = u - \lambda_s (u_{xx} + u_{yy}) = u - \lambda_s \nabla \cdot u. \tag{33}$$

which is exactly the sharpen filter based on Laplacian subtraction.

Note that we do not claim that the form of the objective function (Equation 30) is unique. Indeed, adding other functions that do not include f or its derivatives to the integrand L_{LS} will yield the same result. Rather, it illustrates one objective function that has intuitive meeting and does lead to the Laplacian subtraction filter.

It is also instructive to relate the integrand L_{LS} to the integrand in gradient amplification:

$$L_{\mathrm{GA}} = \lambda_d (f-u)^2 + \|\nabla f - c_s \nabla u\|^2 \tag{34}$$

$$= \lambda_d (f-u)^2 + (f_x - c_s u_x)^2 + (f_y - c_s u_y)^2. \tag{35}$$

We can divide this integrand by λ_d, because it does not affect the minimum solution. Doing so and expanding gives us:

$$L_{\mathrm{GA}} = (f-u)^2 + \frac{1}{\lambda_d} f_x^2 - 2\frac{c_s}{\lambda_d} f_x u_x + \frac{c_s^2}{\lambda_d} u_x^2 + \frac{1}{\lambda_d} f_y^2 - 2\frac{c_s}{\lambda_d} f_y u_y + \frac{c_s^2}{\lambda_d} u_y^2. \tag{36}$$

We can drop the terms that do not depend on f or its derivatives, since again these do not affect the solution. Thus, we can omit $\frac{c_s^2}{\lambda_d} u_x^2$ and $\frac{c_s^2}{\lambda_d} u_y^2$. After also substituting $\lambda_s = c_s/\lambda_d$, we arrive at:

$$L_{\mathrm{GA}} = (f-u)^2 + \frac{1}{\lambda_d} f_x^2 - 2\lambda_s f_x u_x + \frac{1}{\lambda_d} f_y^2 - 2\lambda_s f_y u_y \tag{37}$$

$$= (f-u)^2 - 2\lambda_s \nabla f \cdot \nabla u + \frac{1}{\lambda_d} \|\nabla f\|^2. \tag{38}$$

We can now see that the gradient amplification integrand can be related to the Laplacian subtraction integrand very simply:

$$L_{\mathrm{GA}} = L_{\mathrm{LS}} + \frac{1}{\lambda_d} \|\nabla f\|^2. \tag{39}$$

Again, as $\lambda_d \to \infty$ while holding λ_s constant, the gradient amplification integrand becomes the Laplacian subtraction integrand.

References

1. Trottenberg, U., Oosterlee, C.W., Schller, A.: Multigrid. Academic Press (2000)
2. Ho, J., Lim, J., Yang, M., Kriegman, D.: Integrating surface normal vectors using fast marching method. In: European Conference on Computer Vision. (2006) III: 239–250
3. Strang, G.: Introduction to Applied Mathematics. Wellesley-Cambridge Press (1986)
4. Fetter, A.L., Walecka, J.D.: Theoretical Mechanics of Particles and Continua. Courier Dover (2003)
5. Agarwala, A.: Efficient gradient-domain compositing using quadtrees. ACM Trans. Graph. **26** (2007) 94:1–94:5
6. Kazhdan, M., Hoppe, H.: Streaming multigrid for gradient-domain operations on large images. ACM Transactions on Graphics (Proc. of ACM SIGGRAPH 2008, to appear) (2008)

7. Simchony, T., Chellappa, R., Shao, M.: Direct analytical methods for solving poisson equations in computer vision problems. Pattern Analysis and Machine Intelligence, IEEE Transactions on **12** (May 1990) 435–446
8. Horn, B.: Robot Vision. MIT Press (1986)
9. Zickler, T.E., Belhumeur, P.N., Kriegman, D.J.: Helmholtz stereopsis: Exploiting reciprocity for surface reconstruction. Int. J. Comput. Vision **49** (2002) 215–227
10. Fattal, R., Lischinski, D., Werman, M.: Gradient domain high dynamic range compression. In: SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM Press (2002) 249–256
11. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. In: SIGGRAPH '03: ACM SIGGRAPH 2003 Papers, New York, NY, USA, ACM Press (2003) 313–318
12. Agarwala, A., Dontcheva, M., Agrawala, M., Drucker, S., Colburn, A., Curless, B., Salesin, D., Cohen, M.: Interactive digital photomontage. ACM Trans. Graph. **23** (2004) 294–302
13. Horn, B.K.P.: Height and gradient from shading. Int. J. Comput. Vision **5** (1990) 37–75
14. Horovitz, I., Kiryati, N.: Depth from gradient fields and control points: bias correction in photometric stereo. Image Vision Comput. **22** (2004) 681–694
15. Nehab, D., Rusinkiewicz, S., Davis, J., Ramamoorthi, R.: Efficiently combining positions and normals for precise 3d geometry. ACM Trans. Graph. **24** (2005) 536–543
16. Ng, H., Wu, T., Tang, C.: Surface-from-gradients with incomplete data for single view modeling. In: International Conference on Computer Vision. (2007) 1–8
17. Lischinski, D., Farbman, Z., Uyttendaele, M., Szeliski, R.: Interactive local adjustment of tonal values. In: SIGGRAPH '06: ACM SIGGRAPH 2006 Papers, New York, NY, USA, ACM Press (2006) 646–653
18. Bhat, P., Zitnick, L., Cohen, M., Curless, B.: A perceptually-motivated optimization-framework for image and video processing. Technical Report UW-CSE-08-06-02, University of Washington (2008)
19. Agrawal, A.K., Raskar, R., Chellappa, R.: What is the range of surface reconstructions from a gradient field? In: ECCV (1). (2006) 578–591
20. Agrawal, A.: Scene Analysis under Variable Illumination using Gradient Domain Methods. PhD thesis, University of Maryland (2006)
21. Szeliski, R.: Fast surface interpolation using hierarchical basis functions. IEEE Trans. Pattern Anal. Mach. Intell. **12** (1990) 513–528
22. Frankot, R.T., Chellappa, R.: A method for enforcing integrability in shape from shading algorithms. IEEE Trans. Pattern Anal. Mach. Intell. **10** (1988) 439–451
23. Georghiades, A.S., Belhumeur, P.N., Kriegman, D.J.: From few to many: Illumination cone models for face recognition under variable lighting and pose. IEEE Transactions on Pattern Analysis and Machine Intelligence **23** (2001) 643–660
24. Weiss, Y.: Deriving intrinsic images from image sequences. In: International Conference on Computer Vision. (2001) II: 68–75
25. Bracewell, R.: The Fourier Transform and Its Applications. second edn. Mcgraw-Hill (1986)
26. Frigo, M., Johnson, S.G.: FFTW for version 3.0 (2003)
27. Szeliski, R.: Locally adapted hierarchical basis preconditioning. In: SIGGRAPH '06: ACM SIGGRAPH 2006 Papers, New York, NY, USA, ACM Press (2006) 1135–1143