Solving Vision Problems via Filtering

Sean I. Young[1] sean0@stanford.edu
Aous T. Naman[2] aous@unsw.edu.au
Bernd Girod[1] bgirod@stanford.edu
David Taubman[2] d.taubman@unsw.edu.au

[1]Stanford University [2]University of New South Wales

Abstract

We propose a new, filtering approach for solving a large number of regularized inverse problems commonly found in computer vision. Traditionally, such problems are solved by finding the solution to the system of equations that expresses the first-order optimality conditions of the problem. This can be slow if the system of equations is dense due to the use of nonlocal regularization, necessitating iterative solvers such as successive over-relaxation or conjugate gradients. In this paper, we show that similar solutions can be obtained more easily via filtering, obviating the need to solve a potentially dense system of equations using slow iterative methods. Our filtered solutions are very similar to the true ones, but often up to 10 times faster to compute.

1. Introduction

Inverse problems are mathematical problems where one's objective is to recover a latent variable given observed input data. In computer vision, a classic inverse problem is that of estimating the optical flow [1], where the goal is to recover the apparent motion between an image pair. The problems of image super-resolution, denoising, deblurring, disparity and illumination estimation are examples of inverse problems in imaging and computer vision [2]–[5]. The ubiquity of these inverse problems for real-time computer vision applications places significant importance on efficient numerical solvers for such inverse problems. Traditionally, an inverse problem is formulated as a regularized optimization problem and the optimization problem then solved by finding the solution to its first-order optimality conditions, which can be expressed as a system of linear (or linearized) equations.

Recently, edge-preserving regularizers based on bilateral or nonlocal means weighting have found use in many vision problems [5]–[7]. Whereas such nonlocal regularizers often produce better solutions than local ones, they generate dense systems of equations that in practice can only be solved via slow numerical methods like successive over-relaxation and conjugate gradients. Such numerical methods are inherently iterative, and are sensitive to the conditioning of the overall problem. Iterative methods such as conjugate gradients also require the problem to be symmetric (and semi-definite).

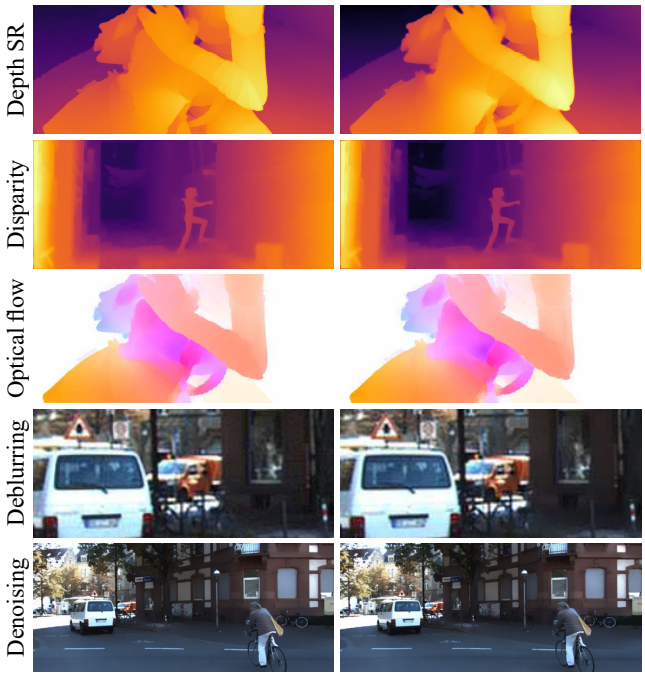

Figure 1. Solving regularized inverse problems in vision typically requires using iterative solvers like conjugate gradients. We solve the same type of problems via filtering for a 10× speed-up.

In this work, we solve regularized optimization problems of the form

$$\text{minimize} \;\; f(\mathbf{u}) = \|\mathbf{H}\mathbf{u} - \mathbf{z}\|_2^2 + \lambda \mathbf{u}^* \mathbf{L} \mathbf{u} \qquad (1)$$

using fast non-iterative filtering, obviating the need to solve dense systems of linear equations produced by geodesic and bilateral regularizers for example. We validate our approach on three classic vision problems: optical flow (and disparity) estimation, depth superresolution, and image deblurring and denoising, all of which are expressible in the form (1). Our filtered solutions to such problems are all very similar to the the true ones as seen in Figure 1, but 10× faster to compute in some cases. Compared to the fast bilateral solver [5], our formalism is not specific to the bilateral regularizer, and can solve more advanced inverse problems such as the disparity and the optical flow estimation problems.

2. Inverse Problems

One feature of many inverse problems is that they either do not have a unique solution, or the solution is unstable—it does not depend continuously on the input. We refer to such problems as ill-posed. Therefore, inverse problems are often reformulated for uniqueness and stability. The reformulation can be demonstrated with a simple least-squares problem of the form

$$\text{minimize} \;\; f(\mathbf{u}) = \|\mathbf{H}\mathbf{u} - \mathbf{z}\|_2^2, \tag{2}$$

in which $\mathbf{H} \in \mathbb{R}^{n \times m}$, $\mathbf{z} \in \mathbb{R}^m$. Problem (2) admits infinitely many solutions when $n \leq m$, failing the uniqueness test, so a reformulation of (2) is needed in this case.

Even when $n > m$, problem (2) can still fail the stability test. Consider the problem instance with input data

$$\mathbf{H} = \begin{bmatrix} 1.0 & 0.0 \\ 1.0 & 0.0 \\ 0.9 & 0.1 \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} 1.0 \\ 1.0 \\ 1.0 \end{bmatrix} + \boldsymbol{\epsilon}, \tag{3}$$

for example. One can consider $\boldsymbol{\epsilon}$ a perturbation on the exact right-hand side vector of $\mathbf{1}$—unless $\boldsymbol{\epsilon} = \mathbf{0}$, there is no vector $\mathbf{u}$ such that $\mathbf{H}\mathbf{u} = \mathbf{z}$. While problem (2) admits the (unique) solution $\mathbf{u}^{\text{ls}} = \mathbf{H}^\dagger \mathbf{z} = (\mathbf{H}^*\mathbf{H})^{-1}\mathbf{H}^*\mathbf{z}$, the solution becomes unduly influenced by perturbation if $\boldsymbol{\epsilon}$ lies along a particular direction. This direction is $\mathbf{H}\mathbf{u}_1$, where $\mathbf{u}_1$ is a vector along the minor eigen-axis of $\mathbf{H}^*\mathbf{H}$. This calls for a reformulation of (2) similarly to the case where $n \leq m$.

2.1. Regularization

In computer vision problems, $\mathbf{u}$ often represents a hidden field of variables (such as the scene depth), each element of $\mathbf{z}$ associated with a particular pixel location in the image. In such problems, (2) is often reformulated by regularization:

$$\text{minimize} \;\; f(\mathbf{u}) = \|\mathbf{H}\mathbf{u} - \mathbf{z}\|_2^2 + \lambda \mathbf{u}^*\mathbf{L}\mathbf{u}, \tag{4}$$

in which $\mathbf{L} \in \mathbb{S}_+^{n \times n}$ is a (graph) Laplacian matrix penalizing the changes between adjacent vertices, and parameter $\lambda > 0$ specifies a tradeoff between the fidelity of the solution to the input $(\mathbf{H}, \mathbf{z})$ and solution smoothness. Problem (4) admits a unique solution when $\mathbf{ker}(\mathbf{H}^*\mathbf{H}) \cap \mathbf{ker}(\mathbf{L}) = \{\mathbf{0}\}$, and this condition holds under most circumstances since $\mathbf{L}$ is often a high-pass operator corresponding to the Laplacian matrix of some graph whereas $\mathbf{H}^*\mathbf{H}$ is a low-pass operator (for image deblurring), or a non-negative diagonal matrix (for disparity and optical flow estimation). Since problem (4) is quadratic in $\mathbf{u}$, its solution may be expressed in closed form concisely as $\mathbf{u}^{\text{opt}} = (\mathbf{H}^*\mathbf{H} + \lambda\mathbf{L})^{-1}\mathbf{H}^*\mathbf{z}$.

Despite the simplicity, the objective of problem (4) has a sufficiently general form, and suitably defining $\mathbf{H}$ expresses most inverse problems in vision and imaging like depth and optical flow estimation, depth super-resolution, colorization [6], image inpainting [8], de-blurring and de-noising [4]. By suitably defining $\mathbf{L}$, the objective of (4) expresses both local [1], [2] and non-local [5]–[11] regularity terms. Problem (4) is also sufficiently general to express non-quadratic models based on, for example, Charbonnier and Huber losses.

One notable non-quadratic objective is the total-variation function of Rudin *et al.* [2]

$$\text{minimize} \;\; f(\mathbf{u}) = \|\mathbf{H}\mathbf{u} - \mathbf{z}\|_2^2 + \lambda \|w(\mathbf{K}\mathbf{u})\|_1, \tag{5}$$

in which $w(x) = |x|$, and $\mathbf{K}$ is the difference matrix, so that $\mathbf{L} = \mathbf{K}^*\mathbf{K}$. Although (5) appears quite different from (4), it is shown by Chambolle and Lions [12] that (5) can readily be solved using the lagged diffusivity method (or iteratively re-weighted least-squares), which solves in the kth iteration the least-squares problem

$$\text{minimize} \;\; f^{k+1}(\mathbf{u}) = \|\mathbf{H}\mathbf{u} - \mathbf{z}\|_2^2 + \lambda \mathbf{u}^*\mathbf{L}^k\mathbf{u}, \tag{6}$$

in which

$$\mathbf{L}^k = \mathbf{K}^* \, \mathbf{diag}(\text{abs}(\mathbf{K}\mathbf{u}^k))^\dagger \, \mathbf{K}, \tag{7}$$

and $\mathbf{u}^k$ is the minimizer of f^k. Since each problem (6) is in the same form as (4), we do not need to separately consider a fast method for solving (5).

2.2. Local vs Non-Local

Solving regularized inverse problems of the form (4) can be traced back to Phillips [13], Tikhonov [14], and Twomey [15], [16] in the one-dimensional case, which was extended to the two-dimensional case by Hunt [17]. A popular choice of $\mathbf{L}$ in two dimensions is one based on the finite-difference (fd) or the finite-element (fe) stencils, which are

$$\mathbb{L}_{\text{fd}} = \begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}, \mathbb{L}_{\text{fe}} = \begin{bmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{bmatrix}, \tag{8}$$

respectively. The latter is used by Horn and Schunck [1].

Gilboa and Osher [8] demonstrate the benefits of using a non-local Laplacian for image denoising and inpainting. As the authors pointed out, their non-local Laplacian is itself an adaptation of graph Laplacians of [18]. Given an N-sample image whose N vertices are $\mathbf{p}_n$, $1 \leq n \leq N$, we can define a graph Laplacian over the vertices as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, with $\mathbf{A}$ denoting the weighted adjacency matrix of some graph over the vertices $\{\mathbf{p}_n\}$, and $\mathbf{D} = \mathbf{diag}(\mathbf{A}\mathbf{1})$ is the degree matrix of this graph. In vision applications, the weighted adjacency between $\mathbf{p}_n$ and $\mathbf{p}_m$ is usually a function of $\|\mathbf{p}_n - \mathbf{p}_m\|$, so one can define $\mathbf{A}$ as $a_{mn} = r(\|\mathbf{p}_n - \mathbf{p}_m\|)$ in terms of some non-increasing function $r \in \mathbb{R}_+ \to \mathbb{R}_+$.

2.3. Bilateral vs Geodesic

One notable graph Laplacian is inspired by the success of the bilateral filter [19], [20]. Suppose we have an N-sample image $\mathbf{z} \in \mathbb{R}^N \cong \mathbb{R}^D$, whose sample locations are the points D of a rectangular grid in the x-y plane. The bilateral-space representation [21] of the image vertices is

$$\mathbf{p}_n = \begin{pmatrix} \frac{x_n}{\sigma_X} & \frac{y_n}{\sigma_Y} & \frac{z_n}{\sigma_Z} \end{pmatrix} \in D \oplus [0{,}255], \tag{9}$$

in which $\sigma_{X,Y,Z}$ are the scales of the bilateral space in their respective dimensions. If we define the graph adjacencies $\mathbf{A}$ over $\mathbf{p}_n$ as $a_{mn} = \mathrm{e}^{-|\mathbf{p}_n - \mathbf{p}_m|^2/2}$, then $\mathbf{D}^\dagger\mathbf{A}$ and $\mathbf{D} - \mathbf{A}$ are respectively, the bilateral filter and the bilaterally-weighted graph Laplacian matrices. Observe that when $\sigma_Z = \infty$, $\mathbf{A}$ is simply a Gaussian blur operator with scales σ_X and σ_Y.

Another graph Laplacian often found in edge-preserving regularization is one based on the geodesic distance. In such a case, matrix $\mathbf{A}$ is defined as $a_{mn} = \mathrm{e}^{-\mathrm{geod}(\mathbf{p}_n, \mathbf{p}_m)}$, where $\mathrm{geod}(\mathbf{p}_n, \mathbf{p}_m)$ is the distance of the shortest path from point $\mathbf{p}_n$ to point $\mathbf{p}_m$ on the two-dimensional manifold defined by the vertices $\{\mathbf{p}_n\}$. That is,

$$\mathrm{geod}(\mathbf{p}_n, \mathbf{p}_m) = \min_{\substack{p, (\mathbf{v}_i)_{1 \le i \le p} : \mathbf{v}_i \sim \mathbf{v}_{i+1}, \\ \mathbf{p}_n = \mathbf{v}_1, \mathbf{v}_p = \mathbf{p}_m}} \sum_{i=1}^{p-1} |\mathbf{v}_i - \mathbf{v}_{i+1}|, \tag{10}$$

in which $\mathbf{v} \sim \mathbf{v}'$ means that $\mathbf{v}$ and $\mathbf{v}'$ are adjacent pixels on the two-dimensional grid.

Since bilateral and geodesic graph Laplacians often have degrees that differ across vertices, normalization is typically applied for more uniform regularization. The most common form of normalization is $\widetilde{\mathbf{L}} = \mathbf{D}^{\dagger/2}\mathbf{L}\mathbf{D}^{\dagger/2}$, referred to as the symmetric-normalized Laplacian, and $\widehat{\mathbf{L}} = \mathbf{D}^\dagger\mathbf{L}$, referred to as the random-walk normalized Laplacian [18], [22]. Barron *et al.* [7] use the Sinkhorn-normalized form [23], [24] of the bilateral-weighted graph Laplacian. By contrast, Laplacians based on stencils (8) are already normalized up to a constant scaling factor (except possibly at the image boundaries).

3. Related Work

Whereas the solution $\mathbf{u}^{\mathrm{opt}} = (\mathbf{H}^*\mathbf{H} + \lambda\mathbf{L})^{-1}\mathbf{H}^*\mathbf{z}$ of (4) is simple, its numerical evaluation can be expensive. Except in a handful of scenarios, $\mathbf{u}^{\mathrm{opt}}$ must be evaluated iteratively using numerical methods such as successive over-relaxation or conjugate gradients, both of which require us to evaluate the mappings $\mathbf{t} \mapsto \mathbf{H}^*\mathbf{H}\mathbf{t}$ and $\mathbf{t} \mapsto \mathbf{L}\mathbf{t}$ repeatedly. The latter mapping can be particularly expensive to evaluate if $\mathbf{L}$ has a nonlocal (dense) matrix structure. Krylov-subspace methods like conjugate gradients additionally require the spectrum of $\mathbf{H}^*\mathbf{H} + \lambda\mathbf{L}$ to be clustered for faster convergence.

3.1. Fast Solvers

For optical flow estimation, Krähenbühl and Koltun [11] consider the bilaterally-regularized instance of (4), but with the Charbonnier penalty for regularization. They essentially use the fact that $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{A}$ is the unnormalized bilateral filter, and evaluate the mapping $\mathbf{t} \mapsto \mathbf{A}\mathbf{t}$ efficiently inside conjugate gradients with a fast implementation of the bilateral filter. However, ten or more iterations of conjugate gradients are usually required even when preconditioning is used, which is not as efficient as a non-iterative approach.

Barron and Poole [5] propose their bilateral solver for the specific case where $\mathbf{L}$ is bilateral-weighted, and $\mathbf{H}$ is square and diagonal. Forming the Laplacian $\widetilde{\mathbf{L}} = \mathbf{I} - \widetilde{\mathbf{A}}$ in terms of the bi-stochasticized $\widetilde{\mathbf{A}}$, they factorize $\widetilde{\mathbf{A}} = \mathbf{S}\mathbf{B}\mathbf{S}^*$, where $\mathbf{S}$ and $\mathbf{B}$ are the slice and the blur operators respectively. They reformulate problem (4) in terms of $\mathbf{u} = \mathbf{S}\mathbf{y}$ as

$$\text{minimize} \;\; f(\mathbf{y}) = \left\|\widehat{\mathbf{H}}\mathbf{y} - \hat{\mathbf{z}}\right\|_2^2 + \lambda\mathbf{y}^*(\mathbf{I} - \mathbf{B})\mathbf{y}, \tag{11}$$

the solution $\mathbf{y}^{\mathrm{opt}}$ of which is obtained using pre-conditioned conjugate gradients. The solution of the original problem (4) is finally obtained as $\mathbf{u}^{\mathrm{opt}} \approx \mathbf{S}\mathbf{y}^{\mathrm{opt}}$.

Although the bilateral solver produces efficient solutions in practice, the solver is iterative, and does not generalize to problems with other edge-preserving regularizers. Also, the solution $\mathbf{S}\mathbf{y}^{\mathrm{opt}}$ suffers from block artifacts, requiring further post-processing by a second edge-preserving filter, as stated by the authors themselves.

3.2. Fast Filtering

Fast solvers like the bilateral solver ultimately depend on the ability to perform bilateral filtering efficiently. Many fast bilateral filtering methods have been proposed. They include the adaptive manifold [25], the Gaussian KD-tree [26], and the permutohedral lattice [21] filters. All of them exploit the fact that filtering with a large kernel can also be achieved by (i) down-sampling the input, (ii) filtering the down-sampled signal using a smaller filter kernel, finally (iii) up-sampling the filtered signal. This series of operations is often referred to as the splat-blur-slice pipeline. Such a pipeline guarantees a computational complexity constant in the size of the filter kernel. By contrast, the complexity of a naïve bilateral filter implementation would scale linearly with kernel size.

Similarly, efficient geodesic regularization depends upon efficient geodesic filtering. Fast implementations of the filter include the geodesic distance transform [27] and the domain transforms [28]. Since geodesic filtering requires computing the shortest path between every pair of vertices (10), a naïve implementation of the filter would be quite expensive. As an example, if Dijkstra's algorithm is used to find all pixelwise shortest paths, geodesic filtering would have an $O(N^3)$ cost in the number N of pixels.

4. Our Filtering Method

We now present the main result of our work. We assume that $\mathbf{L}$ is Sinkhorn-normalized as in [5], although this is not required in practical implementations. The proposed method originates from the observation that without regularization,

$$\operatorname*{argmin}_{\mathbf{u}} \|\mathbf{H}\mathbf{u} - \mathbf{z}\|_2^2 = \operatorname*{argmin}_{\mathbf{u}} \|\mathbf{H}(\mathbf{u} - \mathbf{H}^\dagger\mathbf{z})\|_2^2, \tag{12}$$

so we can consider $\mathbf{H}^\dagger\mathbf{z}$ some transformed signal to filter by least-squares using the weights (inverse covariance matrix)

$\mathbf{H}^*\mathbf{H}$ (but the problem is still ill-posed). Note the structural (in contrast to the numeric) pseudo-inverse of $\mathbf{H} \in \mathbb{R}^{n\times m}$ is defined as

$$\mathbf{H}^\dagger = \begin{cases} \mathbf{H}^*(\mathbf{H}\mathbf{H}^*)^{-1} & \text{if } n \le m \\ (\mathbf{H}^*\mathbf{H})^{-1}\mathbf{H}^* & \text{if } n > m\,, \end{cases} \tag{13}$$

so whereas relationship (12) always holds, it is generally *not* the case that $\|\mathbf{H}\mathbf{u}-\mathbf{z}\|_2^2 = \|\mathbf{H}(\mathbf{u}-\mathbf{H}^\dagger\mathbf{z})\|_2^2$ when $n > m$.

Using the weighing $\mathbf{C} = \mathbf{H}^*\mathbf{H}$, we propose to obtain the solution of the regularized problem (4) non-iteratively as[1]

$$\begin{aligned} \mathbf{u}^{\text{filt}} &= \mathbf{F}^\dagger\mathbf{A}\mathbf{C}\mathbf{H}^\dagger\mathbf{z} \\ &= \mathbf{F}^\dagger\mathbf{A}\mathbf{H}^*\mathbf{z}, \end{aligned} \qquad \mathbf{F} = \mathbf{diag}(\mathbf{A}\mathbf{C}\mathbf{1}), \tag{14a}$$

in which $\mathbf{A}$ is the graph adjacency (or low-pass filter) matrix of $\mathbf{L}$. Said simply, $\mathbf{u}^{\text{filt}}$ is filtering of the naïve solution $\mathbf{H}^\dagger\mathbf{z}$ of the ill-posed problem (2) using the least-squares weights $\mathbf{C}$, normalized by $\mathbf{F}$ to preserve the mean of the signal. This is the idea of normalized convolution [29] applied to solving regularized inverse problems.

For some instances of problem (3), the weighting by $\mathbf{C}$ is neither necessary nor desirable. In the deblurring instance of problem (2) for example, the original ill-posed problem is to solve $\mathbf{H}\mathbf{u}-\mathbf{z}=\mathbf{0}$ for $\mathbf{u}$. The blur operator $\mathbf{H}$ is structurally (but not numerically) invertible, and it would have been just as valid to formulate the regularized inverse problem as[2]

$$\text{minimize } f(\mathbf{u}) = \|\mathbf{u}-\mathbf{H}^\dagger\mathbf{z}\|_2^2 + \lambda\mathbf{u}^*\mathbf{L}\mathbf{u}, \tag{15}$$

in which case the weighting by $\mathbf{C}$ disappears. Depending on the application, we therefore use the de-weighted variant of our filtering strategy

$$\hat{\mathbf{u}}^{\text{filt}} = \mathbf{F}^\dagger\mathbf{A}\mathbf{H}^\dagger\mathbf{z}, \quad \mathbf{F} = \mathbf{diag}(\mathbf{A}\mathbf{1}), \tag{14b}$$

cf. the original pixelwise weighted formulation (14a).

Note that since $\mathbf{A}$ is a low-pass filter, our approach (14a) is valid only if $(\mathbf{C}+\lambda\mathbf{L})^{-1}$ has a low-pass response. This is fortunately the case for most inverse problems in vision. The image deblurring problem is unique in that $(\mathbf{C}+\lambda\mathbf{L})^{-1}$ has a high-pass response, and is ill-approximated by $\mathbf{A}$. In such a case, one can apply the de-weighted variant of our method (14b) to solve the problem. The supplement discusses this in more detail. We make no specific assumptions regarding the structural rank of $\mathbf{H} \in \mathbb{R}^{n\times m}$, while we continue to assume that $\mathbf{ker}(\mathbf{C}) \cap \mathbf{ker}(\mathbf{L}) = \{\mathbf{0}\}$ for a unique solution.

4.1. Analysis when $\mathbf{C} = \mathbf{I}$

To observe $\mathbf{u}^{\text{filt}} \approx \mathbf{u}^{\text{opt}}$ in (14a), let us consider a simpler instructive instance of problem (4), where $\mathbf{C} = \mathbf{I}$. Then, our solution (14a) can be written as $\mathbf{u}^{\text{filt}} = \mathbf{A}\mathbf{z}$, and the true one as $\mathbf{u}^{\text{opt}} = \mathbf{G}\mathbf{z}$ with $\mathbf{G} = (\mathbf{I}+\lambda\mathbf{L})^{-1}$. Since $\mathbf{L}$ is symmetric and positive semi-definite, we can write $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^*$, where $\mathbf{U}$ are the eigenvectors of $\mathbf{L}$, the corresponding eigenvalues of which are $\mathbf{\Lambda} = \mathbf{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$. We assume λ_n are ordered as $0 = \lambda_1 \le \lambda_2 \le \ldots \le \lambda_N \le 1$.

One can observe that the filter $\mathbf{A} = \mathbf{U}(\mathbf{I}-\mathbf{\Lambda})\mathbf{U}^*$ has the spectral filter [22] factors

$$1 = 1-\lambda_1 \ge 1-\lambda_2 \ge \ldots \ge 1-\lambda_N \ge 0, \tag{16}$$

and $\mathbf{G} = \mathbf{U}(\mathbf{I}+\lambda\mathbf{\Lambda})^{-1}\mathbf{U}^*$ has the factors

$$1 = \frac{1}{1+\lambda_1} \ge \frac{1}{1+\lambda_2} \ge \ldots \ge \frac{1}{1+\lambda_N} \ge \frac{1}{2}, \tag{17}$$

assuming $\lambda = 1$ for simplicity. The eigenvalues of $\mathbf{A}$ decay towards 0 while those of $\mathbf{G}$, towards $1/2$. However, we can easily equalize the two spectral filter responses by applying the mapping $\mathbf{A} \mapsto (\mathbf{A}+\mathbf{I})/2$. In any case, they both have a unit DC gain (or a unit filter response to the constant vector $\mathbf{1}$) as can be seen in the left-hand side of (16)–(17) (the first eigenvector of $\mathbf{L}$ is $N^{-1/2}\mathbf{1}$).

Since the two spectral factors $(\mathbf{I}-\mathbf{\Lambda})$ and $(\mathbf{I}+\mathbf{\Lambda})^{-1}$ are generally not the same, our filtered solutions $\mathbf{u}^{\text{filt}} = \mathbf{A}\mathbf{z}$ are necessarily an approximation of $\mathbf{u}^{\text{opt}} = \mathbf{G}\mathbf{z}$. However, such an approximation is reasonable since our true objective is to obtain a good solution to a vision problem, not to accurately solve problem (4) *per se*.

4.2. Analysis when $\mathbf{C} \neq \mathbf{I}$ but (block-) diagonal

Let us relate our filter solution $\mathbf{u}^{\text{filt}}$ to the true $\mathbf{u}^{\text{opt}}$ when $\mathbf{C}$ is no longer the identity but diagonal. We write $\hat{\mathbf{z}} = \mathbf{H}^\dagger\mathbf{z}$ for convenience. Then, the solution of problem (4) becomes $\mathbf{u}^{\text{opt}} = (\mathbf{C}+\lambda\mathbf{L})^{-1}\mathbf{C}\hat{\mathbf{z}}$ and $\mathbf{u}^{\text{filt}} = \mathbf{F}^\dagger\mathbf{A}\mathbf{C}\hat{\mathbf{z}}$ (14a). Suppose all weights are initially $\mathbf{C} = \mathbf{I}$ as in 4.1. To observe how the solution $\mathbf{u}^{\text{opt}}$ changes when an arbitrary weight c_{nn} is set to $1-\epsilon_n$, we invoke the Sherman-Morrison formula to write

$$\begin{aligned} (\mathbf{C}+\lambda\mathbf{L})^{-1} &= (\mathbf{I}+\lambda\mathbf{L}-\epsilon\mathbf{d}_n\mathbf{d}_n^*)^{-1} \\ &= \mathbf{G} + \frac{\epsilon_n\mathbf{G}\mathbf{d}_n\mathbf{d}_n^*\mathbf{G}^*}{1-\epsilon_n\mathbf{d}_n^*\mathbf{G}\mathbf{d}_n} \\ &= \mathbf{G} + \alpha_n\mathbf{g}_n\mathbf{g}_n^*, \end{aligned} \tag{18}$$

in which

$$\alpha_n = \frac{\epsilon_n}{1-\epsilon_n g_{nn}}, \quad 0 \le \alpha_n \le 1, \tag{19}$$

and $\mathbf{d}_n$ is the nth column of the identity matrix.

The equalities (18) tell us that setting $c_{nn} = 1-\epsilon_n$ adds $\alpha_n\mathbf{g}_n\mathbf{g}_n^*$ to $\mathbf{G}$, which is the unique adjustment guaranteeing that $(\mathbf{G}+\alpha_n\mathbf{g}_n\mathbf{g}_n^*)\mathbf{C}$ has a unit row-sum. This adjustment is small if $\mathbf{G}$ has a large effective filter scale since $\mathbf{g}_n^*\mathbf{1} = 1$ implies the elements of $\mathbf{g}_n$ are small. We similarly guarantee that the filter $\mathbf{F}^\dagger\mathbf{A}\mathbf{C}$ has a unit row-sum but normalizing by $\mathbf{F}^\dagger$ explicitly. A similar argument to (18)–(19) may be given for the general vectorial case where $\mathbf{C}$ is block-diagonal, as is the case in the optical flow estimation problem.

[1] Despite the cosmetic resemblance, $\mathbf{F}^\dagger\mathbf{A}$ has no relationship to iteration matrices seen in e.g. the Jacobi, Gauss-Seidel or SOR methods.

[2] Numerically, $\mathbf{H}^\dagger$ is computed using the truncated SVD in the general case, or efficiently via the FFT if $\mathbf{H}$ is shift-invariant (e.g. a blur operator).

Figure 2. The 16× super-resolution disparity maps produced using the geodesic and the bilateral variants of our method for the 1088 × 1376 **Art** scene. Best viewed online by zooming in. Results are typical (more results are available in the supplement).

5. Robust Estimation

Our filtering method (14) can be robustified by changing the filter $\mathbf{A}$ in the graph domain. By augmenting the vertices (9) of the underlying graph as

$$\mathbf{p}_n = \Big(\frac{x_n}{\sigma_X} \quad \frac{y_n}{\sigma_Y} \quad \frac{z_n}{\sigma_Z} \quad \frac{u_n}{\sigma_U}\Big), \tag{20}$$

in which u_n is the nth element of the previous solution, and σ_U is the scale of this solution. (For the optical flow and the illumination estimation problems, both components u_{1n} and u_{2n} are added to $\mathbf{p}_n$ with their respective scales.)

If $\mathbf{A}$ is Gaussianly weighted as $a_{mn} = \mathrm{e}^{-|\mathbf{p}_n - \mathbf{p}_m|^2/2}$, the introduction of u_n corresponds to the use of the Welsch loss for our regularization. However, in the case where $\mathbf{A}$ is the geodesic filter with $a_{mn} = \mathrm{e}^{-\mathrm{geod}(\mathbf{p}_n, \mathbf{p}_m)}$, introducing u_n is difficult to interpret within the established robust estimation framework. Since the Welsch loss

$$w(x) = \sigma^2(1 - \exp(-x^2/2\sigma^2)) \tag{21}$$

is non-convex (and non-homogeneous), the scale parameter σ_U plays an important role in guaranteeing the convexity of our problem. Observing that w is convex across the interval $[-\sigma, \sigma]$, we should set σ such that most of the input to w fall inside this interval. The input may fall outside of the convex interval some of the time as long as the Hessian of the overall objective in (5) is positive semidefinite. Krähenbühl and Koltun [11] on the other hand propose an efficient method to incorporate other convex robust losses w in (5).

6. Solving Vision Problems

We apply our method to a number of vision problems, all of which can be written in the form (4). One may also apply our method to other simpler problems discussed in [5], such as semantic segmentation and colorization, which can all be converted into the form (4).

6.1. Depth Super-resolution

In the depth super-resolution problem [30]–[32], the goal is to upsample a depth map captured by a depth camera to a higher resolution one in an edge-aware manner. For a given low-resolution depth map $\mathbf{z}$, the super-resolution problem is expressed by

$$\text{minimize} \;\; f(\mathbf{u}) = \|\mathbf{H}\mathbf{u} - \mathbf{z}\|_2^2 + \lambda \mathbf{u}^* \mathbf{L}\mathbf{u}, \tag{22}$$

in which the down-sampler $\mathbf{H} = \mathbf{S}\mathbf{B}$, where $\mathbf{B}$ represents a pre-filter (a windowed sinc, in accordance with the Nyquist theorem), and $\mathbf{S}$ is a sub-sampler. We use (14b) to obtain

$$\mathbf{u}^{\mathrm{filt}} = \mathbf{F}^\dagger \mathbf{A}\mathbf{B}^*\mathbf{S}^*\mathbf{z}, \quad \mathbf{F} = \mathbf{diag}(\mathbf{A}\mathbf{B}^*\mathbf{S}^*\mathbf{S}\mathbf{B}\mathbf{1}), \tag{23}$$

so we first upsample $\mathbf{z}$ using $\mathbf{B}^*\mathbf{S}^*$, filter the result using $\mathbf{A}$ and normalize. Figure 2 illustrates depth super-resolution.

6.2. Disparity Estimation

Disparity estimation can be formed as a non-linear least-squares problem at first, and solved iteratively as a series of linear(ized) least-squares problems using the Gauss-Newton algorithm. In the $k+1$th iteration, the estimated disparity is given by the solution of the regularized inverse problem

$$\text{minimize}\, f^{k+1}(\mathbf{u}) = \underbrace{\|\mathbf{Z}_x(\mathbf{u} - \mathbf{u}^k) + \mathbf{z}_t^k\|_2^2}_{d^k(\mathbf{u})} + \lambda \mathbf{u}^* \mathbf{L}\mathbf{u} \tag{24}$$

in which $\mathbf{u}^k$ is the minimizer of f^k, $\mathbf{Z}_x = \mathbf{diag}(\mathbf{z}_x)$ and $\mathbf{z}_t^k$ are the x- and t-derivatives of the image pair, warped using the disparity estimate $\mathbf{u}^k$. We can then write the first term of the objective function of (24) as

$$d^k(\mathbf{u}) = \|\mathbf{Z}_x(\mathbf{u} - \hat{\mathbf{z}}^k)\|_2^2, \quad \hat{\mathbf{z}}^k = \mathbf{u}^k - \mathbf{Z}_x^\dagger \mathbf{z}_t^k, \tag{25}$$

using the relationship (12) on $d^k(\mathbf{u})$. We obtain the $k+1$th estimate of the disparity via filtering

$$\mathbf{u}^{k+1} = \mathbf{F}^\dagger \mathbf{A}(\mathbf{Z}_x^2 \mathbf{u}^k - \mathbf{Z}_x^* \mathbf{z}_t^k), \;\; \mathbf{F} = \mathbf{diag}(\mathbf{A}\mathbf{Z}_x^2 \mathbf{1}). \tag{26}$$

6.3. Optical Flow Estimation

The optical flow estimation problem is a vector extension of disparity estimation (24). Since there are now two values to estimate at each pixel, one may wonder if our method can still be applied. In fact, our formalism remains the same. To recap, the $k+1$th flow estimate is the solution of

$$\begin{aligned}\text{minimize} \;\; f^{k+1}(\mathbf{u}) &= \lambda \mathbf{u}_x^* \mathbf{L}\mathbf{u}_x + \lambda \mathbf{u}_y^* \mathbf{L}\mathbf{u}_y \\ &+ \underbrace{\|(\mathbf{Z}_x, \mathbf{Z}_y)(\mathbf{u} - \mathbf{u}^k) + \mathbf{z}_t^k\|_2^2}_{d^k(\mathbf{u})}\end{aligned} \tag{27}$$

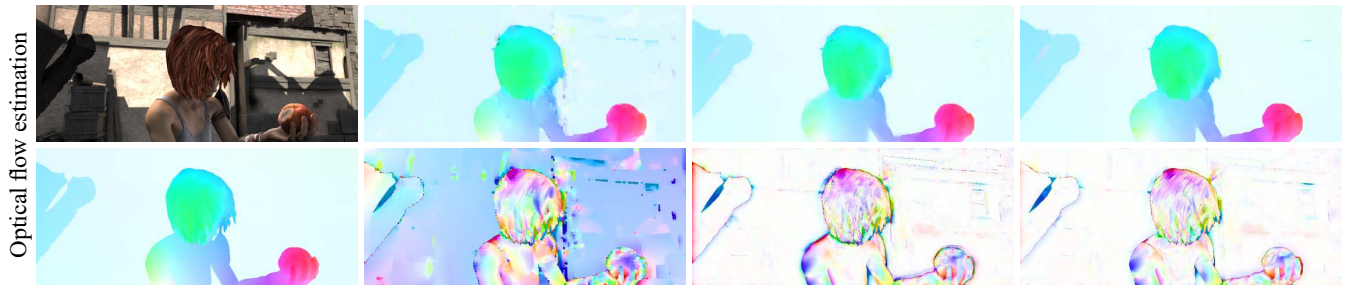

Figure 3. Optical flow (top row) and the corresponding flow error (bottom row) produced using the geodesic and the bilateral variants of our method. The baseline flow is [41] and we perform 3 warping iterations. Whiter pixels correspond to smaller flow vectors.

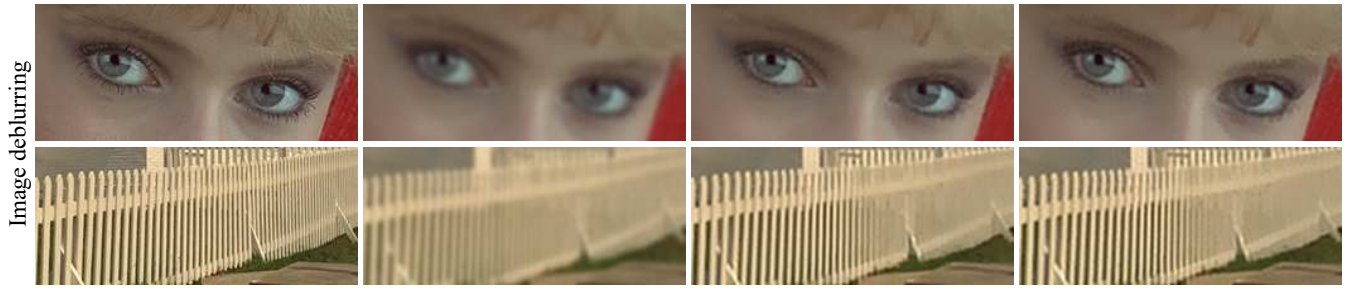

Figure 4. Crops of the deblurred images from the Kodak dataset, produced using the geodesic and the bilateral variants of our method when the standard deviation of the blur kernel is 2. Noise variance is 10^{-5}. Results are typical (more results are available in the supplement).

[33], where $\mathbf{u} = (\mathbf{u}_x^*, \mathbf{u}_y^*)^*$, and $\mathbf{u}^k$, $\mathbf{Z}_x$, and $\mathbf{Z}_y$ are defined similarly as for (24). We can rewrite the last term of (27) as

$$d^k(\mathbf{u}) = \left\|(\mathbf{Z}_x, \mathbf{Z}_y)(\mathbf{u} - \hat{\mathbf{z}}^k)\right\|_2^2, \tag{28}$$

in which $\hat{\mathbf{z}}^k = \mathbf{u}^k - (\mathbf{Z}_x, \mathbf{Z}_y)^\dagger \mathbf{z}_t^k$.

Unlike in the disparity estimation problem, we now have two flow components that cannot be filtered separately—the inverse covariances $(\mathbf{Z}_x, \mathbf{Z}_y)^*(\mathbf{Z}_x, \mathbf{Z}_y)$ now couple the two flow components $\mathbf{u}_x^*$ and $\mathbf{u}_y^*$. The original equation (14a) is therefore generalized to the vectorial case. The new estimate of flow is obtained as

$$\mathbf{u}^{k+1} = \mathbf{F}^\dagger \mathbf{A}(\mathbf{Z}\mathbf{u}^k - (\mathbf{Z}_x, \mathbf{Z}_y)^* \mathbf{z}_t^k), \tag{29}$$

in which

$$\mathbf{A} = \begin{bmatrix} \mathbf{A} & \\ & \mathbf{A} \end{bmatrix}, \qquad \mathbf{Z} = \begin{bmatrix} \mathbf{Z}_x^2 & \mathbf{Z}_x\mathbf{Z}_y \\ \mathbf{Z}_x\mathbf{Z}_y & \mathbf{Z}_y^2 \end{bmatrix}, \tag{30}$$

and

$$\mathbf{F} = \begin{bmatrix} \mathbf{diag}(\mathbf{A}\mathbf{Z}_x^2\mathbf{1}) & \mathbf{diag}(\mathbf{A}\mathbf{Z}_x\mathbf{Z}_y\mathbf{1}) \\ \mathbf{diag}(\mathbf{A}\mathbf{Z}_x\mathbf{Z}_y\mathbf{1}) & \mathbf{diag}(\mathbf{A}\mathbf{Z}_y^2\mathbf{1}) \end{bmatrix}, \tag{31}$$

so that as well as filtering the signal $\mathbf{Z}\mathbf{u}^k - (\mathbf{Z}_x, \mathbf{Z}_y)^*\mathbf{z}_t^k$, we need also to filter $\mathbf{Z}_x^2\mathbf{1}$, $\mathbf{Z}_y^2\mathbf{1}$ and $\mathbf{Z}_x\mathbf{Z}_y\mathbf{1}$.

Observe that the mapping $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$ simply filters the two components of $\mathbf{x}$ separately, whereas the matrices $\mathbf{Z}$ and $\mathbf{F}$ can be permuted to be block-diagonal, whose nth blocks are the 2×2 matrices

$$\mathbf{Z}_n = \begin{bmatrix} z_{xn}^2 & z_{xn}z_{yn} \\ z_{xn}z_{yn} & z_{yn}^2 \end{bmatrix}, \tag{32}$$

and

$$\mathbf{F}_n = \begin{bmatrix} \mathbf{a}_n^* & \\ & \mathbf{a}_n^* \end{bmatrix} \begin{bmatrix} \mathbf{Z}_x^2\mathbf{1} & \mathbf{Z}_x\mathbf{Z}_y\mathbf{1} \\ \mathbf{Z}_x\mathbf{Z}_y\mathbf{1} & \mathbf{Z}_y^2\mathbf{1} \end{bmatrix}, \tag{33}$$

respectively. Essentially, $\mathbf{F}_n$ is a weighted sum of the 2×2 inverse covariance matrices with which to normalize the nth filtered vector. Figure 3 illustrates optical flow estimation.

6.4. Image Deblurring

In the classic image deblurring problem, our objective is to recover a deblurred image from some blurry image $\mathbf{z}$. We use the de-weighted variant (14b) of our method to recover the deblurred image as

$$\mathbf{u}^{\text{filt}} = \mathbf{F}^\dagger \mathbf{A}\mathbf{H}^\dagger \mathbf{z}, \quad \mathbf{F} = \mathbf{diag}(\mathbf{A}\mathbf{1}), \tag{34}$$

in which $\mathbf{H}$ is some known blur operator. When $\mathbf{H} = \mathbf{I}$, (34) simply reduces to edge-aware filtering.

One can express $\mathbf{H}^\dagger = \mathbf{U}\mathbf{\Lambda}^\dagger\mathbf{U}^*$, where $\mathbf{U}$ is the discrete two-dimensional Fourier basis, and $\mathbf{\Lambda}$ is their corresponding magnitude response. We can compute $\mathbf{H}^\dagger\mathbf{z}$ in the frequency domain by multiplying the Fourier coefficients of $\mathbf{z}$ with the inverse magnitude response $\mathbf{\Lambda}^\dagger$, and transforming the result back into the image domain.

For practical implementations, however, one needs to use the numerical definition of $\mathbf{H}^\dagger$. Expressing the blur operator

	Methods	Art				Books				Möbius				Average				Time
		2×	4×	8×	16×	2×	4×	8×	16×	2×	4×	8×	16×	2×	4×	8×	16×	
Other methods	Guided filter [35]	37.13	35.44	33.18	29.83	40.64	39.41	37.45	35.03	40.24	39.13	37.15	35.01	39.19	37.80	35.70	32.93	23.9s
	Diebel and Thrun [34]	37.27	35.24	32.23	28.94	41.85	38.59	35.96	33.93	41.56	38.30	35.79	33.95	39.97	37.24	34.48	31.94	–
	Chan *et al.*[36]	37.40	35.30	32.60	29.63	41.73	39.28	36.58	33.40	41.77	39.54	36.70	33.60	40.05	37.81	35.07	32.01	3.02s
	Park *et al.* [30]	36.63	35.11	32.86	29.29	42.33	39.83	37.76	34.40	42.29	40.21	38.00	35.11	39.98	38.05	35.87	32.52	24.1s
	Yang *et al.* [33]	38.56	36.27	34.42	30.55	42.69	40.60	39.00	35.54	42.46	40.49	38.70	35.32	41.02	38.87	37.11	33.48	–
	Ferstl *et al.*[31]	38.05	35.96	34.01	30.50	44.49	41.24	40.28	37.15	44.78	41.98	39.90	37.25	41.85	39.29	37.56	34.36	140.s
Iterative	Bilateral Solver[3] [5]	40.16	37.24	34.87	31.41	47.58	44.76	42.37	39.56	48.47	45.80	**43.37**	**40.84**	43.70	40.82	38.44	35.15	1.61s
	Geodesic[4] (22)	41.80	37.88	35.41	31.67	48.95	45.53	42.41	39.74	49.67	**46.16**	42.81	39.78	45.25	41.45	38.78	35.27	1.60s
	Bilateral[4] (22)	43.02	38.59	35.94	**32.26**	49.43	45.79	42.96	39.77	**49.82**	45.78	43.20	40.65	46.22	41.96	39.29	**35.82**	8.23s
Ours	Geodesic	41.73	38.31	35.79	31.66	49.06	45.49	42.77	39.70	49.50	46.07	43.23	40.39	45.19	41.75	39.16	35.32	**0.44s**
	Bilateral	**43.63**	**38.98**	**36.15**	32.22	**49.72**	**45.96**	**43.09**	**39.87**	48.89	45.78	42.96	40.30	**46.51**	**42.26**	**39.43**	35.76	1.41s

Table 1. Depth super-resolution performance of different methods. The PSNR (dB) values are of the supperresolution disparity to the ground truth. Running times are for the 16× case. The results for other methods (first six rows) are based on the mean squared errors reported in [5].

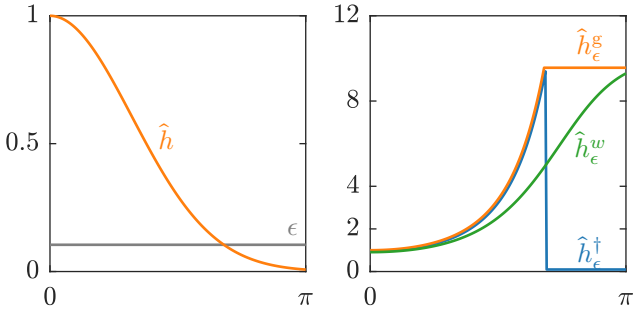

Figure 5. Magnitude responses of a blur kernel (left) and different inverse responses (right). The Wiener response $\hat{h}^w$ varies smoothly across frequencies. The pseudo-inverse response $\hat{h}^\dagger$ is thresholded to zero. Our generalized inverse one $\hat{h}^{\mathrm{g}}$ has a rectified response.

as $\mathbf{H} = \mathbf{F}\mathbf{\Lambda}\mathbf{F}^*$, where $\mathbf{F}$ denote the discrete Fourier vectors and $\mathbf{\Lambda}$ is the diagonal matrix of the magnitude response, we define $\mathbf{H}^\dagger_\epsilon = \mathbf{F}\mathbf{\Lambda}^\dagger_\epsilon\mathbf{F}^*$, where

$$(\mathbf{\Lambda}^\dagger_\epsilon)_n = \begin{cases} \lambda_n^{-1} & \text{if } \lambda_n > \epsilon\,, \\ 0 & \text{otherwise} \end{cases} \tag{35}$$

is the nth diagonal element of $\mathbf{\Lambda}^\dagger_\epsilon$. Essentially, the numerical pseudo-inverse $\mathbf{\Lambda}^\dagger_\epsilon$ treats all $\lambda_n \leq \epsilon$ as 0. We can regard our solution $\mathbf{F}^\dagger\mathbf{A}\mathbf{H}^\dagger_\epsilon\mathbf{z}$ as a noiseless Wiener deblurring solution filtered by an edge-aware filter $\mathbf{A}$ which is then normalized.

Another choice of inverse filter is $\mathbf{H}^{\mathrm{g}}_\epsilon = \mathbf{F}\mathbf{\Lambda}^{\mathrm{g}}_\epsilon\mathbf{F}^*$, where

$$(\mathbf{\Lambda}^{\mathrm{g}}_\epsilon)_n = \min(\lambda_n^{-1}, \epsilon^{-1}) \tag{36}$$

and one can verify that $\mathbf{H}^{\mathrm{g}}_\epsilon$ defined via (36) is a generalized inverse but not the pseudo-inverse of $\mathbf{H}$. Since thresholding (35) introduces ringing artifacts in the de-blurred image, the rectified filter factors (36) are preferable over (35). Observe that the generalized inverse $\mathbf{H}^{\mathrm{g}}$ yields the relation

$$\operatorname{argmin} \|\mathbf{H}\mathbf{u} - \mathbf{z}\|_2^2 = \operatorname{argmin} \|\mathbf{H}(\mathbf{u} - \mathbf{H}^{\mathrm{g}}\mathbf{z})\|_2^2 \tag{37}$$

similarly to the relation regarding $\mathbf{H}^\dagger$ in (12). Figure 5 plots the pseudo-inverse and the generalized inverse responses.

[3]Using $\sigma_{X,Y} = 8, \sigma_L = 4, \sigma_{U,V} = 3, \lambda = 4^{s-1/2}$, as suggested in [5].

[4]Here, $\sigma_{X,Y}, \sigma_Z, \sigma_U, \lambda$ are found by separate grid search for each scale.

7. Experimental Results

To demonstrate the proposed method, we implement our filter to solve a few problems from the previous section. The disparity estimation problem (24) is a special case of optical flow estimation (27), so we consider the latter problem only in this section. We use the domain transforms filter [28] and the permutohedral lattice filter [21] implementations for the geodesic filter, and the bilateral filter, respectively. Running times are obtained on a single core of an Intel 2.7GHz Core i7 processor (iMac mid-2011). Note, the bilateral variants of our methods are slower than their geodesic counterparts due solely to the speed of the bilateral filter implementation [21] used. However, the bilateral variants perform slightly better than the geodesic ones.

In all applications, we formulate the graph vertices in the x-y-u-l-a-b space as

$$\mathbf{p}_n = \left(\frac{x_n}{\sigma_X} \;\; \frac{y_n}{\sigma_Y} \;\; \frac{u_n}{\sigma_U} \;\; \frac{\ell_n}{\sigma_Z} \;\; \frac{a_n}{\sigma_Z} \;\; \frac{b_n}{\sigma_Z}\right), \tag{38}$$

and optimize $\sigma_{X,Y}$, σ_Z and σ_U using grid search separately for each problem. The results for our two iterative solutions in Tables 1 and 2 (geodesic and bilateral) are computed with conjugate gradients (24 iterations, norm tolerance of 10^{-6}).

7.1. Depth Super-resolution

Using the depth map super-resolution dataset of [30], we measure the accuracy and efficiency of our super-resolution method based on our filtering formalism. The method is also compared with a number of other well-performing ones. We assume $\mathbf{B}$ in (23) is the lanczos3 windowed sinc resampling operator. We set our filter scales adaptively using

$$\sigma_{X,Y} = s + 2, \;\; \sigma_Z = 160/s\,, \;\; \sigma_U = s + 10 \tag{39}$$

for the bilateral variant of our method, and

$$\sigma_{X,Y} = 3s, \;\; \sigma_Z = 48, \;\; \sigma_U = 16\sqrt{s} \tag{40}$$

for the geodesic variant (s is the super-resolution factor).

Table 1 lists the peak SNR for the depth super-resolution

Sequence	Initial EPE	Iterative solutions					Ours	
		HS[5]	NL	Epic	Geo[6]	Bilat[7]	Geo	Bilat
alley1	0.797	0.438	0.232	0.280	0.256	0.248	**0.231**	**0.228**
alley2	0.741	0.381	0.257	**0.244**	0.279	0.273	0.245	0.252
ambush7	0.738	2.436	0.573	**0.538**	0.592	0.566	0.577	0.549
bamboo1	0.893	0.473	**0.335**	0.390	0.345	0.346	0.343	0.351
bamboo2	1.969	2.322	1.543	1.562	1.543	**1.536**	1.556	1.561
bandage1	0.999	0.973	**0.578**	0.610	0.598	0.603	0.600	0.606
bandage2	0.619	0.516	**0.294**	0.296	0.304	0.302	0.304	0.305
cave4	3.940	5.822	**3.503**	3.567	3.610	3.587	3.583	3.544
market2	1.100	1.155	**0.619**	0.635	0.650	0.628	0.648	0.638
mountain1	0.817	0.471	0.409	**0.379**	0.429	0.442	0.388	0.390
shaman2	0.514	0.239	**0.182**	0.206	0.215	0.205	0.198	0.191
shaman3	0.589	0.279	0.180	0.174	0.193	0.174	0.182	**0.167**
sleeping1	0.486	0.134	0.110	**0.082**	0.110	0.111	0.087	0.093
temple2	2.508	4.537	**1.993**	2.011	2.041	2.032	2.037	2.022
Average	1.194	1.441	**0.772**	0.784	0.797	0.790	0.784	0.778
Time	–	0.53s	18.2s	1.19s	2.69s	14.4s	**0.65s**	3.32s

Table 2. Average EPE on MPI-Sintel (3 warping stages). All flow initialized using [41]. In each warping iteration, EpicFlow, NL and HS use SOR (iterative), while we use non-iterative filtering.

methods and their running times. The results in the top rows (other methods) are computed using Table 2 of [5] (from the supplement). The results of Bilateral Solver [5] are obtained using the publicly available code. Our two filtering methods are 1–100 times faster than most methods specialized to the super-resolution application. Figure 2 shows our $16\times$ depth maps obtained using the geodesic variant of our method.

7.2. Optical Flow Estimation

Using the training set of the MPI-Sintel optical flow data set [34], we now compare the accuracy and the efficiency of our filtering method with the iterative variational optimizer of EpicFlow [35] also used by [36]–[40], the Horn-Schunck [1] and the Classic+NL [41] methods. Our iterative bilateral baseline is similar to [11], but uses the Welsch loss in place of the Charbonnier loss for regularity. We initialize the flow (27) using the interpolation of DeepMatching [42]. Both our method and EpicFlow use 3 outer warping iterations. We set our filter parameters adaptively using

$$\sigma_{X,Y} = 10, \;\; \sigma_Z = 12, \;\; \sigma_U = 0.5s \tag{41}$$

for the bilateral variant of our method, in which s is the root mean-square magnitudes of the initial flow vectors, and

$$\sigma_{X,Y} = 20, \;\; \sigma_Z = 96, \;\; \sigma_U = s \tag{42}$$

for the geodesic variant. We set the parameters of EpicFlow to the Sintel settings. Table 2 provides the average endpoint errors and the run times after optical flow estimation.

The geodesic variant of our method has a similar average end-point error as the variational optimizer of EpicFlow (or successive overrelaxation) while being 1.8 times fast. In the timing results, we include the time spent on computation of the elements of $\mathbf{Z}$ (30), which is 0.13s per warping iteration for all methods. Figure 3 visualizes our flow estimates.

[5]Using $\lambda = 40$ and successive over-relaxation (SOR).
[6]Using $\sigma_{X,Y} = 8, \sigma_Z = 48, \sigma_U = 0.5 + 0.25s$ and $\lambda = 2$.
[7]Using $\sigma_{X,Y} = 6, \sigma_Z = 10, \sigma_U = 0.5 + 0.25s$ and $\lambda = 2$.

Blur scale	Input PSNR	FFT-based		Iterative methods			Our methods	
		Wiener	$L2$	TV	Geo	Bilat	Geo	Bilat
$\sqrt{0.5}$	30.70	34.84	35.40	36.27	**36.44**	36.41	36.42	36.29
$\sqrt{1.0}$	28.39	31.93	32.11	32.90	32.79	32.56	**32.95**	**32.95**
$\sqrt{2.0}$	26.64	29.43	29.53	29.86	**30.18**	29.86	30.10	30.13
$\sqrt{4.0}$	25.28	27.62	27.66	28.06	**28.15**	27.86	28.07	28.12
Average	27.75	30.96	31.18	31.77	**31.89**	31.67	**31.89**	31.87
Time	–	0.10s	0.14s	1.46s	0.65s	4.80s	**0.07s**	1.68s

Table 3. Average peak SNR (dB) of the deblurred images (Kodak dataset, 24 images). The Gaussian blur kernels used are discrete B-splines of order $2n$, for $n = 1, 2, 4, 8$. The noise variance is 10^{-5}.

7.3. Deblurring and Denoising

For deblurring, we assume that the point spread function of the blur is known. The blur kernels we use have the form $\mathbf{h}\mathbf{h}^*$, where $\mathbf{h}$ is a discrete Gaussian with the z-transform

$$h(z) = 2^{-2n}(1z^{-1} + 2z^0 + 1z^1)^n, \tag{43}$$

that is, the B-spline kernel of order $2n$. As n increases from 1 to 8, we increase $\sigma_{X,Y}$ and σ_Z from 4 to 10, and 28 to 36 respectively for our geodesic variant, and from 3 to 4, and 9 to 12 respectively for our bilateral one.

Table 3 provides the peak SNRs of the de-blurred images for different blur kernels. For comparison, the results for $L2$ (quadratic regularity), TV (total variation) [43] and Wiener-filtered solutions. All algorithm parameters used in different models are found using a grid search. The Wiener filter uses a uniform image power spectrum model. Separability of the blur kernels may be used to accelerate the iterative methods further (our times are for direct 2D deconvolution). Note the bilateral filter is not optimal for de-noising as pointed out by Buades *et al*. [9], who demonstrate the advantages of patch-based filtering (nonlocal means denoising) over pixel-based filtering (bilateral filter), so we can also choose the nonlocal means for $\mathbf{A}$. Figure 4 shows crops of our deblurred images.

8. Conclusion

In this paper, we solved regularized inverse problems via filtering. While such optimization problems are traditionally solved by finding a solution to a system of equations which expresses the optimality conditions, we showed that the act of solving such equations can actually be seen as a filtering operation, and reformulated the regularized inverse problem as a filtering task. We proceeded to solve a number of vision problems which are traditionally solved using iterations. We showed that the performance of our method is comparable to the methods specifically tailored and implemented for these applications. We hope that other vision researchers also find our approach useful for solving their own vision problems.

References

[1] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artif. Intell.*, 17(1):185–203, 1981.

[2] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Phys. Nonlinear Phenom.*, 60(1):259–268, 1992.

[3] Antonin Chambolle. An algorithm for total variation minimization and applications. *J. Math. Imaging Vis.*, 20(1–2):89–97, 2004.

[4] Antonin Chambolle and Thomas Pock. A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging. *J. Math. Imaging Vis.*, 40(1):120–145, 2011.

[5] Jonathan T. Barron and Ben Poole. The fast bilateral solver. In *ECCV*, 2016.

[6] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization Using Optimization. In *SIGGRAPH*, 2004.

[7] Jonathan T. Barron, Andrew Adams, YiChang Shih, and Carlos Hernandez. Fast bilateral-space stereo for synthetic defocus. In *CVPR*, 2015.

[8] Guy Gilboa and Stanley Osher. Nonlocal operators with applications to image processing. *Multiscale Model. Simul.*, 7(3):1005–1028, 2008.

[9] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. A non-local algorithm for image denoising. In *CVPR*, 2005.

[10] Manuel Werlberger, Thomas Pock, and Horst Bischof. Motion estimation with non-local total variation regularization. In *CVPR*, 2010.

[11] Philipp Krähenbühl and Vladlen Koltun. Efficient nonlocal regularization for optical flow. In *ECCV*, 2012.

[12] Antonin Chambolle and Pierre-Louis Lions. Image recovery via total variation minimization and related problems. *Numer. Math.*, 76(2):167–188, 1997.

[13] David L. Phillips. A technique for the numerical solution of certain integral equations of the first kind. *J ACM*, 9(1):84–97, 1962.

[14] Andrei Nikolaevich Tikhonov. On the solution of ill-posed problems and the method of regularization. In *Doklady Akademii Nauk*, 1963.

[15] Sean Twomey. On the Numerical Solution of Fredholm Integral Equations of the First Kind by the Inversion of the Linear System Produced by Quadrature. *J ACM*, 10(1):97–101, 1963.

[16] Sean Twomey. The application of numerical filtering to the solution of integral equations encountered in indirect sensing measurements. *J. Frankl. Inst.*, 279(2):95–109, 1965.

[17] Bobby R. Hunt. The application of constrained least squares estimation to image restoration by digital computer. *IEEE Trans. Comput.*, C–22(9):805–812, 1973.

[18] Alexander J. Smola and Risi Kondor. Kernels and regularization on graphs. In *Learning Theory and Kernel Machines*, Springer, 2003, pages 144–158.

[19] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *ICCV*, 1998.

[20] Volker Aurich and Jörg Weule. Non-Linear Gaussian Filters Performing Edge Preserving Diffusion. In *Mustererkennung 1995, 17. DAGM-Symposium*, 1995.

[21] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. Fast high-dimensional filtering using the permutohedral lattice. *Comput. Graph. Forum*, 29(2):753–762, 2010.

[22] Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Soc., 1997.

[23] Richard Sinkhorn. A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices. *Ann. Math. Stat.*, 35(2):876–879, 1964.

[24] Peyman Milanfar. Symmetrizing Smoothing Filters. *SIAM J. Imaging Sci.*, 6(1):263–284, 2013.

[25] Eduardo S. L. Gastal and Manuel M. Oliveira. Adaptive manifolds for real-time high-dimensional filtering. *ACM Trans Graph*, 31(4):33:1–33:13, 2012.

[26] Andrew Adams, Natasha Gelfand, Jennifer Dolson, and Marc Levoy. Gaussian KD-trees for fast high-dimensional filtering. In *SIGGRAPH*, 2009.

[27] Antonio Criminisi, Toby Sharp, Carsten Rother, and Patrick Pérez. Geodesic Image and Video Editing. *ACM Trans Graph*, 29(5):134:1–134:15, 2010.

[28] Eduardo S. L. Gastal and Manuel M. Oliveira. Domain transform for edge-aware image and video processing. In *SIGGRAPH*, 2011.

[29] Hans Knutsson and Carl-Fredrik Westin. Normalized and differential convolution. In *CVPR*, 1993.

[30] Jaesik Park, Hyeongwoo Kim, Yu-Wing Tai, Michael S. Brown, and In So Kweon. High quality depth map upsampling for 3D-TOF cameras. In *ICCV*, 2011.

[31] David Ferstl, Christian Reinbacher, Rene Ranftl, Matthias Ruether, and Horst Bischof. Image guided depth upsampling using anisotropic total generalized variation. In *ICCV*, 2013.

[32] Jiajun Lu and David Forsyth. Sparse depth super resolution. In *CVPR*, 2015.

[33] Nils Papenberg, Andrés Bruhn, Thomas Brox, Stephan Didas, and Joachim Weickert. Highly accurate optic flow computation with theoretically justified warping. *Int. J. Comput. Vis.*, 67(2):141–158, 2006.

[34] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012.

[35] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015.

[36] Qifeng Chen and Vladlen Koltun. Full flow: Optical flow estimation by global optimization over regular grids. In *CVPR*, 2016.

[37] Yinlin Hu, Rui Song, and Yunsong Li. Efficient coarse-to-fine patchmatch for large displacement optical flow. In *CVPR*, 2016.

[38] Christian Bailer, Bertram Taetz, and Didier Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *ICCV*, 2015.

[39] Moritz Menze, Christian Heipke, and Andreas Geiger. Discrete optimization for optical flow. In *GCPR*, 2015.

[40] Yu Li, Dongbo Min, Minh N. Do, and Jiangbo Lu. Fast guided global interpolation for depth and motion. In *ECCV*, 2016.

[41] Deqing Sun, Stefan Roth, and Michael J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *Int. J. Comput.*

Vis., 106(2):115–137, 2014.

[42] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. DeepFlow: Large Displacement Optical Flow with Deep Matching. In *ICCV*, 2013.

[43] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sci.*, 2(1):183–202, 2009.