Edge-preserving image denoising via group coordinate descent on the GPU

Madison G. McGaffin, *Student Member, IEEE,* and Jeffrey A. Fessler, *Fellow, IEEE*

***Abstract*—Image denoising is a fundamental operation in image processing, and its applications range from the direct (photographic enhancement) to the technical (as a subproblem in image reconstruction algorithms). In many applications, the number of pixels has continued to grow, while the serial execution speed of computational hardware has begun to stall. New image processing algorithms must exploit the power offered by massively parallel architectures like graphics processing units (GPUs). This paper describes a family of image denoising algorithms well-suited to the GPU. The algorithms iteratively perform a set of independent, parallel one-dimensional pixel-update subproblems. To match GPU memory limitations, they perform these pixel updates in-place and only store the noisy data, denoised image and problem parameters. The algorithms can handle a wide range of edge-preserving roughness penalties, including differentiable convex penalties and anisotropic total variation (TV). Both algorithms use the majorize-minimize (MM) framework to solve the one-dimensional pixel update subproblem. Results from a large 2D image denoising problem and a 3D medical imaging denoising problem demonstrate that the proposed algorithms converge rapidly in terms of both iteration and run-time.**

I. Introduction

Image acquisition systems produce measurements corrupted by noise. Removing that noise is called image denoising. Despite decades of research and remarkable successes, image denoising remains a vibrant field [6]. Over that time, image sizes have increased, the computational machinery available has grown in power and undergone significant architectural changes, and new algorithms have been developed for recovering useful information from noise-corrupted data.

Meanwhile, developments in image *reconstruction* have produced algorithms that rely on efficient denoising routines [17], [22]. The measurements in this setting are corrupted by noise and distorted by some physical process. Through variable splitting and alternating minimization techniques, the task of forming an image is decomposed into a series of smaller iterated subproblems. One successful family of algorithms separates "inverting" the physical system's behavior from denoising the image. Majorize-minimize algorithms like [1], [13] also involve denoising-like subproblems. These problems can be very high-dimensional: a routine chest X-ray computed tomography (CT) scan has the equivalent number of voxels as a 40 megapixel image and the reconstruction must account for 3D correlations between voxels.

The authors can be contacted at mcgaffin@umich.edu and fessler@umich.edu.

Supported in part by NIH grants R01 HL 098686 and U01 EB018753, and by equipment donations from Intel Corporation.

Growing problem sizes pose computational challenges for algorithm designers. Transistor densities continue to increase roughly with Moore's Law, but advances in modern hardware increasingly appear mostly in greater parallel-computing capabilities rather than single-threaded performance. Algorithm designers can no longer rely on developments in processor clock speed to ensure serial algorithms keep pace with increasing problem size. To provide acceptable performance for growing problem sizes, new algorithms should exploit highly parallel hardware architectures.

A poster-child for highly parallel hardware is the graphics processing unit (GPU). GPUs have always been specialized devices for performing many computations in parallel, but using GPU hardware for non-graphics tasks has in the past involved laboriously translating algorithms into "graphics terminology." Fortunately, in the past decade, programming platforms have developed around modern GPUs that enable algorithm designers to harness these massively parallel architectures using familiar C-like languages.

Despite these advances, designing algorithms for the GPU involves different considerations than designing for a conventional CPU. Algorithms for the CPU are often characterized by the number of floating point operations (FLOPs) they perform or the number of times they compute a cost function gradient. To accelerate convergence, algorithms may store extra information (*e.g.*, previous update directions or auxiliary/dual variables) or perform "global" operations (*e.g.*, line searches or inner products). These designs can accelerate an algorithm's per-iteration convergence or reduce the number of FLOPs required to achieve a desired level of accuracy, but their memory requirements do not map well onto the GPU.

An ideal GPU algorithm is composed of a series of entirely independent and parallel tasks performing the same operations on different data. The number of FLOPs can be less important than the parallelizability of those operations. Operations that are classically considered fast, like inner products and FFTs, can be relatively slow on the GPU due to memory accesses. Memory is also a far more scarce resource on the GPU. This makes successful, but memory-hungry, frameworks like the primal-dual algorithm [3] or variable splitting less suitable on the GPU. Fully exploiting GPU parallelism requires algorithms with local memory accesses and limited memory requirements.

This paper presents a pair of image denoising algorithms for the GPU. To exploit GPU parallelism, the algorithms use group coordinate descent (GCD) to decompose the image denoising problem into an iterated sequence of independent one-dimensional pixel-update subproblems. They avoid any additional memory requirements and are highly parallelizable.

Both algorithms solve these inner pixel-update subproblems using the well-known majorize-minimize framework [10], [11] and can handle a range of edge-preserving regularizers. Because of these properties, the proposed algorithms can efficiently solve large image denoising problems.

Section I-A introduces the image denoising framework and poses the two classes of problems our algorithms solve. Section II describes the shared GCD structure of our algorithms, and Section III describes how two specific algorithms solve the inner one-dimensional update problems. The experimental results from large-image denoising and X-ray CT reconstruction in Section IV illustrate the proposed algorithms' performance, and Section V contains some concluding remarks.

A. Optimization-based image denoising

Let $\mathbf{y} \in \mathbb{R}^N$ be noisy pixel measurements collected by an imaging system. In this paper, bold type indicates a vector quantity, and variables not in bold are scalars; the jth element of $\mathbf{y}$ is written y_j. Let w_j be some confidence we have in the jth measurement, *e.g.*, $w_j = \frac{1}{\sigma_j^2}$, the inverse of the variance of y_j. Let $\mathbf{x} \in \chi \subseteq \mathbb{R}^N$ be a candidate denoised image, and let R denote a regularizer on $\mathbf{x}$. The penalized weighted least squares (PWLS) estimate of the image given the noisy measurements $\mathbf{y}$ is the minimizer of the cost function $J(\mathbf{x})$:

$$J(\mathbf{x}) = \frac{1}{2}||\mathbf{x} - \mathbf{y}||_{\mathbf{W}}^2 + \mathsf{R}(\mathbf{x}), \tag{1}$$

$$\widehat{\mathbf{x}} = \underset{\mathbf{x} \in \chi}{\operatorname{argmin}} \; J(\mathbf{x}), \tag{2}$$

where $\mathbf{W} = \operatorname{diag}_j \{w_j\}$. The domain $\chi = \chi_1 \times \chi_2 \times \cdots \times \chi_N$, with χ_j convex, may codify a range of admissible pixel levels (*e.g.*, 0-255 for image denoising) or nonnegative values for *e.g.*, X-ray CT [26]. Similar to a prior distribution on $\mathbf{x}$, R is chosen to encourage expectations we have for the image. A simple and popular choice is the first-order edge-preserving regularizer:

$$\mathsf{R}(\mathbf{x}) = \beta \sum_{j=1}^{N} \sum_{l \in \mathcal{N}_j} \kappa_{jl} \psi(x_j - x_l). \tag{3}$$

This regularizer imposes a higher penalty on $\mathbf{x}$ as its "roughness" (measured as the differences between nearby pixels) increases. The global parameter β and local parameters $\kappa_{jl} \geq 0$ adjust the strength of the regularizer relative to the data-fit term [7]. The set $\mathcal{N}_j$ contains the neighbors of the jth pixel, as selected by the algorithm designer. The neighborhoods do not contain their centers: *i.e.*, $j \notin \mathcal{N}_j$. In 2D image denoising, using the four or eight nearest neighbors of the jth pixel are common choices; in 3D common choices are the six cardinal neighbors or the twenty-six adjacent voxels. This paper focuses on these first-order neighborhoods in 2D and 3D, but the presented algorithms can be extended to larger neighborhoods and higher dimensions.

The symmetric and convex potential function ψ adjusts qualitatively how adjacent pixel differences are penalized. Examples of ψ are:

- the quadratic function, $\psi_{\text{quad}}(t) = \frac{1}{2}t^2$;
- smooth nonquadratic regularizers, *e.g.*, the Fair potential $\psi_{\text{Fair}}(t;\delta) = \delta^2(|t/\delta| - \log(1 + |t/\delta|))$ [15]; and
- the absolute value function, $\psi_{\text{abs}}(t) = |t|$.

Potential functions that are relatively small around the origin (*e.g.*, ψ_{quad} and ψ_{Fair}) preserve small variations between neighboring pixels. The absolute value function is comparatively large around the origin, and can lead to denoised images with "cartoony" uniform regions [19]. On the other hand, potential functions that are relatively small away from the origin (*e.g.*, ψ_{abs} and ψ_{Fair}) penalize large differences (*i.e.*, edges) less than ψ_{quad}. Choosing one of these potential functions makes R an edge-preserving regularizer, and avoids over-smoothing edges in the denoised image $\widehat{\mathbf{x}}$, but it also makes the denoising problem (2) more difficult to solve.

Using ψ_{abs} in (3) yields the anisotropic TV regularizer [23].

II. Group coordinate descent

This section describes the "outer loop" of algorithms designed to solve (2) rapidly on the GPU. We use a superscript (n), *e.g.*, $\mathbf{x}^{(n)}$, to indicate the state of a variable in the nth iteration of the algorithm.

Consider optimizing $J(\mathbf{x})$ in (2) with respect to the jth pixel while holding the other pixels constant at $\mathbf{x} = \mathbf{x}^{(n)}$:

$$\underset{x_j \,:\, \mathbf{x} \in \chi}{\operatorname{argmin}} \; \frac{w_j}{2}(x_j - y_j)^2 + 2\beta \sum_{l \in \mathcal{N}_j} \kappa_{jl} \psi\Big(x_j - x_l^{(n)}\Big). \tag{4}$$

The only pixels involved in this optimization are the jth pixel and its neighbors, $\mathcal{N}_j$. Consequently, if the pixels in $\mathcal{N}_j$ are held constant, we can optimize over the jth pixel without any regard for the pixels outside $\mathcal{N}_j$.

Looping j through the pixels of $\mathbf{x}$, $j = 1, \ldots, N$, and performing the one-dimensional update (4) is called the coordinate descent algorithm [20]. This algorithm is convergent and monotone in cost function. However, because each optimization is performed serially, coordinate descent is ill-suited to modern highly parallel hardware like the GPU.

GCD algorithms instead optimize over a *group* of elements of $\mathbf{x}$ at a time while holding the others constant. The key to using GCD on a GPU efficiently is choosing appropriate groups that allow massive parallelism. Let $\mathcal{S}_1, \ldots, \mathcal{S}_M$ be a partition of the pixel coordinates of $\mathbf{x}$; we write $\mathbf{x} = [\mathbf{x}_{\mathcal{S}_1}, \ldots, \mathbf{x}_{\mathcal{S}_M}]$. A GCD algorithm that uses these groups to optimize (2) will loop over $m = 1, \ldots M$ and solve

$$\mathbf{x}_{\mathcal{S}_m}^{(n+1)} = \underset{\mathbf{x}_{\mathcal{S}_m} \,:\, \mathbf{x} \in \chi}{\operatorname{argmin}} \; J\Big(\mathbf{x}_{\mathcal{S}_1}^{(n+1)}, \ldots \mathbf{x}_{\mathcal{S}_{m-1}}^{(n+1)}, \mathbf{x}_{\mathcal{S}_m}, \mathbf{x}_{\mathcal{S}_{m+1}}^{(n)}, \ldots, \mathbf{x}_{\mathcal{S}_M}^{(n)}\Big). \tag{5}$$

The mth group update subproblem (5) is a $|\mathcal{S}_m|$-dimensional problem in general. However, we can design the groups such that each of these subproblems decomposes into $|\mathcal{S}_m|$ completely independent one-dimensional subproblems. If

$$\forall m, \forall j \in \mathcal{S}_m, \quad \mathcal{N}_j \cap \mathcal{S}_m = \emptyset, \tag{6}$$

then in each of the group update subproblems (5), the neighbors of all the pixels being optimized are held constant. By the Markov-like property observed above, this breaks the optimization over the pixels in $\mathcal{S}_m$ into $|\mathcal{S}_m|$ *independent* one-dimensional subproblems.

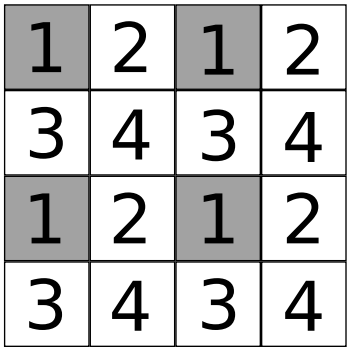


Fig. 1: Illustration of the groups in (6) for a 2D imaging problem with $\mathcal{N}_j$ containing the four or eight pixels adjacent to the jth pixel. Optimizing over the pixels in $\mathcal{S}_1$ (shaded) involves independent one-dimensional update problems for each pixel in the group.

for $n = 1$ up to N_{iter} **do**
  for $m = 1$ up to M **do**
    Parfor $j \in \mathcal{S}_m$
      Minimize $\Psi_j^{(n)}(x_j)$ *w.r.t.* $x_j \in \chi_j$.
    EndParfor
  end for
end for

Fig. 2: The GCD algorithm structure. The **Parfor** block contains $|\mathcal{S}_k|$ minimizations that are independent and implemented in parallel. Section III details these optimizations.

Figure 1 illustrates a set of groups that satisfies the "contains no neighbors" (6) requirement for a 2D problem and $\mathcal{N}_j$ containing the four or eight pixels adjacent to j. In 3D, both six-neighbor and twenty-six-neighbor $\mathcal{N}_j$ use eight groups arranged in a $2 \times 2 \times 2$ "checkerboard" pattern.

To summarize, we propose GCD algorithms for (2) that loop over the groups $m = 1, \dots, M$ and update the pixels in $\mathcal{S}_m$:

$$\mathbf{x}_{\mathcal{S}_m}^{(n+1)} = \underset{\mathbf{x}_{\mathcal{S}_m} : \mathbf{x} \in \chi}{\operatorname{argmin}} \sum_{j \in \mathcal{S}_m} \Psi_j^{(n)}(x_j), \qquad \text{where} \tag{7}$$

$$\Psi_j^{(n)}(x_j) = \frac{w_j}{2}(x_j - y_j)^2 + 2\beta \sum_{l \in \mathcal{N}_j} \kappa_{jl} \psi\Big(x_j - x_l^{(n)}\Big). \tag{8}$$

Each of the $\Psi_j^{(n)}$ are independent one-dimensional functions and are minimized in parallel. Because the pixel updates are performed in-place, this algorithm requires no additional memory beyond storing $\mathbf{x}$, $\mathbf{y}$, $\mathbf{W}$ and the regularizer weights. In many cases, $\mathbf{W}$ and the regularizer weights are uniform, and the algorithm must store only two image-sized vectors! These low memory requirements make the GCD algorithm remarkably well-suited to the GPU. This GCD algorithm is guaranteed to decrease the cost function J monotonically. Convergence to a minimizer of J is ensured under mild regularity conditions [11], [12]. Figure 2 summarizes the proposed algorithm structure.

III. One-dimensional subproblems

The complexity of solving each of the one-dimensional subproblems in (7) depends on the choice of potential function ψ. In this paper, we consider two cases:

- when ψ is convex and differentiable (Section III-A); and
- when ψ is the absolute value function, thus convex but not differentiable (Section III-B).

One could also adapt these methods to non-convex potential functions ψ, albeit with weaker convergence guarantees. In all cases, we approximately solve the one-dimensional subproblem (7) using the well-known majorize-minimize (MM) approach, also called optimization transfer and functional substitution [5], [8]. In iteration n, the MM framework generates a surrogate function $\Phi_j^{(n)}$ that may depend on $\mathbf{x}^{(n)}$ and satisfies the following "equality" and "lies-above" properties:

$$\Phi_j^{(n)}\Big(x_j^{(n)}\Big) = \Psi_j^{(n)}\Big(x_j^{(n)}\Big) \tag{9}$$

$$\Phi_j^{(n)}(x_j) \geq \Psi_j^{(n)}(x_j) \qquad \forall x_j \in \chi_j. \tag{10}$$

Majorize-minimize methods update x_j by minimizing $\Phi_j^{(n)}$,

$$x_j^{(n+1)} = \underset{x_j \in \chi_j}{\operatorname{argmin}} \ \Phi_j^{(n)}(x_j). \tag{11}$$

Because χ_j is convex, we find the unconstrained solution to (11) then project it onto χ_j. This update is guaranteed to decrease both the 1D cost function $\Psi_j^{(n)}(x_j)$ and the global cost function J. Even though we are minimizing the surrogate instead of the single-pixel cost function $\Psi_j^{(n)}(x_j)$, the GCD-MM algorithm is convergent [11].

To implement the MM iteration (11), we need to efficiently construct and minimize the surrogate $\Phi_j^{(n)}$. The one-dimensional cost function $\Psi_j^{(n)}$ is the sum of a quadratic term and $|\mathcal{N}_j|$ often nonquadratically penalized differences (the $\psi\Big(x_j - x_l^{(n)}\Big)$ terms). Figure 3 illustrates an example $\Psi_j^{(n)}$ using only three neighbors and the absolute value potential function. The next two subsections describe how we construct a surrogate $\phi_{jl}^{(n)}$ for each of the nonquadratic terms in $\Psi_j^{(n)}$. Replacing each $\psi\Big(x_j - x_l^{(n)}\Big)$ in (8) with its surrogate $\phi_{jl}^{(n)}(x_j)$ gives us the following majorizer for $\Psi_j^{(n)}$ in (11):

$$\Phi_j^{(n)}(x_j) = \frac{w_j}{2}(x_j - y_j)^2 + 2\beta \sum_{l \in \mathcal{N}_j} \kappa_{jl} \phi_{jl}^{(n)}(x_j). \tag{12}$$

Constructing and minimizing (12) requires only a few registers and a small number of visits to each pixel in $\mathcal{N}_j$. This keeps the number of memory accesses low and the acccess pattern regular, which is necessary for good GPU performance.

A. Convex and differentiable potential function

First we consider the simpler case of a convex and differentiable cost function. Define the Huber curvature $\omega_{jl}^{(n)}$ as

$$\omega_{jl}^{(n)} = \frac{\psi'\Big(x_j^{(n)} - x_l^{(n)}\Big)}{x_j^{(n)} - x_l^{(n)}}. \tag{13}$$

If $\omega_{jl}^{(n)}$ is bounded and nonincreasing as $\left|x_j^{(n)} - x_l^{(n)}\right| \to \infty$, then the following quadratic surrogate majorizes $\psi\Big(x_j - x_{jl}^{(n)}\Big)$ at $x_j^{(n)}$ and has optimal (*i.e.*, minimal) curvature [9, page 185]:

$$\phi_{jl}^{(n)}(x_j) = \psi\Big(x_j^{(n)}\Big) + \Big(x_j - x_l^{(n)}\Big)\psi'\Big(x_j^{(n)} - x_l^{(n)}\Big) + \frac{\omega_{jl}^{(n)}}{2}\Big(x_j - x_l^{(n)}\Big)^2. \quad (14)$$

Many potential functions have bounded and monotone nonincreasing Huber curvatures, including the Fair potential [15] and the q-Generalized Gaussian potential function sometimes used in X-ray CT reconstruction [26]. Because the Huber curvature is optimally small, the closed-form MM update,

$$x_j^{(n+1)} = x_j^{(n)} - \frac{w_j\Big(x_j^{(n)} - y\Big) + 2\beta\sum_{l\in\mathcal{N}_j}\kappa_{jl}\psi'\Big(x_j^{(n)} - x_l^{(n)}\Big)}{w_j + 2\beta\sum_{l\in\mathcal{N}_j}\kappa_{jl}\omega_{jl}^{(n)}}, \quad (15)$$

takes the largest step possible for a quadratic majorizer of the form (12). To implement (15) efficiently, we use (13) to replace the ψ' terms with the product of $\omega_{jl}^{(n)}$ and $x_j^{(n)} - x_l^{(n)}$. The resulting algorithm is implemented with only one potential function derivative per neighboring pixel.

B. The absolute value potential function

The quadratic majorizer in (14) applies to a class of differentiable potential functions. TV uses the absolute value potential function, and ψ_{abs} is not differentiable at the origin. In the previous section's terminology, the curvature $\omega_{jl}^{(n)}$ "explodes" if $x_j^{(n)} \approx x_l^{(n)}$. TV denoising encourages neighboring pixels to be identical to one another so this is a significant concern. Even if $x_j^{(n)} \neq x_l^{(n)}$ in practice [21], the exploding surrogate curvature may cause numerical problems.

A way to avoid this problem is to modify the curvatures to prevent the $\omega_{jl}^{(n)}$ from exploding. One approach is to replace ψ_{abs} with the hyperbola potential function, $\psi(t) = \sqrt{\epsilon + t^2} - \sqrt{\epsilon}$, with $\epsilon > 0$ small, or similar "corner-rounded" absolute-value-like function. While this makes the techniques in the previous section directly applicable, it changes the global cost function J, which may be suboptimal.

Another corner rounding approach is to "cap" the curvatures at ϵ^{-1} for small $\epsilon > 0$:

$$\omega_{jl,\epsilon}^{(n)} = \frac{1}{\max\left\{\epsilon, \left|x_j^{(n)} - x_l^{(n)}\right|\right\}}. \quad (16)$$

Unfortunately, the quadratic function with curvature $\omega_{jl,\epsilon}^{(n)}$ does not satisfy the "lies above" surrogate requirement (10) when $\left|x_j^{(n)} - x_l^{(n)}\right| < \epsilon$. Because $\Phi_j^{(n)}$ would not then be a "proper" surrogate for $\Psi_j^{(n)}$, a GCD algorithm based on (16) may not monotonically decrease the cost function J. Empirically, we found that using $\omega_{jl,\epsilon}^{(n)}$ appears to cause $\mathbf{x}^{(n)}$ to enter a suboptimal limit cycle around the optimum. Thus we developed the following duality approach.

1) Duality approach: One way to handle the absolute value function is to use its dual formulation [3], [4], [16], [27]. We write the absolute value function implicitly in terms of a maximization over a dual variable $\gamma_{jl}^{(n)}$:

$$\left|x_j - x_l^{(n)}\right| = \max_{\gamma_{jl}^{(n)}\in[-1,1]} \gamma_{jl}^{(n)}\Big(x_j - x_l^{(n)}\Big). \quad (17)$$

Thus, by choosing any closed interval $\Omega^{(n)} \supseteq [-1,1]$, the following is a surrogate for $\left|x_j - x_l^{(n)}\right|$ that satisfies both the "equality" (9) and "lies above" (10) majorizer properties:

$$\phi_{jl,\Omega^{(n)}}^{(n)}(x_j) = \max_{\gamma_{jl}^{(n)}\in\Omega^{(n)}} \gamma_{jl}^{(n)}\Big(x_j - x_l^{(n)}\Big)\kappa_{jl} - \frac{\delta_{jl}^{(n)}}{2}\left(\Big(\gamma_{jl}^{(n)}\Big)^2 - 1\right), \quad (18)$$

where $\delta_{jl}^{(n)} = \left|x_j^{(n)} - x_l^{(n)}\right|\kappa_{jl}$. When $\Omega^{(n)} = [-1,1]$, $\phi_{jl,\Omega^{(n)}}^{(n)} = \psi_{jl}^{(n)}$. Selecting $\Omega^{(n)}$ larger than $[-1,1]$ increases the domain of maximization in (18) and loosens the majorization, and satisfies the "equality" (9) and "lies above" (10) majorization conditions. Figure 4 illustrates $\phi_{jl,\Omega^{(n)}}^{(n)}$ for several choices of $\Omega^{(n)}$.

Let $D = |\mathcal{N}_j|$ be the number of neighbors of the jth pixel. Denote the vector of dual variables $\boldsymbol{\gamma}_j = \left[\gamma_{j1}^{(n)}, \ldots, \gamma_{jD}^{(n)}\right]$ and their domain $\boldsymbol{\Omega}^{(n)} = \Omega^{(n)} \times \cdots \times \Omega^{(n)}$. We plug $\phi_{jl,\Omega^{(n)}}^{(n)}$ into (12) to construct the surrogate function $\Phi_j^{(n)}$:

$$\Phi_j^{(n)}(x_j) = \operatorname*{argmax}_{\boldsymbol{\gamma}_j\in\boldsymbol{\Omega}^{(n)}} \mathcal{L}_j^{(n)}(x_j, \boldsymbol{\gamma}_j), \qquad \text{where} \quad (19)$$

$$\mathcal{L}_j^{(n)}(x_j, \boldsymbol{\gamma}_j) = \frac{w_j}{2}(x_j - y_j)^2 + 2\beta\sum_{l\in\mathcal{N}_j}\kappa_{jl}\gamma_{jl}^{(n)}\Big(x_j - x_l^{(n)}\Big) - \frac{\delta_{jl}^{(n)}}{2}\left(\Big(\gamma_{jl}^{(n)}\Big)^2 - 1\right) \quad (20)$$

Figure 3 illustrates $\Psi_j^{(n)}$ and $\Phi_j^{(n)}$ for two values of $x_j^{(n)}$. Note that, unlike the "corner-rounding" approximations, $\Phi_j^{(n)}$ faithfully preserves the nondifferentiable "corner" of $\Psi_j^{(n)}$ at the minimizer, $x_j^{(n)} = 0.1$.

To implement the majorize-minimize procedure (11) by minimizing (20), we pass into the dual domain. Observe that $\mathcal{L}_j^{(n)}$ is convex and continuous in x_j and concave and continuous in the $\gamma_{jl}^{(n)}$, and the set $\boldsymbol{\Omega}^{(n)}$ is compact. We invoke Sion's minimax theorem [24] to transpose the order of minimization and maximization:

$$\operatorname*{argmin}_{x_j} \operatorname*{argmax}_{\boldsymbol{\gamma}_j\in\boldsymbol{\Omega}^{(n)}} \mathcal{L}_j^{(n)}(x_j, \boldsymbol{\gamma}_j) = \operatorname*{argmax}_{\boldsymbol{\gamma}_j\in\boldsymbol{\Omega}^{(n)}} \operatorname*{argmin}_{x_j} \mathcal{L}_j^{(n)}(x_j, \boldsymbol{\gamma}_j). \quad (21)$$

The inner minimization over x_j can now be solved trivially in terms of $\boldsymbol{\gamma}_j^{(n)}$:

$$x_j(\boldsymbol{\gamma}_j) = y_j - \frac{\beta}{w}\sum_{l\in\mathcal{N}_j}\gamma_{jl}^{(n)}\kappa_{jl}. \quad (22)$$

Plugging (22) into (20) and maximizing over $\boldsymbol{\gamma}_j \in \boldsymbol{\Omega}^{(n)}$, we arrive at the following quadratic dual problem:

$$\boldsymbol{\gamma}_j^* \in \underset{\boldsymbol{\gamma}_j \in \boldsymbol{\Omega}^{(n)}}{\operatorname{argmax}} \; D^{(n)}(\boldsymbol{\gamma}_j), \qquad \text{where} \tag{23}$$

$$D^{(n)}(\boldsymbol{\gamma}_j) = -\frac{1}{2}\boldsymbol{\gamma}_j{}'\left(\mathbf{D} + \frac{1}{w}\boldsymbol{\beta}\boldsymbol{\beta}'\right)\boldsymbol{\gamma}_j + \boldsymbol{\gamma}_j{}'\boldsymbol{\Lambda}\boldsymbol{\beta}, \tag{24}$$

where $\mathbf{D} = \operatorname{diag}_l\left\{2\beta\delta_{jl}^{(n)}\right\}$, $\boldsymbol{\Lambda} = \operatorname{diag}_l\left\{y_j - x_l^{(n)}\right\}$, and $\boldsymbol{\beta} = \operatorname{vec}_l\{2\beta\kappa_{jl}\}$. Because expanding $\Omega^{(n)}$ only "loosens" the majorization $\phi_{jl,\Omega^{(n)}}^{(n)}$ we simply define $\boldsymbol{\Omega}^{(n)}$ to include the pseudoinverse

$$\boldsymbol{\gamma}_j^+ = \left(\mathbf{D} + \frac{1}{w}\boldsymbol{\beta}\boldsymbol{\beta}'\right)^+ \boldsymbol{\Lambda}\boldsymbol{\beta}, \tag{25}$$

and then solve (23) by finding the pseudoinverse. In practice, this means we can solve the dual problem (23) as if it were unconstrained.

2) Solving the dual problem: The dual problem (23) has a diagonal-plus-rank-1 Hessian that can be trivially inverted when the diagonal matrix $\mathbf{D}$ is full rank. However, when at least one entry of $\mathbf{D}$ is small (*i.e.*, when $x_j^{(n)} \approx x_l^{(n)}$ for some l), the problem becomes ill-conditioned and requires an iterative method or an expensive "direct method" (*e.g.*, computing the eigenvalue decomposition of $\mathbf{D} + \frac{1}{w}\boldsymbol{\beta}\boldsymbol{\beta}'$ or the "matrix pseudoinverse lemma" [14]). We propose an iterative *minor*ize-*max*imize procedure that exploits the diagonal-plus-rank-1 Hessian.

This inner minorize-maximize procedure is iterative, so we denote the subiteration number with a superscripted m. The following function, $S_j^{(m)}(\boldsymbol{\gamma}_j)$ is a minorizer for $D_j^{(n)}(\boldsymbol{\gamma}_j)$ at $\boldsymbol{\gamma}_j^{(m)}$ in the sense that it satisfies the "equality" property (9) at $\boldsymbol{\gamma}_j^{(m)}$ and a "lies-below" property analogous to the "lies above" majorization property (10):

$$\begin{aligned} S_j^{(m)}(\boldsymbol{\gamma}_j) = D_j^{(n)}\left(\boldsymbol{\gamma}_j^{(m)}\right) + \left(\boldsymbol{\gamma}_j - \boldsymbol{\gamma}_j^{(m)}\right)'\nabla D_j^{(n)}\left(\boldsymbol{\gamma}_j^{(m)}\right) \\ - \frac{1}{2}\left(\boldsymbol{\gamma}_j - \boldsymbol{\gamma}_j^{(m)}\right)'\left(\mathbf{D}_\epsilon + \frac{1}{w}\boldsymbol{\beta}\boldsymbol{\beta}'\right)\left(\boldsymbol{\gamma}_j - \boldsymbol{\gamma}_j^{(m)}\right), \end{aligned} \tag{26}$$

where $\mathbf{D}_\epsilon = \operatorname{diag}_l\{\max\{\epsilon, \mathbf{D}_{ll}\}\}$. Let $\mathbf{H}_\epsilon = \mathbf{D}_\epsilon + \frac{1}{w}\boldsymbol{\beta}\boldsymbol{\beta}'$. Substituting the "min" for a "max" in the MM procedure (11) leads to the following iterative procedure for solving (23):

$$\boldsymbol{\gamma}_j^{(m+1)} = \underset{\boldsymbol{\gamma}_j}{\operatorname{argmax}} \; S_j^{(m)}(\boldsymbol{\gamma}_j) \tag{27}$$

$$= \mathbf{H}_\epsilon^{-1}\left(\boldsymbol{\Lambda}\boldsymbol{\beta} - \mathbf{M}_\epsilon\boldsymbol{\gamma}_j^{(m)}\right), \tag{28}$$

where $\mathbf{M}_\epsilon = \operatorname{diag}_l\left\{\max\left\{0, \epsilon - \delta_{jl}^{(n)}\right\}\right\}$. We multiply by $\mathbf{H}_\epsilon^{-1}$ efficiently using the matrix inversion lemma.

The recursion (28) reveals an interesting quality of the minorize-maximize procedure. When all the neighbors $x_l^{(n)}$ are sufficiently different from $x_j^{(n)}$, $\mathbf{M}_\epsilon$ is the zero-matrix and the MM recursion (28) is stationary. In other words, $\boldsymbol{\gamma}_j^{(m)}$ converges in a single iteration. This corresponds to the case where the heuristic "capped-curvature" majorize-minimize algorithm produces a valid surrogate. On the other hand, when some $\delta_{jl}^{(n)} \approx 0$, the "capped-curvature" algorithm may produce an invalid majorizer, but the recursion (28) will eventually produce (by finding appropriate values for the corresponding $\boldsymbol{\gamma}_{jl}^{(n)}$) and minimize a valid majorizer for $\Psi_j^{(n)}$. A practical alternative to running an arbitrarily large number of inner minorize-majorize iterations is to track the cost function value $\Psi_j^{(n)}\left(x_j\left(\boldsymbol{\gamma}_j^{(m)}\right)\right)$ and terminate the minorize-maximize algorithm when

$$\Psi_j^{(n)}\left(x_j\left(\boldsymbol{\gamma}_j^{(m)}\right)\right) \le \Psi_j^{(n)}\left(x_j^{(n)}\right). \tag{29}$$

This check was inexpensive to integrate into the minorize-maximize iteration, so we used it in the experiments below. Nonetheless, it is possible that in late iterations, as $x_j^{(n)} \approx x_l^{(n)}$, the domain $\boldsymbol{\Omega}^{(n)}$ grows and the majorizer $\Phi_j^{(n)}$ becomes increasingly loose. This would slow the convergence of $\mathbf{x}^{(n)} \to \widehat{\mathbf{x}}$.

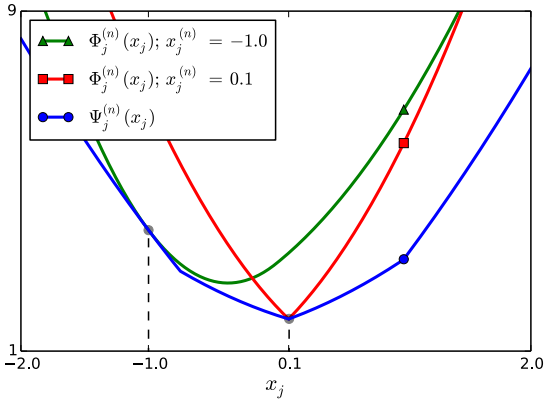


Fig. 3: An example of the pixel-update cost function $\Psi_j^{(n)}$ with three neighbors and the absolute value potential function. The majorizer $\Phi_j^{(n)}$ described in Section III-B1 is drawn at two points: the suboptimal point $x_j^{(n)} = -1.0$ and the optimum $x_j^{(n)} = 0.1$. In both cases, $\Omega = [-3, 3]$.

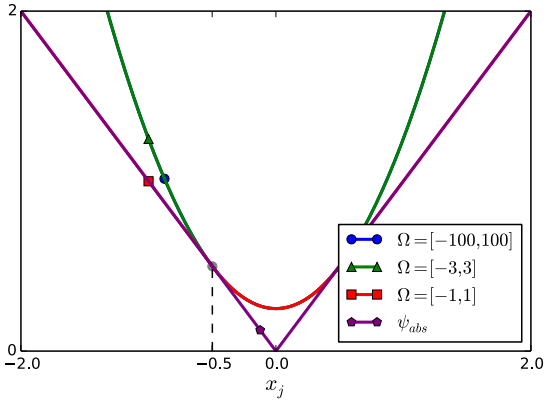


Fig. 4: The absolute value potential function and the majorizer $\phi_\Omega^{(n)}(x_j)$ described in Section III-B1 with $x_j^{(n)} = -0.5$. Enlarging the domain Ω "loosens" the majorizer.

IV. Experiments

This section presents two experiments using the TV regularizer (Section IV-A) and a differentiable edge-preserving regularizer used in CT reconstruction (Section IV-B). All the algorithms in the following experiments were run on an NVIDIA Tesla C2050 GPU with 3 GB of memory and implemented in OpenCL.

In addition to the algorithms described above, we applied Nesterov's first-order acceleration [18] to the GCD algorithm after each loop through all the groups. Future research may establish the theoretical convergence properties of these accelerated algorithms, and they appear to be stable.

A. Anisotropic TV denoising

In 2004, the Mars Opportunity rover transmitted photographs of its landing site in the "Eagle Crater" back to Earth. Scientists at NASA/JPL combined these photographs into a 22,780 × 3,301-pixel (approximately 75 megapixel) grayscale image [2]. Pixels were represented by floating-point numbers between 0 and 255; storing each copy of the image required approximately 300 MB of memory.

We corrupted the composite image with additive white Gaussian noise with standard deviation $\sigma = 20$ gray levels (see Figure 5a). Then we denoised the corrupted image by solving the iterative denoising problem (2) with anisotropic total variation ($\psi = \psi_{\text{abs}}$) using all eight adjacent pixels ($|\mathcal{N}_j| = 8$), empirically selected regularizer weight $\beta = 7$, uniform weights ($\mathbf{W} = \mathbf{I}$, $\kappa_{jl} = 1$), and the constraint $x_j \in [0, 255]$. Figure 5b shows an effectively converged reference image, $\mathbf{x}^*$. All the algorithms in this section are initialized from the noisy data, $\mathbf{x}^{(0)} = \mathbf{y}$.

We ran the Chambolle-Pock primal-dual algorithm (CP-PDA) (Algorithm 2 in [3], adapted to anisotropic TV), the separable quadratic surrogates [1] (SQS-ϵ) algorithm with the "capped-curvature" corner-rounding approximation and the proposed GCD algorithm with the same corner-rounding approximation (GCD-ϵ). We also applied Nesterov's first-order acceleration to SQS (SQS-ϵ-N) and corner-rounded GCD (GCD-ϵ-N). Finally, we ran GCD with two inner iterations of the proposed duality-based majorizer and Nesterov's first-order acceleration (GCD(2)-N). In all cases, we chose $\epsilon = 2$. Figure 6 plots cost function and root mean-square difference (RMSD) to the reference image against algorithm iteration and time.

The Chambolle-Pock primal-dual algorithm converged rapidly in terms of iteration, but considerably more slowly as a function of time. This behavior, which is hidden when experiments are performed with small images, is a consequence of PDA's high memory requirements. Even on the NVIDIA Tesla with 3GB of memory, we could not store all the algorithm's variables (including the regularizer and data-fit weights) on the GPU at once. Consequently we needed to occasionally transfer memory between RAM and the GPU, which slowed down PDA's convergence speed with respect to time. Because the PDA uses $|\mathcal{N}_j|$ image-sized dual variables, this memory burden would be even greater for a 3D denoising problem. At least with modern GPU hardware, algorithms with lower memory requirements like SQS-ϵ and the GCD algorithms seem more appropriate than PDA for large problems.

The SQS algorithm can be viewed as a one-group GCD algorithm, where surrogate functions are used to decouple the image update into a set of one-dimensional updates. In that light, the major differences between the SQS and GCD algorithms are pixel update order and majorizer looseness, and both of these differences appear to be advantages for GCD.

Although both the SQS-ϵ and GCD-ϵ algorithms in this experiment perform a corner-rounding approximation, GCD-ϵ's pixel update order appears to make it more robust to the error introduced by that approximation. This can be seen in the more accurate limit cycles reached by the GCD-ϵ algorithms compared to the respective SQS-ϵ algorithms. The GCD algorithms also do not need to majorize to produce one-dimensional subproblems; this makes GCD-ϵ's one-dimensional surrogate $\Phi_j^{(n)}$ "tighter" than the corresponding one-dimensional surrogate produced by SQS. This increases the step sizes that the GCDs algorithm take, as seen by GCD-ϵ reaching its limit cycle more rapidly than SQS-ϵ.

Unlike the SQS algorithms, the proposed GCD algorithm can achieve more accurate solutions by performing more iterations of the inner MM algorithm. This allows GCD(2)-N to rapidly achieve a more accurate solution than the corner-rounding algorithms.

1) Late-iteration behavior and multiple MM steps: To further explore the effect of the number of inner MM iterations on algorithm convergence, we also initialized GCD with

$$\mathbf{x}^{(0)} = \mathbf{x}^* + \mathbf{w}, \qquad \mathbf{w} \sim \mathrm{N}(\mathbf{0}, \mathbf{I}), \tag{30}$$

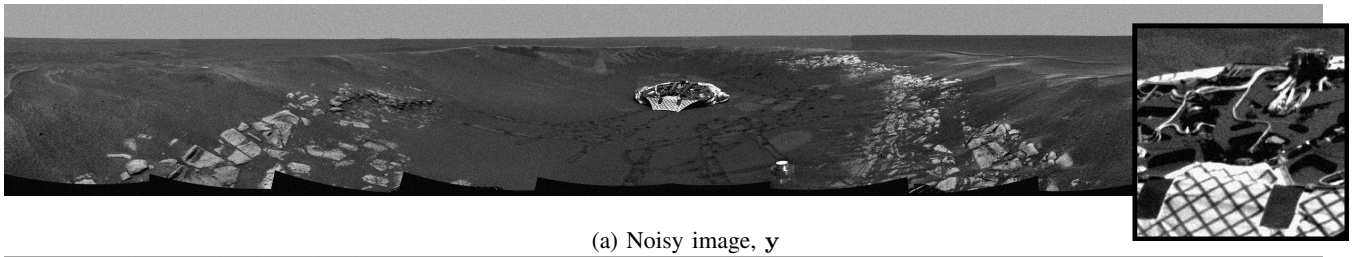
a point near the reference image. We ran GCD with up to 1, 2, 4 and 8 inner MM iterations. Each algorithm was terminated early if possible using the monotone-cost stopping criteria (29). Figure 6c plots RMSD to $\mathbf{x}^*$ against time for each configuration.

This experiment reveals two important things. First, unsurprisingly, increasing the maximum number of inner MM iterations allows the GCD algorithms to converge to a solution closer to $\mathbf{x}^*$. In all cases, the GCD algorithms produced a more accurate solution than SQS-ϵ, including GCD-ϵ, which "corner-rounds" in a similar way. Second, while more inner iterations requires more time per outer iteration, algorithms with more inner iterations may converge more quickly in time than those with fewer. The markers in Figure 6c were all placed at the 12th iteration. Although GCD(4) took nearly half as long per iteration as GCD(8), the eight-inner-iterations algorithm converged roughly as quickly in time and to a more accurate limit cycle.

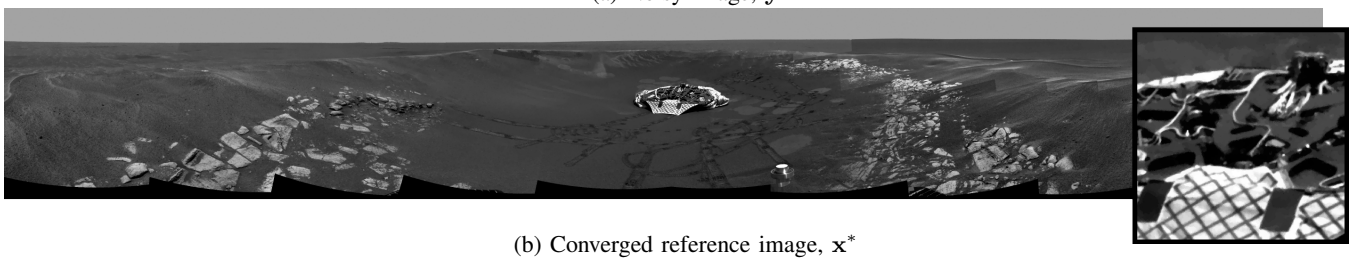
B. X-ray CT denoising

In diagnostic X-ray CT reconstruction, differentiable convex potential functions are often preferred to the absolute value potential function [26]. One choice of potential function is the q-generalized Gaussian (qGG),

$$\psi(t) = \frac{\frac{1}{2}|t|^p}{1 + |t/\delta|^{p-q}}. \tag{31}$$

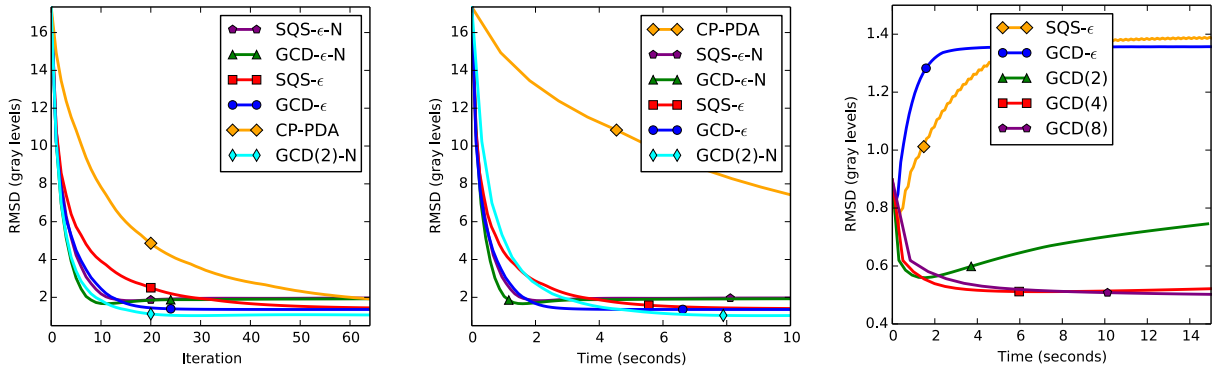


(a) Noisy image, $\mathbf{y}$



(b) Converged reference image, $\mathbf{x}^*$

Fig. 5: Initial noisy and converged reference images from the TV denoising experiment in Section IV-A. The original image is an approximately 75-megapixel composite of pictures taken by NASA's Mars Opportunity Rover; the insets are 512×512-pixel subimages.



(a) RMSD to $\mathbf{x}^*$ by iteration (b) RMSD to $\mathbf{x}^*$ by time (c) RMSD to $\mathbf{x}^*$ by time ($\mathbf{x}^{(0)}$ near $\mathbf{x}^*$)

Fig. 6: Root-mean-squared-difference to the converged reference image $\mathbf{x}^*$ by iteration and time for the total variation denoising experiment in Section IV-A.

The qGG potential function is both convex and differentiable for appropriate choice of p, q and $\delta > 0$.

While CT reconstruction involves solving a more general regularized least-squares problem, variable splitting and alternating minimization methods can produce algorithms that handle the system physics and edge-preserving regularizer in separate subproblems. In some memory-conservative variable splitting approaches [17] or majorize-minimize algorithms using separable quadratic surrogates [1], [13], the regularizer appears in a denoising problem like (2).

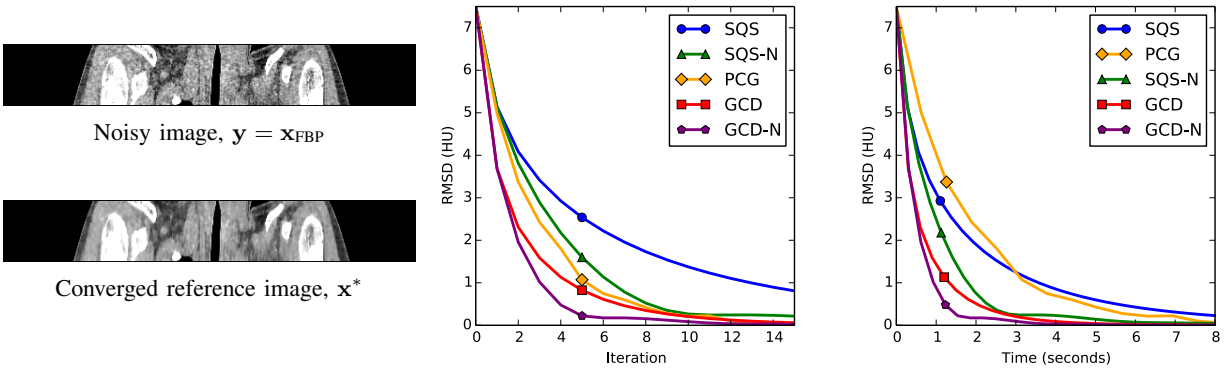
In this experiment we solved a denoising problem that could arise from a variable splitting X-ray CT reconstruction algorithm. The data came from a $512 \times 512 \times 65$-pixel helical shoulder image provided by GE Healthcare. Pixels were represented between 0 and 2,600 modified Hounsfield units (HU). We used the qGG potential function (with $q = 2$, $p = 1.2$ and $\delta = 10$ HU) and nonuniform regularizer weights typical of helical CT reconstruction [25]. The regularizer penalized all adjacent 3D neigbhors, *i.e.*, $|\mathcal{N}_j| = 26$. We set the diagonal weight matrix $\mathbf{W}$ to

$$\mathbf{W} = \operatorname*{diag}_j \left\{ \frac{[\mathbf{A}'\mathbf{S}\mathbf{A}]_{jj}}{2} \right\}, \tag{32}$$

where $\mathbf{A}$ is the so-called CT system matrix and $\mathbf{S}$ contains the statistical weights of the measurements [26].

We initialized each algorithm with $\mathbf{x}^{(0)} = \mathbf{x}_{\text{FBP}}$, the output of the classical analytical filtered backprojection (FBP) algorithm. To include second-order methods like preconditioned conjugate gradients in our comparison, we dropped the conventional nonnegativity constraint used in X-ray CT. Figure 7a illustrates the center slice of $\mathbf{x}_{\text{FBP}}$ and an effectively converged reference image, $\mathbf{x}^*$.

We solved the denoising problem with the proposed GCD algorithm, the separable quadratic surrogate algorithm (SQS), and preconditioned conjugate gradients (PCG) using a diagonal preconditioner. We also ran GCD and SQS with Nesterov's first-order acceleration (GCD-N and SQS-N). Figures 7b



(a) Center slice of intial and converged images

(b) RMSD to $\mathbf{x}^*$ by iteration

(c) RMSD to $\mathbf{x}^*$ by time

Fig. 7: Results from the X-ray CT denoising problem. Figure 7a displays the center slices of the initial noisy filtered backprojection image and the converged reference. Both are displayed on a 800 - 1200 modified Hounsfield unit (HU) scale.

and 7c plot the progress of each algorithm towards $\mathbf{x}^*$ as a function of iteration and time, respectively.

Preconditioned conjugate gradients converged quickly per iteration but comparably to SQS by time. The high computational cost of PCG on the GPU is caused by the algorithm's inner products and multiple inner steps; the diagonal preconditioner added negligible computational cost. Inner products are classically considered to be computationally cheap operations, but on the GPU and for this family of denoising problems, they are a considerable computational burden. The algorithms that perform only local memory accesses (SQS and GCD) and their accelerated variants converged significantly more quickly by wall time. Of these, GCD and GCD-N converged the fastest.

V. Conclusions

The trend in modern computing hardware is towards increased parallelism instead of better serial performance. This paper presented image denoising algorithms for edge-preserving regularization that play to the strengths of GPUs, the exemplar of this parallelism trend. By avoiding operations like inner products or complex preconditioners and minimizing memory usage, the proposed GCD algorithms provide impressive convergence rates. The additional increase in performance provided by Nesterov's first-order acceleration is exciting, and further work is needed to characterize the theoretical behavior of the accelerated algorithms. This paper focuses on gray scale images, but the general approach is extensible to color images and video.

References

[1] H. Erdoğan and J. A. Fessler. Ordered subsets algorithms for transmission tomography. *Phys. Med. Biol.*, 44(11):2835–51, November 1999.

[2] NASA Jet Propulsion Laboratory / Caltech. PIA05600: Eyeing "Eagle Crater", 2004.

[3] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *J. Math. Im. Vision*, 40(1):120–145, 2011.

[4] T. F. Chan, G. H. Golub, and P. Mulet. A nonlinear primal-dual method for total variation-based image restoration. *SIAM J. Sci. Comp.*, 20(6):1964–77, 1999.

[5] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Trans. Im. Proc.*, 6(2):298–311, February 1997.

[6] P. Chatterjee and P. Milanfar. Is denoising dead? *IEEE Trans. Im. Proc.*, 19(4):985–911, April 2010.

[7] J. A. Fessler and W. L. Rogers. Spatial resolution properties of penalized-likelihood image reconstruction methods: Space-invariant tomographs. *IEEE Trans. Im. Proc.*, 5(9):1346–58, September 1996.

[8] D. Geman and G. Reynolds. Constrained restoration and the recovery of discontinuities. *IEEE Trans. Patt. Anal. Mach. Int.*, 14(3):367–83, March 1992.

[9] P. J. Huber. *Robust statistics*. Wiley, New York, 1981.

[10] D. R. Hunter and K. Lange. A tutorial on MM algorithms. *American Statistician*, 58(1):30–7, February 2004.

[11] M. W. Jacobson and J. A. Fessler. An expanded theoretical treatment of iteration-dependent majorize-minimize algorithms. *IEEE Trans. Im. Proc.*, 16(10):2411–22, October 2007.

[12] S. T. Jensen, S. Johansen, and S. L. Lauritzen. Globally convergent algorithms for maximizing a likelihood function. *Biometrika*, 78(4):867–77, December 1991.

[13] D. Kim, D. Pal, J-B. Thibault, and J. A. Fessler. Accelerating ordered subsets image reconstruction for X-ray CT using spatially non-uniform optimization transfer. *IEEE Trans. Med. Imag.*, 32(11):1965–78, November 2013.

[14] K. Kohno, M. Kawamoto, and Y. Inouye. A matrix pseudoinversion lemma and its application to block-based adaptive blind deconvolution for MIMO systems. *IEEE Trans. Circ. Sys. I, Fundamental theory and applications*, 57(7):1499–1512, July 2010.

[15] K. Lange. Convergence of EM image reconstruction algorithms with Gibbs smoothing. *IEEE Trans. Med. Imag.*, 9(4):439–46, December 1990. Corrections, T-MI, 10:2(288), June 1991.

[16] M. G. McGaffin and J. A. Fessler. Fast edge-preserving image denoising via group coordinate descent on the GPU. In *Proc. SPIE 9020 Computational Imaging XII*, page 90200P, 2014.

[17] M. G. McGaffin, S. Ramani, and J. A. Fessler. Reduced memory augmented Lagrangian algorithm for 3D iterative X-ray CT image reconstruction. In *Proc. SPIE 8313 Medical Imaging 2012: Phys. Med. Im.*, page 831327, 2012.

[18] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Math. Dokl.*, 27(2):372–76, 1983.

[19] M. Nikolova, M. K. Ng, and C-P. Tam. Fast nonconvex nonsmooth minimization methods for image restoration and reconstruction. *IEEE Trans. Im. Proc.*, 19(12):3073–88, December 2010.

[20] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, New York, 1999.

[21] J. P. Oliveira, J. M. Bioucas-Dias, and M. A. T. Figueiredo. Adaptive total variation image deblurring: A majorization-minimization approach. *Signal Processing*, 89(9):1683–93, September 2009.

[22] S. Ramani and J. A. Fessler. A splitting-based iterative algorithm for

accelerated statistical X-ray CT reconstruction. *IEEE Trans. Med. Imag.*, 31(3):677–88, March 2012.

[23] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithm. *Physica D*, 60(1-4):259–68, November 1992.

[24] M. Sion. On general minimax theorems. *Pacific J. Math.*, 8(1):171–6, 1958.

[25] J. W. Stayman and J. A. Fessler. Regularization for uniform spatial resolution properties in penalized-likelihood image reconstruction. *IEEE Trans. Med. Imag.*, 19(6):601–15, June 2000.

[26] J-B. Thibault, K. Sauer, C. Bouman, and J. Hsieh. A three-dimensional statistical approach to improved image quality for multi-slice helical CT. *Med. Phys.*, 34(11):4526–44, November 2007.

[27] M. Zhu, S. Wright, and T. Chan. Duality-based algorithms for total-variation-regularized image restoration. *Comput. Optim. Appl.*, 47(3):377–400, 2010.

Madison G. McGaffin received the BSEE degree in 2010 from Tufts University in Medford, Massachusetts and the MSEE degree in 2012 from the University of Michigan in Ann Arbor, where he is currently pursuing the Ph.D. degree, also in electrical engineering.

His research interests include statistical image reconstruction and parallel computing.

Jeffrey A. Fessler received the BSEE degree from Purdue University in 1985, the MSEE degree from Stanford University in 1986, and the M.S. degree in Statistics from Stanford University in 1989. From 1985 to 1988 he was a National Science Foundation Graduate Fellow at Stanford, where he earned a Ph.D. in electrical engineering in 1990. He has worked at the University of Michigan since then. From 1991 to 1992 he was a Department of Energy Alexander Hollaender Post-Doctoral Fellow in the Division of Nuclear Medicine. From 1993 to 1995 he was an Assistant Professor in Nuclear Medicine and the Bioengineering Program. He is now a Professor in the Departments of Electrical Engineering and Computer Science, Radiology, and Biomedical Engineering. He is a Fellow of the IEEE, for contributions to the theory and practice of image reconstruction. He received the Francois Erbsmann award for his IPMI93 presentation, and received the Edward Hoffman Medical Imaging Scientist Award in 2013. He has been an associate editor for the IEEE Signal Processing Letters, the IEEE Trans. on Medical Imaging, and the IEEE Trans. on Image Processing. He is currently an associate editor for the IEEE Trans. on Computational Imaging. He was co-chair of the 1997 SPIE conference on Image Reconstruction and Restoration, technical program co-chair of the 2002 IEEE Intl. Symposium on Biomedical Imaging (ISBI), and was general chair of ISBI 2007. He served as chair of the Steering Committee of the IEEE Trans. on Medical Imaging, and as Chair of the ISBI Steering Committee. He served as Associate Chair of his Department from 2006-2008. His research interests are in statistical aspects of imaging problems, and he has supervised doctoral research in PET, SPECT, X-ray CT, MRI, and optical imaging problems.