Reduced memory augmented Lagrangian algorithm for 3D iterative X-ray CT image reconstruction

Madison G. McGaffin, Sathish Ramani, and Jeffrey A. Fessler

Department of Electrical Engineering and Computer Science
University of Michigan, Ann Arbor, MI

ABSTRACT

Although statistical image reconstruction methods for X-ray CT can provide improved image quality at reduced patient doses, computation times for 3D axial and helical CT are a challenge. Rapidly converging algorithms are needed for practical use. Augmented Lagrangian methods based on variable splitting recently have been found to be effective for image denoising and deblurring applications.[5] These methods are particularly effective for non-smooth regularizers such as total variation or those involving the ℓ_1 norm. However, when standard "split Bregman" methods[6] are applied directly to 3D X-ray CT problems, numerous auxiliary variables are needed, leading to undesirably high memory requirements.[7] For minimizing regularized, weighted least-squares (WLS) cost functions, we propose a new splitting approach for CT, based on the alternating direction method of multipliers (ADMM)[1],[5] that has multiple benefits over previous methods: (i) reduced memory requirements, (ii) effective preconditioning using modified ramp/cone filters, (iii) accommodating very general regularizers including edge-preserving roughness penalties, total variation methods, and sparsifying transforms like wavelets. Numerical results show that the proposed algorithm converges rapidly, and that the cone filter is particularly effective for accelerating convergence.

1. INTRODUCTION

Consider a statistical 3D CT image reconstruction algorithm that estimates the unknown volume $\boldsymbol{x} \in \mathbb{R}^N$ as $\hat{\boldsymbol{x}}$, the minimizer of a cost function, $f(\boldsymbol{x})$:

$$\begin{aligned} \hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \quad & f(\boldsymbol{x}) \\ & f(\boldsymbol{x}) = \frac{1}{2} \left\| \boldsymbol{y} - \boldsymbol{A}\boldsymbol{x} \right\|_{\mathbf{W}}^2 + \mathsf{R}\left(\boldsymbol{C}\boldsymbol{x}\right). \end{aligned} \tag{1}$$

The statistical weights $\mathbf{W}$ are generated from the noise-corrupted projections $\boldsymbol{y} \in \mathbb{R}^M$ and have significant dynamic range. The 3D CT forward projection operator is $\boldsymbol{A} \in \mathbb{R}^{M \times N}$, and $\mathsf{R}\left(\boldsymbol{C}\boldsymbol{x}\right)$ is an edge-preserving regularizer of the form

$$\mathsf{R}\left(\boldsymbol{C}\boldsymbol{x}\right) = \beta \sum_{r=1}^{N_r} \kappa_r \phi\left(\left[\boldsymbol{C}\boldsymbol{x}\right]_r\right). \tag{2}$$

The linear transform $\boldsymbol{C} \in \mathbb{R}^{N_r \times N}$ may be a finite-differencing matrix, wavelet transform, or some other combination of sparsifying linear transforms. Often, N_r is larger than N: for a difference-of-neighbors $\boldsymbol{C}$ in 3D CT, $N_r = 13N$.[10] The parameters β and $\{\kappa_r\}_r$ control the strength of the regularization and encourage uniform spatial resolution, respectively. The penalty function ϕ is even, nonnegative and convex but not necessarily differentiable. Possible choices include $\phi(x) = |x|$ for ℓ_1-style regularizers such as anisotropic total-variation or the differentiable Fair potential $\phi_{\text{FP}}(x) = \left|\frac{x}{\delta}\right| - \log\left(1 + \left|\frac{x}{\delta}\right|\right)$, where $\delta > 0$.

Conjugate gradient-based approaches to solving (1) will obviously have difficulty with non-differentiable regularizers, but even with smooth regularizers, problems arise. The Hessian of f is

$$\boldsymbol{A}^{\mathrm{T}}\mathbf{W}\boldsymbol{A} + \nabla^2 \mathsf{R}\left(\boldsymbol{C}\boldsymbol{x}\right).$$

Supported in part by NIH grant R01 HL 098686.

Madison G. McGaffin: mcgaffin@umich.edu, 734-763-5022
Sathish Ramani: sramani@umich.edu, 734-615-5735
Jeffrey A. Fessler: fessler@umich.edu, 734-763-1434

Medical Imaging 2012: Physics of Medical Imaging, edited by Norbert J. Pelc, Robert M. Nishikawa, Bruce R. Whiting,
Proc. of SPIE Vol. 8313, 831327 · © 2012 SPIE · CCC code: 1605-7422/12/$18 · doi: 10.1117/12.910941

With highly space-varying $\mathbf{W}$, the $\boldsymbol{A}^{\mathrm{T}}\mathbf{W}\boldsymbol{A}$ term becomes difficult to efficiently precondition,[4] leading to slow convergence. Storage of $\boldsymbol{C}\boldsymbol{x}$ should also be avoided if possible, as it may incur unacceptably large memory requirements. For example, for a 512^3 image volume $\boldsymbol{C}\boldsymbol{x}$ is almost 7 gigabytes using 32-bit floating point values.

A convergent iterative algorithm for solving (1) in the 2D setting was proposed earlier[7] using the alternating directions method of multipliers.[3] The algorithm performs well, but relies on storing several vectors in $\mathbb{R}^{N_r}$, requiring significant amounts of memory especially in the 3D setting. We present here a modification of this algorithm requiring less memory and intended for use in 3D CT.

2. METHODS

We use a variable splitting scheme to separate the terms of the cost function f. Define the auxiliary variables $\boldsymbol{u} \in \mathbb{R}^M$, $\boldsymbol{v} \in \mathbb{R}^N$ and consider the following constrained minimization, equivalent to (1):

$$\begin{aligned} \hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \quad & \frac{1}{2}\left|\left|\boldsymbol{y}-\boldsymbol{u}\right|\right|_{\mathbf{W}}^2 + \mathsf{R}\left(\boldsymbol{C}\boldsymbol{v}\right) \\ \text{subject to} \quad & \boldsymbol{u} = \boldsymbol{A}\boldsymbol{x} \\ & \boldsymbol{v} = \boldsymbol{x}. \end{aligned} \tag{3}$$

This splitting differs from the splitting presented earlier[9] by defining $\boldsymbol{v} = \boldsymbol{x}$ instead of $\boldsymbol{v} = \boldsymbol{C}\boldsymbol{x}$.

With a convex regularizer, (3) can be solved using the alternating directions method of multipliers,[3] which involves numerically tractable sequential minimizations with respect to each of $\boldsymbol{x}$, $\boldsymbol{u}$ and $\boldsymbol{v}$ individually.

Define the augmented Lagrangian function $\mathcal{L}$ for (3):

$$\mathcal{L}\left(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v};\boldsymbol{\gamma}_{\boldsymbol{u}},\boldsymbol{\gamma}_{\boldsymbol{v}}\right) = \frac{1}{2}\left|\left|\boldsymbol{y}-\boldsymbol{u}\right|\right|_{\mathbf{W}}^2 + \mathsf{R}\left(\boldsymbol{C}\boldsymbol{v}\right) + \boldsymbol{\gamma}_{\boldsymbol{u}}^{\mathrm{T}}\left(\boldsymbol{u}-\boldsymbol{A}\boldsymbol{x}\right) + \frac{\mu_u}{2}\left|\left|\boldsymbol{u}-\boldsymbol{A}\boldsymbol{x}\right|\right|^2 + \boldsymbol{\gamma}_{\boldsymbol{v}}^{\mathrm{T}}\left(\boldsymbol{v}-\boldsymbol{x}\right) + \frac{\mu_v}{2}\left|\left|\boldsymbol{v}-\boldsymbol{x}\right|\right|^2. \tag{4}$$

The positive real parameters μ_u, μ_v can affect the rate of convergence but not the final solution.[3] Define $\boldsymbol{\eta}_{\boldsymbol{u}} = -\frac{1}{\mu_u}\boldsymbol{\gamma}_{\boldsymbol{u}}$, $\boldsymbol{\eta}_{\boldsymbol{v}} = -\frac{1}{\mu_v}\boldsymbol{\gamma}_{\boldsymbol{v}}$. The Lagrangian function can be rewritten by absorbing the $\boldsymbol{\gamma}_{\boldsymbol{u}}$ and $\boldsymbol{\gamma}_{\boldsymbol{v}}$ terms into penalty terms and discarding irrelevant constants:[8]

$$\mathcal{L}\left(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v};\boldsymbol{\eta}_{\boldsymbol{u}},\boldsymbol{\eta}_{\boldsymbol{v}}\right) = \frac{1}{2}\left|\left|\boldsymbol{y}-\boldsymbol{u}\right|\right|_{\mathbf{W}}^2 + \mathsf{R}\left(\boldsymbol{C}\boldsymbol{v}\right) + \frac{\mu_u}{2}\left|\left|\boldsymbol{u}-\left(\boldsymbol{A}\boldsymbol{x}+\boldsymbol{\eta}_{\boldsymbol{u}}\right)\right|\right|^2 + \frac{\mu_v}{2}\left|\left|\boldsymbol{v}-\left(\boldsymbol{x}+\boldsymbol{\eta}_{\boldsymbol{v}}\right)\right|\right|^2. \tag{5}$$

The standard method of multipliers augmented Lagrangian technique prescribes the following updates to achieve a solution of the original constrained minimization problem:

$$\boldsymbol{x}^{(j+1)},\boldsymbol{u}^{(j+1)},\boldsymbol{v}^{(j+1)} \leftarrow \underset{\boldsymbol{x},\boldsymbol{u},\boldsymbol{v}}{\operatorname{argmin}}\, \mathcal{L}\left(\boldsymbol{x},\boldsymbol{u},\boldsymbol{v};\boldsymbol{\eta}_{\boldsymbol{u}}^{(j)},\boldsymbol{\eta}_{\boldsymbol{v}}^{(j)}\right) \tag{6}$$

$$\boldsymbol{\eta}_{\boldsymbol{u}}^{(j+1)} \leftarrow \boldsymbol{\eta}_{\boldsymbol{u}}^{(j)} + \left(\boldsymbol{A}\boldsymbol{x}^{(j+1)} - \boldsymbol{u}^{(j+1)}\right); \qquad \boldsymbol{\eta}_{\boldsymbol{v}}^{(j+1)} \leftarrow \boldsymbol{\eta}_{\boldsymbol{v}}^{(j)} + \left(\boldsymbol{x}^{(j+1)} - \boldsymbol{v}^{(j+1)}\right). \tag{7}$$

The joint minimization with respect to $\boldsymbol{x}$, $\boldsymbol{u}$ and $\boldsymbol{v}$ remains numerically difficult. Fortunately, the alternating directions method of multipliers allows (6) to be decoupled into three sequential updates:

$$\boldsymbol{x}^{(j+1)} \leftarrow \underset{\boldsymbol{x}}{\operatorname{argmin}}\, \mathcal{L}\left(\boldsymbol{x},\boldsymbol{u}^{(j)},\boldsymbol{v}^{(j)};\boldsymbol{\eta}_{\boldsymbol{u}}^{(j)},\boldsymbol{\eta}_{\boldsymbol{v}}^{(j)}\right) \tag{8}$$

$$\boldsymbol{u}^{(j+1)} \leftarrow \underset{\boldsymbol{u}}{\operatorname{argmin}}\, \mathcal{L}\left(\boldsymbol{x}^{(j+1)},\boldsymbol{u},\boldsymbol{v}^{(j)};\boldsymbol{\eta}_{\boldsymbol{u}}^{(j)},\boldsymbol{\eta}_{\boldsymbol{v}}^{(j)}\right) \tag{9}$$

$$\boldsymbol{v}^{(j+1)} \leftarrow \underset{\boldsymbol{v}}{\operatorname{argmin}}\, \mathcal{L}\left(\boldsymbol{x}^{(j+1)},\boldsymbol{u}^{(j+1)},\boldsymbol{v};\boldsymbol{\eta}_{\boldsymbol{u}}^{(j)},\boldsymbol{\eta}_{\boldsymbol{v}}^{(j)}\right), \tag{10}$$

which are numerically easier to perform.

- Initialize variables: $\boldsymbol{\eta_v} \leftarrow 0$, $\boldsymbol{\eta_u} \leftarrow 0$, $\boldsymbol{x} \leftarrow \boldsymbol{x}_{fdk}$, $\boldsymbol{u} \leftarrow \boldsymbol{A}\boldsymbol{x}$, $\boldsymbol{v} \leftarrow \boldsymbol{x}$.
- Iterate n times:
  - Update $\boldsymbol{x}$ by approximating (11) using preconditioned conjugate gradients with one of the preconditioners from Table 1.
  - Update $\boldsymbol{u}$ with (12).
  - Approximately solve (13) to update $\boldsymbol{v}$ with some iterative procedure.
  - Update the Lagrange variables $\boldsymbol{\eta_u}$, $\boldsymbol{\eta_v}$ using (7).

Figure 1. Summary of the proposed algorithm.

Another remarkable feature of the alternating directions method of multipliers framework is that the updates (8)-(10) do not need to be performed exactly to guarantee convergence:

Let $\boldsymbol{x}_*^{(j)}$, $\boldsymbol{u}_*^{(j)}$, and $\boldsymbol{v}_*^{(j)}$ be the exact solutions to updates (8)-(10). Define $\varepsilon_x^{(j)} = \left|\left|\boldsymbol{x}_*^{(j)} - \boldsymbol{x}^{(j)}\right|\right|$, $\varepsilon_u^{(j)} = \left|\left|\boldsymbol{u}_*^{(j)} - \boldsymbol{u}^{(j)}\right|\right|$, $\varepsilon_v^{(j)} = \left|\left|\boldsymbol{v}_*^{(j)} - \boldsymbol{v}^{(j)}\right|\right|$. If the series

$$\sum_{j=1}^{\infty} \varepsilon_x^{(j)}; \qquad \sum_{j=1}^{\infty} \varepsilon_u^{(j)}; \qquad \sum_{j=1}^{\infty} \varepsilon_v^{(j)}$$

all converge, then $\left\{\boldsymbol{x}^{(j)}\right\}_j$, $\left\{\boldsymbol{u}^{(j)}\right\}_j$ and $\left\{\boldsymbol{v}^{(j)}\right\}_j$ will all converge[3] to a solution of the original constrained optimization problem (3).

The Lagrangian function $\mathcal{L}$ is quadratic in $\boldsymbol{x}$ and $\boldsymbol{u}$, so updates (8) and (9) can be written in closed form:

$$\boldsymbol{x}^{(j+1)} \leftarrow \left(\mu_u \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \mu_v \boldsymbol{I}\right)^{-1} \left(\mu_u \boldsymbol{A}^{\mathrm{T}} \left(\boldsymbol{u}^{(j)} - \boldsymbol{\eta}_{\boldsymbol{u}}^{(j)}\right) + \mu_v \left(\boldsymbol{v}^{(j)} - \boldsymbol{\eta}_{\boldsymbol{v}}^{(j)}\right)\right), \tag{11}$$

$$\boldsymbol{u}^{(j+1)} \leftarrow \left(\mathbf{W} + \mu_u \boldsymbol{I}\right)^{-1} \left(\mathbf{W}\boldsymbol{y} + \mu_u \left(\boldsymbol{A}\boldsymbol{x}^{(j+1)} + \boldsymbol{\eta}_{\boldsymbol{u}}^{(j)}\right)\right) \tag{12}$$

In the $\boldsymbol{u}$ update (12), the matrix $(\mathbf{W} + \mu_u \boldsymbol{I})$ is diagonal. If $\boldsymbol{A}\boldsymbol{x}^{(j+1)}$ is precomputed after the $\boldsymbol{x}$ update and stored for the $\boldsymbol{\eta_u}$ update (7), then an exact $\boldsymbol{u}$ update involves only vector addition and element-wise scaling.

The matrix $\left(\mu_u \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \mu_v \boldsymbol{I}\right)$ is too large in any practical 3D CT problem to invert directly: an iterative algorithm like preconditioned gradient descent (PCG) may be used instead. If the PCG-driven update achieves a residual that decreases faster than $1/j$ at the jth iteration, convergence is guaranteed.[3] This will potentially require more inner iterations of PCG for each outer iteration j. Warm-starting PCG with the previous value of $\boldsymbol{x}$ may help decrease the number of inner iterations. Regardless, each inner iteration of PCG will require the application of the time-consuming forward and back-projection operations, $\boldsymbol{A}$ and $\boldsymbol{A}^{\mathrm{T}}$. Consequently, preconditioners that allow less iterations of PCG to be run are of great interest.

In the 2D setting $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ is "somewhat" shift invariant,[2] so (11) can be effectively preconditioned using shift invariant cone-type filters. Such preconditioners require only one 2D FFT/inverse-FFT pair per application, and were shown to be effective in accelerating PCG applied to inverting $\left(\mu_u \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \mu_v \boldsymbol{I}\right)$ in 2D CT.[9]

The "nearly" shift invariant assumption can be extended to the 3D setting in a number of ways, with variations arising in how the assumption of shift-invariance is treated in the axial dimension. If $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ is assumed to be nearly shift-invariant throughout the entire volume, a single 3D filter may be used. The preconditioner `filt3d` in Table 1 is built with this assumption. However, this is a rather strong assumption particularly in axial CT, because slices towards the edge of the imaged volume have significantly different footprints on the X-ray detector. Another additional preconditioner is suggested in Table 1 that does not assume $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$ shift-invariant in the axial dimension. The `slice-filt2d` preconditioner assumes shift invariance within each slice but does not attempt to invert any axial behavior of $\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$. A benefit of `slice-filt2d` is a slight decrease in computational complexity relative to the `filt3d` preconditioner.

Table 1. Two linear shift invariant (LSI) preconditioners approximating $\left(\mu_u \boldsymbol{A}^{\mathrm{T}}\boldsymbol{A} + \mu_v \boldsymbol{I}\right)^{-1}$.

	Description	**Storage**	**Complexity**
`filt3d`	3D LSI filter	$N_x N_y N_z$	$O\left(N_x N_y N_z \log\left(N_x N_y N_z\right)\right)$
`slice-filt2d`	2D LSI filter for each slice	$N_x N_y N_z$	$O\left(N_x N_y N_z \log\left(N_x N_y\right)\right)$

The $\boldsymbol{v}$ update requires the minimization of a function that is convex but not typically not quadratic. It is a regularized denoising problem of the following form:

$$\boldsymbol{v}^{(j+1)} \leftarrow \underset{\boldsymbol{v}}{\operatorname{argmin}} \frac{\mu_v}{2} \left|\left| \boldsymbol{v} - \left(\boldsymbol{x}^{(j+1)} + \boldsymbol{\eta}_{\boldsymbol{v}}^{(j)}\right) \right|\right|^2 + \mathsf{R}\left(\boldsymbol{C}\boldsymbol{v}\right). \tag{13}$$

This is a penalized least squares denoising operation, for which many algorithms exist. The denoising step does not need to be exact, so long as approximation error decreases faster than $1/j$ for the jth iteration. Some care must be taken to select denoising algorithms which do not rely on splitting the $\boldsymbol{v}$ and $\boldsymbol{C}\boldsymbol{v}$ terms, as this splitting incurs the memory cost the proposed algorithm is designed to avoid without a clear benefit.

The inexactness of the $\boldsymbol{v}$ update and the amount of time required to run an iterative algorithm to approximately solve (12) are the primary costs of this algorithm's lower memory requirement relative to the "$\boldsymbol{v} = \boldsymbol{C}\boldsymbol{x}$" splitting.

The parameters μ_u and μ_v do not effect the final solution of the algorithm. Selecting $\mu_u = \text{median}\left\{w_i\right\}$ makes $\boldsymbol{u}$ update (12) well conditioned. In a similar vein, $\mu_v = \frac{\lambda_{max}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)}{\mu_u}$ would better condition the $\boldsymbol{x}$ update (11), but we found $\mu_v = \frac{1}{1000}\frac{\lambda_{max}\left(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}\right)}{\mu_u}$ leads to swifter convergence.

The proposed algorithm is summarized in Figure 1. Table 2 summarizes the differences in memory requirements between the proposed algorithm and the "$\boldsymbol{v} = \boldsymbol{C}\boldsymbol{x}$"-splitting ADMM algorithm.

3. EXPERIMENTS

We performed several reconstructions of a simulated $256 \times 256 \times 96$-pixel phantom downsampled 4 times. The projection data was numerically generated from a simulation of an axial scan with GE Lightspeed fan-beam geometry[11] and a monoenergic source at 10^5 photon counts per ray. The Fair potential penalty function ϕ_{FP} was used with hyperparameter $\delta = .01$, and we set the regularization strength parameter $\beta = 2^5$. A finite-differencing matrix using all 26 neighbors of each interior voxel (*i.e.*, $N_r = 13N$), was selected for $\boldsymbol{C}$. The Fair potential is strictly convex, so (1) has a unique solution, $\boldsymbol{x}^*$. A nonlinear conjugate gradient solver was run for 500 iterations to generate an estimate of the final converged solution, $\boldsymbol{x}^*$; and the Feldkamp filtered backprojection algorithm was used to generate $\boldsymbol{x}_{fdk}$.

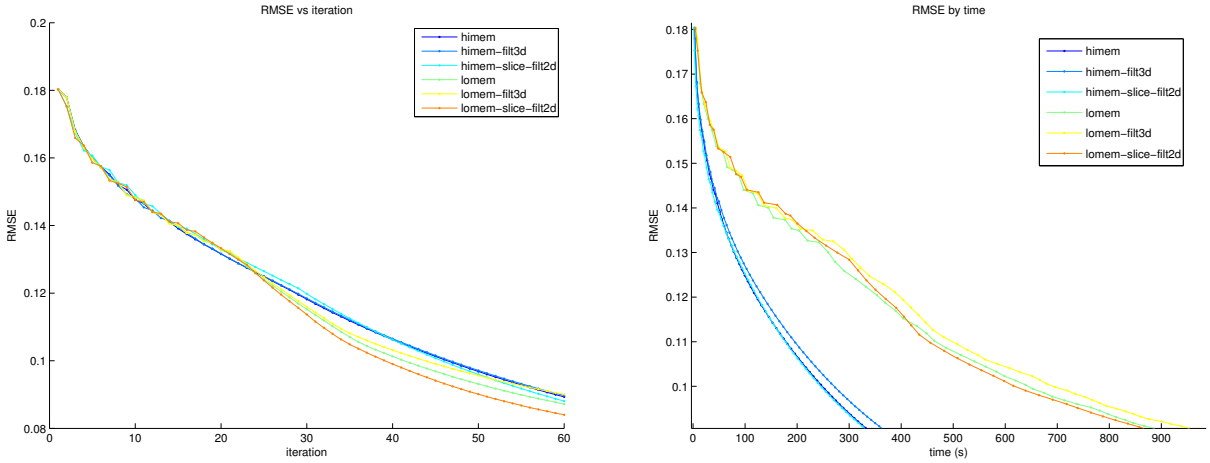
The $\boldsymbol{v}$ update step in the proposed algorithm was implemented with unpreconditioned nonlinear conjugate gradients, and the $\boldsymbol{x}$ update step was performed with at most 15 steps of PCG with warm starting. The proposed algorithm and the high-memory ADMM algorithm were run for 60 iterations with each of the preconditioners from Table 1. The root mean square error between each intermediate reconstruction at every iteration and the converged solution $\boldsymbol{x}^*$ was computed and plotted against both time and iteration number; see Figure 2.

The high-memory ADMM algorithm converged more quickly than the proposed low-memory algorithm in general and for each preconditioner selection. The 3D preconditioner `filt3d` decreased both algorithm's convergence rate, even ignoring the computational cost of applying the preconditioner. However, the `slice-filt2d` preconditioner increased the convergence rate of both the low-memory and high-memory ADMM algorithms.

Table 2. Comparison of memory requirements of "low-memory" and "high-memory" ADMM algorithms

	Number of values stored							
	W	$\boldsymbol{y}$	$\boldsymbol{x}$	$\boldsymbol{u}$	$\boldsymbol{v}$	$\boldsymbol{\eta_u}$	$\boldsymbol{\eta_v}$	**Example**
high-memory	M	M	N	M	N_r	M	N_r	≈ 2.7 GIB
proposed	M	M	N	M	N	M	N	≈ 1.1 GIB

The "example" is a $N = 512 \times 512 \times 64$ volume imaged with the GE Lightspeed with a single turn, *i.e.*, $M = 888 \times 64 \times 984$. All 26 voxel neighbors are used for regularization, $N_r = 13N$ discounting edge voxels. All floating point values are assumed to be stored in 32-bit single-precision. Results are given in gibibytes, 1 GIB = 2^{30} bytes.



(a) (b)

Figure 2. RMSE series of high-memory and proposed low-memory ADMM algorithms for two preconditioners plotted against (a) iteration and (b) time.

The low-memory ADMM algorithm performed as well as if not slightly better than the high-memory ADMM algorithm on a per-iteration basis. This suggests that, while our current implementation is significantly slower than the high-memory version, speeding up the parallelizable $\boldsymbol{v}$ update step may yield an algorithm with both modest memory requirements and rapid convergence.

4. CONCLUSIONS

We have developed a flexible new X-ray CT reconstruction algorithm with moderate memory requirements. The most performance-critical aspects of the new algorithm are a penalized weighted least squares denoising operation and a matrix inversion. Both operations can be performed approximately without sacrificing convergence, and the matrix inversion may be effectively preconditioned using a series of 2D linear shift invariant filters. The next steps will be to accelerate the $\boldsymbol{v}$ update step, compare the proposed algorithm to other axial CT image reconstruction algorithms in the literature, and explore extensions to helical CT.

REFERENCES

[1] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo. An augmented Lagrangian approach to the constrained optimization formulation of imaging inverse problems. *IEEE Trans. Im. Proc.*, 20(3):681–695, 2011.

[2] N. H. Clinthorne, T. S. Pan, P. C. Chiao, W. L. Rogers, and J. A. Stamos. Preconditioning methods for improved convergence rates in iterative reconstructions. *IEEE Trans. Med. Imag.*, 12(1):78–83, March 1993.

[3] Jonathan Eckstein and Dimitri P. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55:293–318, 1992.

[4] J. A. Fessler and S. D. Booth. Conjugate-gradient preconditioning methods for shift-variant PET image reconstruction. *IEEE Trans. Im. Proc.*, 8(5):688–99, May 1999.

[5] M. A. T. Figueiredo and José M Bioucas-Dias. Restoration of Poissonian images using alternating direction optimization, 2010.

[6] T. Goldstein and S. Osher. The split Bregman method for L1-regularized problems. *SIAM J. Imaging Sci.*, 2(2):323–43, 2009.

[7] S. Ramani and J. A. Fessler. Convergent iterative CT reconstruction with sparsity-based regularization. In *Proc. Intl. Mtg. on Fully 3D Image Recon. in Rad. and Nuc. Med*, pages 302–5, 2011.

[8] S. Ramani and J. A. Fessler. Parallel MR image reconstruction using augmented Lagrangian methods. *IEEE Trans. Med. Imag.*, 30(3):694–706, March 2011.

[9] S. Ramani and J. A. Fessler. A splitting-based iterative algorithm for accelerated statistical X-ray CT reconstruction. *IEEE Trans. Med. Imag.*, 2012. To appear.

[10] 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.

[11] J. Wang, T. Li, H. Lu, and Z. Liang. Penalized weighted least-squares approach to sinogram noise reduction and image reconstruction for low-dose X-ray computed tomography. *IEEE Trans. Med. Imag.*, 25(10):1272–83, October 2006.