Fast 3D reconstruction method for differential phase contrast X-ray CT

Michael T. McCann,[1,2,3,*] **Masih Nilchian,**[1,3] **Marco Stampanoni,**[4,5] **and Michael Unser**[1]

[1]*Biomedical Imaging Group, EPFL, Lausanne, Switzerland*
[2]*Center for Biomedical Imaging, Signal Processing Core, EPFL, Lausanne, Switzerland*
[3]*The first two authors contributed equally to the work*
[4]*Institute for Biomedical Engineering, University and ETH Zurich, Zurich, Switzerland*
[5]*Swiss Light Source, Paul Scherrer Institute, Villigen, Switzerland*

[*]*michael.mccann@epfl.ch*

Abstract: We present a fast algorithm for fully 3D regularized X-ray tomography reconstruction for both traditional and differential phase contrast measurements. In many applications, it is critical to reduce the X-ray dose while producing high-quality reconstructions. Regularization is an excellent way to do this, but in the differential phase contrast case it is usually applied in a slice-by-slice manner. We propose using fully 3D regularization to improve the dose/quality trade-off beyond what is possible using slice-by-slice regularization. To make this computationally feasible, we show that the two computational bottlenecks of our iterative optimization process can be expressed as discrete convolutions; the resulting algorithms for computation of the X-ray adjoint and normal operator are fast and simple alternatives to regridding. We validate this algorithm on an analytical phantom as well as conventional CT and differential phase contrast measurements from two real objects. Compared to the slice-by-slice approach, our algorithm provides a more accurate reconstruction of the analytical phantom and better qualitative appearance on one of the two real datasets.

OCIS codes: (100.3190) Inverse problems; (100.3010) Image reconstruction techniques; (110.7440) X-ray imaging.

References and links

1. A. Webb, *Introduction to Biomedical Imaging* (Wiley-IEEE, 2002).
2. E. Maire and P. J. Withers, "Quantitative X-ray tomography," Int. Mater. Rev. **59**(1), 1–43 (2014).
3. A. Sakdinawat and D. Attwood, "Nanoscale X-ray imaging," Nat. Photonics **4**(12), 840–848 (2010).
4. J. Hsieh, *Computed Tomography*, 2nd ed. (SPIE, Bellingham, WA, 2009).
5. V. N. Ingal and E. A. Beliaevskaya, "X-ray plane-wave topography observation of the phase contrast from a non-crystalline object," J. Phys. D: Appl. Phys. **28**(11), 2314–2317 (1995).
6. T. J. Davis, D. Gao, T. E. Gureyev, A. W. Stevenson, and S. W. Wilkins, "Phase-contrast imaging of weakly absorbing materials using hard X-rays," Nature **373**(6515), 595–598 (1995).
7. D. Chapman, W. Thomlinson, R. E. Johnston, D. Washburn, E. Pisano, N. Gmr, Z. Zhong, R. Menk, F. Arfelli, and D. Sayers, "Diffraction enhanced X-ray imaging," Phys. Med. Biol. **42**(11), 2015–2025 (1997).
8. A. Snigirev, I. Snigireva, V. Kohn, S. Kuznetsov, and I. Schelokov, "On the possibilities of X-ray phase contrast microimaging by coherent high-energy synchrotron radiation," Rev. Sci. Instrum. **66**(12), 5486–5492 (1995).
9. K. A. Nugent, T. E. Gureyev, D. F. Cookson, D. Paganin, and Z. Barnea, "Quantitative phase imaging using hard X rays," Phys. Rev. Lett. **77**(14), 2961–2964 (1996).
10. S. W. Wilkins, T. E. Gureyev, D. Gao, A. Pogany, and A. W. Stevenson, "Phase-contrast imaging using polychromatic hard X-rays," Nature **384**(6607), 335–338 (1996).

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016

11. U. Bonse and M. Hart, "An X-ray interferometer," Appl. Phys. Lett. **6**(8), 155–156 (1965).
12. A. Momose, T. Takeda, Y. Itai, and K. Hirano, "Phase-contrast X-ray computed tomography for observing biological soft tissues," Nat. Med. **2**(4), 473–475 (1996).
13. T. Weitkamp, A. Diaz, C. David, F. Pfeiffer, M. Stampanoni, P. Cloetens, and E. Ziegler, "X-ray phase imaging with a grating interferometer," Opt. Express **13**(16), 6296–6304 (2005).
14. B. M. Weon, J. H. Je, Y. Hwu, and G. Margaritondo, "Phase contrast X-ray imaging," Int. J. Nanotechnol. **3**(2-3), 280–297 (2006).
15. R. Fitzgerald, "Phase-sensitive X-ray imaging," Phys. Today **53**(7), 23–26 (2007).
16. F. Pfeiffer, O. Bunk, C. Kottler, and C. David, "Tomographic reconstruction of three-dimensional objects from hard X-ray differential phase contrast projection images," Nucl. Instrum. Methods Phys. Res., Sect. A **580**(2), 925–928 (2007).
17. M. Stampanoni, Z. Wang, T. Thring, C. David, E. Roessl, M. Trippel, R. A. Kubik-Huch, G. Singer, M. K. Hohl, and N. Hauser, "The first analysis and clinical evaluation of native breast tissue using differential phase-contrast mammography," Invest. Radiol. **46**(12), 801–806 (2011).
18. M. Beister, D. Kolditz, and W. A. Kalender, "Iterative reconstruction methods in X-ray CT," Physica Med. **28**(2), 94–108 (2012).
19. K. Zhang, Y. Hong, P. Zhu, Q. Yuan, W. Huang, Z. Wang, S. Chu, S. A. McDonald, F. Marone, M. Stampanoni, and Z. Wu, "Study of OSEM with different subsets in grating-based X-ray differential phase-contrast imaging," Anal. Bioanal. Chem. **401**(3), 837–844 (2011).
20. J. Fu, X. Hu, A. Velroyen, M. Bech, M. Jiang, and F. Pfeiffer, "3D algebraic iterative reconstruction for cone-beam X-ray differential phase-contrast computed tomography," PLoS One **10**(3), e0117502 (2015).
21. Z. Qi, J. Zambelli, N. Bevins, and G.-H. Chen, "A novel method to reduce data acquisition time in differential phase contrast: computed tomography using compressed sensing," in *SPIE Proceedings*, vol. 7258, pp. 72,584A–72,584A–8 (Lake Buena Vista, Florida, United States, 2009).
22. T. Khler, B. Brendel, and E. Roessl, "Iterative reconstruction for differential phase contrast imaging using spherically symmetric basis functions," Med. Phys. **38**(8), 4542–4545 (2011).
23. Q. Xu, E. Y. Sidky, X. Pan, M. Stampanoni, P. Modregger, and M. A. Anastasio, "Investigation of discrete imaging models and iterative image reconstruction in differential X-ray phase-contrast tomography," Opt. Express **20**(10), 10724–10749 (2012).
24. M. Nilchian, C. Vonesch, P. Modregger, M. Stampanoni, and M. Unser, "Fast iterative reconstruction of differential phase contrast X-ray tomograms," Opt. Express **21**(5), 5511–5528 (2013).
25. S. Lefkimmiatis, J. P. Ward, and M. Unser, "Hessian schatten-norm regularization for linear inverse problems," IEEE Trans. Image Process. **22**(5), 1873–1888 (2013).
26. M. McGaffin and J. Fessler, "Alternating dual updates algorithm for X-ray CT reconstruction on the GPU," IEEE Trans. Comput. Imaging **1**(3), 186–199 (2015).
27. H. Nien and J. Fessler, "Fast X-ray CT image reconstruction using a linearized augmented lagrangian method with ordered subsets," IEEE Trans. Med. Imaging **34**(2), 388–399 (2015).
28. F. Arcadu, M. Nilchian, A. Studer, M. Stampanoni, and F. Marone, "A forward regridding method with minimal oversampling for accurate and efficient iterative tomographic algorithms," IEEE Trans. Image Process. **25**(3), 1207–1218 (2016).
29. A. Delaney and Y. Bresler, "A fast and accurate Fourier algorithm for iterative parallel-beam tomography," IEEE Trans. Image Process. **5**(5), 740–753 (1996).
30. S. Horbelt, M. Liebling, and M. Unser, "Discretization of the radon transform and of its inverse by spline convolutions," IEEE Trans. Med. Imaging **21**(4), 363–376 (2002).
31. A. Entezari, M. Nilchian, and M. Unser, "A box spline calculus for the discretization of computed tomography reconstruction problems," IEEE Trans. Med. Imaging **31**(8), 1532–1541 (2012).
32. F. Momey, L. Denis, C. Burnier, E. Thiebaut, J.-M. Becker, and L. Desbat, "Spline driven: high accuracy projectors for tomographic reconstruction from few projections," IEEE Trans. Image Process. **24**(12), 4715–4725 (2015).
33. S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," Found. Trends Mach. Learn. **3**(1), 1–122 (2011).
34. J. Shewchuk, "An introduction to the conjugate gradient method without the agonizing pain," Tech. rep., Carnegie Mellon University (1994).
35. J. A. Fessler, "On NUFFT-based gridding for non-Cartesian MRI," J. Magn. Reson. **188**(2), 191–195 (2007).
36. M. Vetterli, J. Kovacevic, and V. Goyal, *Foundations of Signal Processing* (Cambridge University, 2014).
37. R. M. Lewitt, "Multidimensional digital image representations using generalized Kaiser-Bessel window functions," J. Opt. Soc. Am. A. **7**(10), 1834–1836 (1990).
38. M. Nilchian, J. Ward, C. Vonesch, and M. Unser, "Optimized Kaiser-Bessel window functions for computed tomography," IEEE Trans. Image Process. **24**(11), 3826–3833 (2015).
39. L. A. Shepp, "Computerized tomography and nuclear magnetic resonance," J. Comput. Assisted Tomogr. **4**(1), 94–107 (1980).
40. M. A. T. Figueiredo and R. D. Nowak, "An EM algorithm for wavelet-based image restoration," IEEE Trans.

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016

Image Process. **12**(8), 906–916 (2003).
41. J.-L. Starck, E. J. Candes, and D. L. Donoho, "The curvelet transform for image denoising," IEEE Trans. Image Process. **11**(6), 670–684 (2002).
42. I. Tosic and P. Frossard, "Dictionary learning," IEEE Signal. Proc. Mag. **28**(2), 27–38 (2011).

1. Introduction

In X-ray computed tomography (CT), the images of slices of an object (a *tomogram*) are reconstructed from a set of X-ray images of the object taken from different angles [1]. This technique has found applications in a broad range of areas including materials science engineering, archaeology, biology, and medicine [2, 3]. Because X-rays can have harmful effects on living tissue, it is crucial in biomedical applications to keep the X-ray dose low; however, doing so generally degrades the quality of signal used for reconstruction [4].

In conventional (absorption contrast) X-ray CT, contrast in the X-ray images comes from the X-ray intensity attenuation of the sample. One alternative to absorption contrast is phase-contrast imaging (PCI), where contrast is created by changes in the phase of the X-ray beam induced by the sample. The advantage of PCI is that, for some biologic samples, the X-ray absorption is smaller than the induced phase shift, meaning that PCI offers improved contrast. The challenge of PCI is that phase information is more challenging to measure than intensity. As a result, X-ray PCI measurements require special physical setups. These include analyzer-based [5–7], free space propagation [8–10], and interferometric methods [11–13]. See [14, 15] for more information on the theory of X-ray PCI.

Our focus in this paper is grating interferometry, which produces three complementary contrasts: absorption contrast, phase contrast, and dark field measurements. This makes grating interferometry suitable for a broad range of applications, such as material sciences (e.g., material testing), biomedical research (e.g., monitoring drug effects), or even clinical diagnostics (e.g., mammography). The phase measurement is linked to the derivative of the X-ray transform of the real part of the refractive index map of the object and is therefore a type of differential phase contrast (DPC) imaging. It has been applied successfully to biological samples such as insects [13, 16], rat-brain tissue, and human breast tissue [17].

In a separate line of research, iterative reconstruction methods involving regularization have been developed that can produce high-quality reconstructions from fewer and noisier images than unregularized methods (see [18] for a review). Broadly, regularization works by including prior knowledge about the object into the reconstruction process, promoting, e.g., the piecewise smoothness of the reconstruction. There have been several recent efforts to use these techniques for DPC CT reconstruction: [19] explores 3D unregularized iterative reconstruction; [20] uses algebraic iterative reconstruction with no regularization for 3D reconstruction; [21] and [22] use regularized iterative reconstruction in 2D; and [23] and our own [24] perform 3D regularized iterative reconstruction *slice-by-slice*, where the reconstruction (including the regularization) occurs on a series of 2D slices, which are then concatenated to form a 3D reconstruction.

Slice-by-slice reconstruction is attractive when the X-ray projection angles vary around a fixed axis, as is the case in [23] and [24], because any given X-ray travels only within a single slice. Thus the 3D tomography reconstruction problem naturally decomposes into a set of 2D tomography reconstruction problems. The slice-by-slice paradigm also has roots in light microscopy, where the axial resolution is significantly worse than lateral, which means it is natural to treat data as a stack of slices. Finally, slice-by-slice reconstruction methods are practical to implement: when individual views are one megapixel images, the complete tomographic reconstruction is on the order of a gigavoxel volume, an unwieldy size. The slice-by-slice approach never requires storing this complete reconstruction in memory.

As an alternative to slice-by-slice reconstruction, we propose performing X-ray reconstruc-

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016

tion in 3D. The main benefit of fully 3D reconstruction is that regularization can then be defined in 3D as well. For example, total variation (TV) regularization applied in a slice-by-slice way promotes each slice to be piecewise constant, while TV applied in 3D promotes the reconstruction volume to be piecewise constant. Or, the Hessian Schatten-norm regularization [25] can be used to penalize second-order derivatives. This should enable the 3D reconstruction to achieve a more favorable trade-off between dose reduction and reconstruction quality. The challenge of fully 3D reconstruction is that it requires the entire reconstructed volume to fit in memory and can be computationally slow.

Finally, we note that there is ongoing work on creating fast methods for X-ray CT reconstruction across a variety of scan geometries and problem formulations, including, e.g., [26], [27], and our own work, [28]. Broadly, these either propose faster optimization methods or accelerate the computations necessary for each iteration, including forward projection ($\mathbf{H}$), back projections ($\mathbf{H}^T$), and the normal operator ($\mathbf{H}^T\mathbf{H}$). Our approach here is of the latter type. Distinct from the regridding of [28], we instead follow the lead of [29], where the quantities are computed using discrete convolutions and lookup. We extend this approach to 3D and to the DPC, providing fast, simple, and accurate algorithms for back projection and the normal operator.

In summary, we propose a fast regularized reconstruction algorithm for parallel-beam X-ray CT. This algorithm is applicable to both DPC and conventional X-ray tomography and works both in 2D and 3D; and, to our knowledge, it is the first to reconstruct DPC tomograms in full 3D, including 3D regularization. The speed of the method comes from our proposed extension of the convolutional versions of back projection and the normal operator from [29] to 3D and to DPC.

The structure of the paper is as follows. In Section 2, we formulate the X-ray tomography problem for both the conventional and DPC setting. In Section 3, we present our reconstruction algorithm, including our discretization technique, optimization scheme, and fast implementation methods. In Section 4, we present experiments on the speed and accuracy of our proposed fast algorithms. We then show reconstructions of synthetic and real data using our method in both a slice-by-slice and fully 3D setting. We conclude in Section 5.

2. Problem formulation

In this section, we fix our notation and formulate the CT reconstruction in general terms, both in the traditional absorption contrast setting and for DPC CT.

2.1. CT reconstruction

We begin with the definition of the X-ray transform. Let f be function of $\mathbb{R}^d$, $f:\mathbb{R}^d\to\mathbb{R}$ (usually with $d\in\{2,3\}$). The X-ray transform of f, denoted $\mathscr{P}_\theta\{f\}$, computes the line integrals of f along the direction specified by a unit vector $\theta\in\mathbb{R}^d$,

$$\mathscr{P}_\theta\{f\}(\mathbf{y})=\int_{\mathbb{R}} f(t\theta+\mathbf{P}_{\theta^\perp}^T\mathbf{y})dt, \tag{1}$$

where $\mathbf{P}_{\theta^\perp}^T$ is a $d\times(d-1)$ matrix that has as its columns an orthogonal basis for the tangent space of θ and $\mathbf{y}$ is a vector in that tangent space specifying the offset of the line. See Fig. 1 for a diagram of the geometry of the X-ray transform.

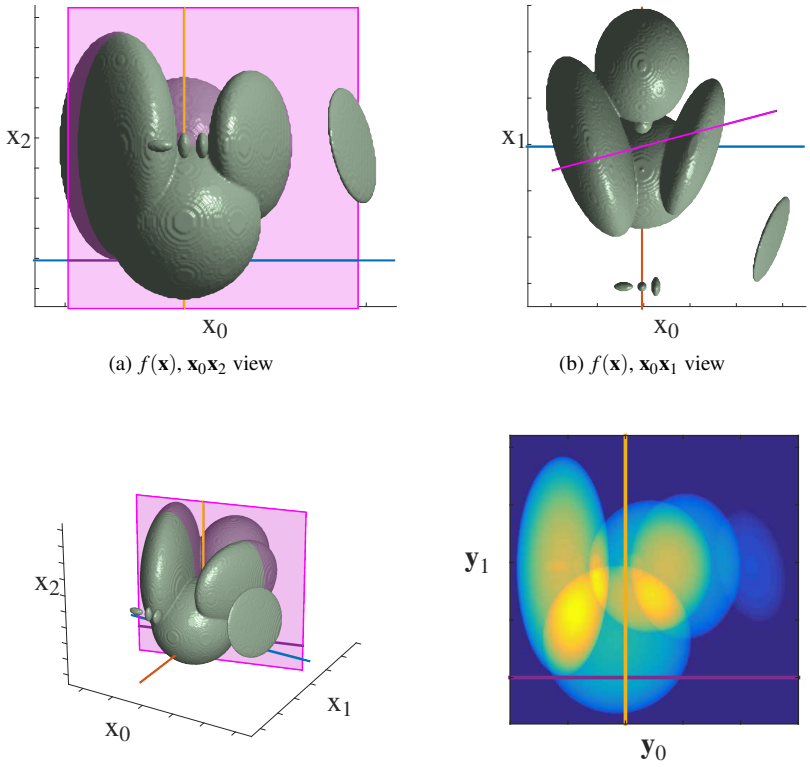
We will rely on two properties of the X-ray transform, linearity,

$$\mathscr{P}_\theta\{\alpha f+\beta g\}(\mathbf{y})=\alpha\mathscr{P}_\theta\{f\}(\mathbf{y})+\beta\mathscr{P}_\theta\{g\}(\mathbf{y}), \tag{2}$$

and projection translation invariance,

$$\mathscr{P}_\theta\{f(\cdot-\mathbf{x}_0)\}(\mathbf{y})=\mathscr{P}_\theta\{f\}(\mathbf{y}-\mathbf{P}_{\theta^\perp}\mathbf{x}_0). \tag{3}$$

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016



(a) $f(\mathbf{x})$, $\mathbf{x}_0\mathbf{x}_2$ view

(b) $f(\mathbf{x})$, $\mathbf{x}_0\mathbf{x}_1$ view

(c) $f(\mathbf{x})$, perspective view

(d) $\mathscr{P}_\theta\{f\}(\mathbf{y})$

Fig. 1. Geometry of X-ray projections. (a)-(c) A function of 3D space, f, (grey-green) and an X-ray projection plane spanned by the columns of $\mathbf{P}_{\theta^\perp}^T$ (pink), shown from two orthogonal views and in perspective. (d) The X-ray projection of f onto the pictured plane plotted in the coordinate system specified by $\mathbf{P}_{\theta^\perp}^T$. In this geometry, $\mathbf{x}_2 = \mathbf{y}_1$.

Both of these can be verified via Eq. (1).

In the computed tomography reconstruction problem, we are given a set of measurements, $\mathbf{g}$, called a *sinogram*, that corresponds to samples of $\mathscr{P}_\theta\{f\}(\mathbf{y})$ at known locations and with known projection directions. We aim to discover f from $\mathbf{g}$. If $\mathbf{g}$ has N elements, then we can summarize the measurement process as the operator $H : L_2(\mathbb{R}^d) \to \mathbb{R}^N$ and write $\mathbf{g} = Hf$. The challenges in this problem are that the measurements are corrupted by noise and may cover only a small number of θs. The result of these is that instead of solving for f directly, we need to solve the optimization problem

$$f^* = \arg\min_f \|\mathbf{g} - Hf\| + \Psi(f), \tag{4}$$

where Ψ is a regularization function that encodes the desirable properties of f^*, e.g., piecewise smoothness, nonnegativity, etc.

2.2. *Differential phase contrast X-ray tomography*

Our formulation so far has been for conventional CT, but it easily generalizes to DPC CT. In DPC CT, what is measured is not the phase contrast, but the derivative of the phase contrast

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016

along a direction perpendicular to the projection direction (i.e. along a direction in the projection plane). Thus the DPC X-ray transform of f is

$$\mathscr{D}_{\theta,\mathbf{u}}\{f\}(\mathbf{y}) = \langle \mathbf{u}, \nabla \mathscr{P}_\theta\{f\}(\mathbf{y})\rangle, \tag{5}$$

where $\mathbf{u} \in \mathbb{R}^{d-1}$ is a unit vector that specifies the direction of the derivative and ∇ is the gradient operator. The key fact about this definition is that the DPC X-ray transform is linear and projection translation invariant (Eqs. (2) and (3)), just like the conventional X-ray transform. Therefore our subsequent discussion holds equally for both transforms. We continue with the understanding that each expression can be generalized to DPC CT by replacing $\mathscr{P}_\theta\{f\}$ with $\mathscr{D}_{\theta,\mathbf{u}}\{f\}$ and applying Eq. (5). As the next section will show, doing this will only require knowing the derivative of the X-ray transform of the discretization function used.

3. Fast 3D reconstruction algorithm

We now describe our fast, fully 3D reconstruction algorithm. While the discretization and optimization steps are standard in the literature, the fast algorithms for $\mathbf{H}^T\mathbf{g}$ and $\mathbf{H}^T\mathbf{H}\mathbf{c}$ in Section 3.3 are new.

3.1. Discretization

The immediate problem with Eq. (4) is that it is an optimization over the continuous quantity f; in order to store f, we need a discrete, finite-length representation of it. To this end, we represent f by a sum of shifted kernels:

$$f(\mathbf{x}) = \sum_{\mathbf{k}\in\mathscr{K}} \mathbf{c}[\mathbf{k}]\varphi(\mathbf{x} - \Lambda_\mathbf{x}\mathbf{k}), \tag{6}$$

where $\mathscr{K}$ is a finite set of integer multi-indexes and $\Lambda_\mathbf{x}$ is a $d \times d$ diagonal matrix containing the sampling step in each of d dimensions. This discretization allows the continuous function f to be represented by $\Lambda_\mathbf{x}$, the values $\mathbf{c}[\mathbf{k}]$ for $\mathbf{k} \in \mathscr{K}$, and a formula for φ. Common choices for φ include voxels, splines [30–32], or other *blobs*. Also note that, in practice, we store the values of φ and (at least in the isotropic case) its X-ray projection in lookup tables, thus there is no computational overhead for rigorously discretizing.

Combining Eqs. (1) and (6) along with the linearity and projection translation invariance of the X-ray transform,

$$\mathscr{P}_\theta\{f\}(\mathbf{y}) = \sum_{\mathbf{k}\in\mathscr{K}} \mathbf{c}[\mathbf{k}]\mathscr{P}_\theta\{\varphi\}(y - \mathbf{P}_{\theta^\perp}\Lambda_\mathbf{x}\mathbf{k}). \tag{7}$$

This makes intuitive sense: the X-ray transform of f is just a sum of the X-ray transform of each φ shifted to the correct locations in the projection domain. We let $\mathbf{g}$ be a sampled version of this to complete our discrete forward model,

$$\mathbf{H}\mathbf{c}[\mathbf{m}] = \mathbf{g}[\mathbf{m}] = \mathscr{P}_\theta\{f\}(\Lambda_\mathbf{y}\mathbf{m}) = \sum_{\mathbf{k}\in\mathscr{K}} \mathbf{c}[\mathbf{k}]\mathscr{P}_\theta\{\varphi\}(\Lambda_\mathbf{y}\mathbf{m} - \mathbf{P}_{\theta^\perp}\Lambda_\mathbf{x}\mathbf{k}), \tag{8}$$

where, paralleling Eq. (6), $\mathbf{m} \in \mathscr{M}$ is a finite set of integer multi-indexes, and $\Lambda_\mathbf{y}$ is a $(d-1)\times(d-1)$ diagonal matrix containing the sampling step of the projection domain in each of $d-1$ dimensions. Note that we do not include a detector model (i.e. we have point measurements of the sinogram); this is for simplicity and because Eq. (6) is enough to restrict the sinogram to be bandlimited as will be required later. A detector model can be added without affecting the algorithms we present.

In practice, $\mathbf{g}$ will consist of measurements of the X-ray transform of f from more than one projection direction, $\{\theta_v\}_{v=0}^{V-1}$. For now, we describe our scheme for one θ. The extension to multiple θs is simple due to the linearity of the X-ray transform; we give the corresponding formulas in the pseudocode for the algorithm in Fig. 2.

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016

3.2. *Optimization scheme*

After this discretization, we rewrite Eq. (4) as a discrete optimization problem,

$$\mathbf{c}^* = \arg\min_{\mathbf{c}} \|\mathbf{g} - \mathbf{H}\mathbf{c}\| + \Psi(\mathbf{c}), \tag{9}$$

where $\mathbf{H}$ is defined as in Eq. (8) and Ψ now represents the discrete version of the regularization function.

We approach the optimization problem, Eq. (9), using the alternating direction method of multipliers (ADMM) [33] (now ubiquitous in medical image reconstruction), which gives a recipe for solving difficult optimization problems via solving a sequence of simpler subproblems. Specifically, we separate the regularization term $\Psi(\mathbf{c})$ into its quadratic and non-quadratic parts so that $\Psi(\mathbf{c}) = \Psi_1(\mathbf{c}) + \Psi_2(\mathbf{L}\mathbf{c})$, where $\mathbf{L}$ is the regularization operator. We perform variable splitting to arrive at

$$\begin{aligned} \mathbf{c}^* = \arg\min_{\mathbf{c}} \quad & \|\mathbf{g} - \mathbf{H}\mathbf{c}\| + \lambda_1 \Psi_1(\mathbf{c}) + \lambda_2 \Psi_2(\mathbf{u}), \\ \text{subject to} \quad & \mathbf{u} = \mathbf{L}\mathbf{c}. \end{aligned} \tag{10}$$

We then form the equivalent augmented Lagrangian,

$$\mathscr{L}(\mathbf{c}, \mathbf{u}, \alpha) = \frac{1}{2}\|\mathbf{g} - \mathbf{H}\mathbf{c}\|^2 + \lambda_1 \Psi_1(\mathbf{c}) + \lambda_2 \Psi_2(\mathbf{u}) + \alpha^T(\mathbf{L}\mathbf{c} - \mathbf{u}) + \frac{\mu}{2}\|\mathbf{L}\mathbf{c} - \mathbf{u}\|^2, \tag{11}$$

and solve via the cyclic update

$$\mathbf{c}^{k+1} \leftarrow \arg\min_{\mathbf{c}} \mathscr{L}(\mathbf{c}, \mathbf{u}^k, \alpha^k) \tag{12}$$

$$\mathbf{u}^{k+1} \leftarrow \arg\min_{\mathbf{u}} \mathscr{L}(\mathbf{c}^{k+1}, \mathbf{u}, \alpha^k) \tag{13}$$

$$\alpha^{k+1} \leftarrow \alpha^k + \mu(\mathbf{L}\mathbf{c}^{k+1} - \mathbf{u}^{k+1}). \tag{14}$$

Update Eq. (12) is a quadratic optimization in $\mathbf{c}$ with gradient

$$\nabla \mathscr{L}(\mathbf{c}, \mathbf{u}^k, \alpha^k) = (\mathbf{H}^T\mathbf{H} + \mu\mathbf{L}^T\mathbf{L} + \lambda_1 \nabla\Psi_1)\mathbf{c} - \left(\mathbf{H}^T\mathbf{g} + \mu\mathbf{L}^T\left(\mathbf{u}^k - \frac{\alpha^k}{\mu}\right)\right). \tag{15}$$

We solve it iteratively using the conjugate gradient algorithm [34]. Update Eq. (13) is solved using the proximal map associated with Ψ_2, which is known for the popular regularizers including total variation (TV).

3.3. *Fast algorithms*

The bottleneck of our optimization scheme is Eq. (12), and specifically the computation of $\mathbf{H}^T\mathbf{H}\mathbf{c}$ and $\mathbf{H}^T\mathbf{g}$ in the conjugate gradient step, Eq. (15). The computation of $\mathbf{H}^T\mathbf{H}\mathbf{c}$ is especially important as it happens once per iteration of the conjugate gradient. The value of $\mathbf{H}^T\mathbf{g}$ can be computed once and stored, but the straightforward implementation remains prohibitively slow (on the order of days for 3D reconstructions with a large number of projections on a consumer-level laptop).

In the following sections, we describe our algorithms for these two computations, which extends the approach of [29] to 3D. In both cases, we uncover a discrete convolution and compute it with the FFT. Though we do use the Fourier transform and lookup/interpolation, our method is distinct from the class of regridding techniques (used both in CT [28] and widely in MRI [35]) which rely on the Fourier slice theorem and interpolation in the Fourier domain. We provide code for these methods on our website, http://bigwww.epfl.ch/algorithms.html.

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016

3.3.1. $\mathbf{H}^T\mathbf{g}$

We write the space-domain version of $\mathbf{H}^T\mathbf{g}$ using the properties of the ℓ_2 inner product,

$$\begin{aligned}\left\langle \mathbf{H}^T\mathbf{g}, \mathbf{c} \right\rangle &= \langle \mathbf{g}, \mathbf{H}\mathbf{c} \rangle && (16)\\ &= \sum_{\mathbf{m}\in\mathscr{M}} \mathbf{g}[\mathbf{m}] \sum_{\mathbf{k}\in\mathscr{K}} \mathbf{c}[\mathbf{k}]\, \mathscr{P}_\theta\{\varphi\}(\Lambda_\mathbf{y}\mathbf{m} - \mathbf{P}_{\theta^\perp}\Lambda_\mathbf{x}\mathbf{k}) && (17)\\ &= \sum_{\mathbf{k}\in\mathscr{K}} \mathbf{c}[\mathbf{k}] \sum_{\mathbf{m}\in\mathscr{M}} \mathbf{g}[\mathbf{m}]\, \mathscr{P}_\theta\{\varphi\}(\Lambda_\mathbf{y}\mathbf{m} - \mathbf{P}_{\theta^\perp}\Lambda_\mathbf{x}\mathbf{k}) && (18)\end{aligned}$$

and thus

$$(\mathbf{H}^T\mathbf{g})[\mathbf{k}] = \sum_{\mathbf{m}\in\mathscr{M}} \mathbf{g}[\mathbf{m}]\, \mathscr{P}_\theta\{\varphi\}(\Lambda_\mathbf{y}\mathbf{m} - \mathbf{P}_{\theta^\perp}\Lambda_\mathbf{x}\mathbf{k}). \quad (19)$$

Computing $\mathbf{H}^T\mathbf{g}$ via Eq. (19) is slow. To compute $\mathbf{H}^T\mathbf{g}$ on a $K\times K\times K$ cubic grid, ($\mathbf{k}\in\mathscr{K} = \mathbb{Z}_K^3$) with measurements from an $M\times M$ square grid ($\mathbf{m}\in\mathscr{M} = \mathbb{Z}_M^2$) the computation requires $O(K^3M^2)$ operations per view. If $\mathscr{P}_\theta\{\varphi\}$ has non-zero support with width $W < M$, then the cost is reduced to $O(K^3W^2)$. We can, instead, approximate Eq. (19) using a lookup table for its entire right hand side. The lookup table is generated by a discrete convolution of $\mathbf{g}$ (optionally upsampled) and a sampled version of $\mathscr{P}_\theta\{\varphi\}$, which can be computed (either via FFT or in space) on any grid that has a step size that is an integer fraction of $\Lambda_\mathbf{y}$. The computation then requires $O(M^2\log M) + O(K^3)$ operations for the convolution and the lookups, respectively.

For comparison, the regridding [28] approach to the same problem uses $O(M^2\log M) + O(W^3M^2)$ (where W is the width of the kernel used in the Fourier domain) per view, plus a 3D FFT, $O(K^3\log K)$. Whether our convolution-based approach is faster than gridding depends on the specific implementations used and number of views (gridding is more advantageous when the number of views is large because the cost of the 3D FFT is amortized). In practice, our method is of comparable speed (or slightly faster) with the advantage of simple implementation and parameter selection. It also offers easy parallelization, either across the views or the locations $\mathbf{k}\in\mathscr{K}$. Finally, it works for DPC simply by replacing $\mathscr{P}_\theta\{\varphi\}$ with $\mathscr{D}_{\theta,\mathbf{u}}\{\varphi\}$, while it is not immediately clear how to extend regridding to DPC.

3.3.2. $\mathbf{H}^T\mathbf{H}\mathbf{c}$

We now turn our attention to computing $\mathbf{H}^T\mathbf{H}\mathbf{c}$. To arrive at the space version, we combine Eqs. (8) and (19), giving

$$(\mathbf{H}^T\mathbf{H}\mathbf{c})[\mathbf{k}] = \sum_{\mathbf{m}\in\mathscr{M}} \sum_{\mathbf{k}'\in\mathscr{K}} \mathbf{c}[\mathbf{k}']\, \mathscr{P}_\theta\{\varphi\}(\Lambda_\mathbf{y}\mathbf{m} - \mathbf{P}_{\theta^\perp}\Lambda_\mathbf{x}\mathbf{k}')\, \mathscr{P}_\theta\{\varphi\}(\Lambda_\mathbf{y}\mathbf{m} - \mathbf{P}_{\theta^\perp}\Lambda_\mathbf{x}\mathbf{k}). \quad (20)$$

Computing $(\mathbf{H}^T\mathbf{H}\mathbf{c})$ using Eq. (20) directly is slow. For $\mathbf{k}\in\mathbb{Z}_K^3$ and $\mathbf{m}\in\mathbb{Z}_M^2$, the computation requires $O(K^3M^2)$ operations per view (the sum of the number of operations in Eqs. (8) and (19)). And, as before, this cost is reduced to $O(K^3W^2)$ if $\mathscr{P}_\theta\{\varphi\}$ has non-zero support with width $W < M$.

Instead of using (20) directly, we manipulate it to uncover a discrete convolution. We first interchange the sums, giving

$$(\mathbf{H}^T\mathbf{H}\mathbf{c})[\mathbf{k}] = \sum_{\mathbf{k}'\in\mathscr{K}} \mathbf{c}[\mathbf{k}'] \sum_{\mathbf{m}\in\mathscr{M}} \mathscr{P}_\theta\{\varphi\}(\Lambda_\mathbf{y}\mathbf{m} - \mathbf{P}_{\theta^\perp}\Lambda_\mathbf{x}\mathbf{k}')\, \mathscr{P}_\theta\{\varphi\}(\Lambda_\mathbf{y}\mathbf{m} - \mathbf{P}_{\theta^\perp}\Lambda_\mathbf{x}\mathbf{k}). \quad (21)$$

We note that for bandlimited functions f and g, we can show using the sampling theorem [36] that

$$\sum_{\mathbf{m}\in\mathbb{Z}^d} f(\Lambda\mathbf{m})g(\Lambda\mathbf{m}) = \frac{1}{\det(\Lambda)} \int_{\mathbb{R}^d} f(\mathbf{x})g(\mathbf{x})d\mathbf{x}, \quad (22)$$

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016

so long as the sampling rate specified by Λ is higher than the Nyquist rate of f and g. Using this fact, we have

$$(21) \overset{(a)}{=} \sum_{\mathbf{k}' \in \mathscr{K}} \frac{\mathbf{c}[\mathbf{k}']}{\det(\Lambda_{\mathbf{y}})} \int_{\mathbb{R}^{d-1}} \mathscr{P}_\theta\{\varphi\}(\mathbf{y} - \mathbf{P}_{\theta^\perp}\Lambda_{\mathbf{x}}\mathbf{k}') \mathscr{P}_\theta\{\varphi\}(\mathbf{y} - \mathbf{P}_{\theta^\perp}\Lambda_{\mathbf{x}}\mathbf{k}) d\mathbf{y} \tag{23}$$

$$\overset{(b)}{=} \sum_{\mathbf{k}' \in \mathscr{K}} \frac{\mathbf{c}[\mathbf{k}']}{\det(\Lambda_{\mathbf{y}})} \int_{\mathbb{R}^{d-1}} \mathscr{P}_\theta\{\varphi\}(\mathbf{y}' - \mathbf{P}_{\theta^\perp}\Lambda_{\mathbf{x}}(\mathbf{k}' - \mathbf{k})) \mathscr{P}_\theta\{\varphi\}(\mathbf{y}') d\mathbf{y}' \tag{24}$$

$$= \sum_{\mathbf{k}' \in \mathscr{K}} \mathbf{c}[\mathbf{k}'] (\mathscr{P}_\theta\{\varphi\} * \mathscr{P}_\theta\{\varphi\}(-\cdot))(\mathbf{P}_{\theta^\perp}\Lambda_{\mathbf{x}}(\mathbf{k}' - \mathbf{k})) \tag{25}$$

$$\overset{(c)}{=} \frac{1}{\det(\Lambda_{\mathbf{y}})} (\mathbf{c} * \mathbf{r})[\mathbf{k}] \tag{26}$$

where (a) holds only when $\mathscr{P}_\theta\{\varphi\}$ is bandlimited and the sampling rate specified by $\Lambda_{\mathbf{y}}$ is higher than the corresponding Nyquist rate; (b) results from the change of variables $\mathbf{y}' = \mathbf{y} - \mathbf{P}_{\theta^\perp}\Lambda_{\mathbf{x}}\mathbf{k}$;

$$\mathbf{r}[\mathbf{k}] = (\mathscr{P}_\theta\{\varphi\} * \mathscr{P}_\theta\{\varphi\}(-\cdot))(\mathbf{P}_{\theta^\perp}\Lambda_{\mathbf{x}}\mathbf{k}); \tag{27}$$

and (c) also relies on $\mathscr{P}_\theta\{\varphi\}$ being bandlimited.

Equation (26) expresses $\mathbf{H}^T\mathbf{H}\mathbf{c}$ as a discrete convolution between the current discretization values, $\mathbf{c}$, and a discrete kernel, $\mathbf{r}$, which is a sampled version of the autocorrelation of $\mathscr{P}_\theta\{\varphi\}$. Thus, using the FFT, computing $\mathbf{H}^T\mathbf{H}\mathbf{c}$ requires $O(K^3 \log K)$ operations, plus the one-time cost of computing $\mathbf{r}[\mathbf{k}]$ for $\mathbf{k} \in \mathscr{K}$. Depending on the choice of φ, we may have an easy-to-compute analytical expression for each $\mathbf{r}[\mathbf{k}]$, resulting in $O(K^3)$ operations. If not, we can approximate $\mathbf{r}$ by computing a discrete autocorrelation of a finely sampled version of $\mathscr{P}_\theta\{\varphi\}$ and then interpolating to find the necessary values, resulting in $O(M^2 \log M) + O(K^3)$ operations for the autocorrelation and interpolation, respectively.

The straightforward way to compute $\mathbf{H}^T\mathbf{H}$ with regridding would require double the time of $\mathbf{H}^T$, so again $O(M^2 \log M) + O(W^3 M^2)$ per view, plus two 3D FFTs, $O(K^3 \log K)$. Thus, the cost of computing the kernel in our method is again comparable in speed to regridding, with the faster method determined by implementation and the specifics of the problem. But, once the kernel is computed, the proposed method offers a clear improvement in runtime over regridding, because it requires only a 3D filtering operation for each subsequent computation.

3.4. Algorithm summary

We now combine the above results and present the complete proposed algorithm, including more implementation details and pseudocode (Fig. 2).

The input of the algorithm is a set of V sampled X-ray projections (or *views*), $\{\mathbf{g}_v\}_{v=0}^{V-1}$ and the description of the physical setup of those projections, comprising the corresponding projection matrices, $\{\mathbf{P}_{\theta_v^\perp}\}$ and the sampling grid, $\mathscr{M}$ and $\Lambda_{\mathbf{y}}$.

There are three sets of parameters for the algorithm:

1. The reconstruction grid, $\mathscr{K}$ and $\Lambda_{\mathbf{x}}$. We typically set $\mathscr{K}$ and $\Lambda_{\mathbf{x}}$ so that $\{\Lambda_{\mathbf{x}}\mathbf{m} : \mathbf{m} \in \mathscr{K}\}$ is an isotropic grid with dimensions ranging from $80 \times 80 \times 80$ to $300 \times 300 \times 300$, that covers approximately the same area in space as the input projections, $\{\mathbf{g}_v\}$. Note that our discretization scheme has the advantage of decoupling the size and resolution of the reconstruction from the size and resolution of the input, which is useful in practice.

2. The discretization function φ. In this work, this is fixed to be an isotropic generalized Kaiser-Bessel window (KBW) [37] with $m = 2$, $\alpha = 10.45$, and a spatial support, a,

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016

procedure RECONSTRUCT($\{\mathbf{g}_v\}, \{\mathbf{P}_{\theta_v^\perp}\}, M, \Lambda_\mathbf{y}$)
  for all views **do**
    compute $\mathbf{H}^T\mathbf{g}_v$ ▷ via Eq. (19)
  end for
  $\mathbf{H}^T\mathbf{g} \leftarrow \sum_v \mathbf{H}^T\mathbf{g}_v$

  for all views **do**
    compute $\mathbf{r}_v$ ▷ via Eq. (27)
  end for
  $\mathbf{r} \leftarrow \sum_v \mathbf{r}_v$

  while not converged **do**
    $\mathbf{c} \leftarrow$ **ADMM update** ▷ via Eqs. (12)-(14) and with $\mathbf{H}^T\mathbf{H}\mathbf{c}$ via Eq. (26)
  end while
  return c
end procedure

Fig. 2. High-level description of the proposed algorithm.

equal to 4 times the sampling step in space. (We pick these values based on our analysis in [38], which shows that they improve the approximation properties of the kernel. We choose the KBW window over, e.g., splines, because using an isotropic window gives a computational advantage and because in the high noise/low views regime we expect approximation errors to be negligible compared to other sources of error.) One subtlety of our algorithm is that our adjoint calculation Eq. (19) is fastest when φ has small spatial support, while our kernelization Eq. (26) only holds with equality when φ is bandlimited. Our choice of φ thus means Eq. (26) is an approximation, but it is highly accurate because φ is nearly bandlimited.

3. The optimization parameters from Eq. (11): λ_1, Ψ_1, λ_2, Ψ_2, α, and μ. The choice of the regularization functions Ψ_1 and Ψ_2 is application dependent, but here we use the popular total variation regularization, which corresponds to $\Psi_1(\mathbf{c}) = 0$ and $\Psi_2(L\mathbf{c}) = \|L\mathbf{c}\|_1$, where $L\mathbf{c}[\mathbf{k}]$ computes the derivative of $f(\Lambda_\mathbf{x}\mathbf{k})$ in each of the spatial dimensions. The parameters α and μ can be reduced to a single parameter by writing the ADMM in the scaled form [33] and only affect the convergence speed of the optimization; we therefore fix this value and use enough ADMM iterations that all our reconstructions converge satisfactorily. Finally, λ_2 controls the strength of the regularization relative to the data fit term and we set it via a parameter sweep.

The output of the algorithm is $\mathbf{c}$, which is the set of coefficients used in the discretization Eq. (6). For display purposes, it is necessary to compute the values of $f(x)$ at discrete locations; we choose to use the same locations as the reconstruction grid, i.e. $\{\Lambda_\mathbf{x}\mathbf{m} : \mathbf{m} \in \mathscr{K}\}$.

4. Experiments

We now present our experiments. In the first set of experiments, we examine the speed and accuracy of our fast algorithms for $\mathbf{H}^T\mathbf{g}$ and $\mathbf{H}^T\mathbf{H}\mathbf{c}$ as compared to the exact implementation of the same. In the second set of experiments, we compare our proposed fully 3D reconstruction algorithm to a slice-by-slice algorithm on a digital phantom, a physical phantom, and a biological specimen.

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016

4.1. Datasets

We use three datasets to evaluate the proposed reconstruction algorithm:

1. The *Shepp3D* dataset comes from an analytical phantom comprised of ellipsoids. Specifically, we use nine of the ellipsoids from the 3D version of the Shepp-Logan phantom [39]: the ten corresponding to the original 2D Shepp-Logan phantom minus the skull, shown in Fig. 1. We select these for their familiarity and ease of visualization. We generate a sinogram, g, from this phantom by computing Eq. (1) analytically on a 601 $\times$ 401 grid for 1,201 angles spaced evenly from 0 to π, inclusive. We then add zero-mean Gaussian noise to the sinogram so that the signal-to-noise ratio (SNR, defined as $10\log_{10}(\sum_{\mathbf{m}}|\mathbf{g}[\mathbf{m}]|^2/\sum_{\mathbf{m}}|\mathbf{n}[\mathbf{m}]|^2)$ dB, where $\mathbf{g}$ is the signal and $\mathbf{n}$ is the noise) is -10dB. The resulting measurements have values in the range [-0.0164, 0.0120] and the noise has a standard deviation of 0.0106. The Shepp3D phantom is piecewise constant and therefore a good fit for TV regularization.

2. The *phantom* dataset comes from a physical phantom composed of a tube and three cylinders containing liquids with different refractive indices. Each view has dimensions 1642 $\times$ 537, and the views were collected for 1,1201 angles spaced evenly from 0 to π, inclusive. For this dataset, we have both conventional and DPC sinograms.

3. The *rat brain* dataset comes from a rat brain embedded in liquid paraffin at room temperature. Each view has dimensions 1493 $\times$ 377, and the views were collected for 721 angles spaced evenly from 0 to π, inclusive. For this dataset, we have both conventional and DPC sinograms.

In all three cases, the X-ray projections are collected around a fixed axis, as shown in Fig. 1. Specifically, the $\mathbf{y}_1$ axis is the same as the $\mathbf{x}_2$ (or, equivalently, the second column of $\mathbf{P}_{\theta^\perp}^T$ is $[0\,0\,1]^T$.) As a result, each projection can be described by a single angle rather than three, as would be the case in general in 3D. We use fixed-axis projections to facilitate comparison between our fully 3D approach and the slice-by-slice approach, but we stress that our formulation and algorithm apply to all parallel-ray geometries.

4.2. Speed and accuracy of proposed fast algorithms

4.2.1. $\mathbf{H}^T$

To assess the speed and accuracy of our algorithm for $\mathbf{H}^T\mathbf{g}$, we use the Shepp3D dataset and calculate $\mathbf{H}^T\mathbf{g}$ using two methods: (1) The *exact* method implements Eq. (19) by looping over $\mathbf{k}$ and $\mathbf{m}$ using a precomputed lookup table for evaluating $\mathscr{P}_\theta\{\varphi\}(y)$. It is written in C and compiled into a MATLAB MEX file. Because the lookup table used is 1D and can therefore be very fine, we take this method to be the ground truth for computing $\mathbf{H}^T\mathbf{g}$. (2) The *fast* method computes a lookup table for the entire right hand side of Eq. (19) using discrete convolution as we proposed in Section 3.3.1. It is written in MATLAB. The upsampling rate of the grid used for this convolution is a parameter of the method, with an upsampling rate of u meaning the lookup table is computed on a grid with step $\Lambda_{\mathbf{y}}/u$. For both methods, the width of the spatial support of φ was 7 pixels of the reconstruction grid.

We compare these methods across several problem sizes, where a problem size of $K\times K\times K$ corresponds to a reconstruction grid of size $K\times K\times K$ and a sinogram of size $4K\times 4K$ with 101 views. At each problem size, we measure the time each method takes and the SNR ($10\log_{10}(\sum_{\mathbf{m}}|\mathbf{g}[\mathbf{m}]|^2/\sum_{\mathbf{m}}|\hat{\mathbf{g}}[\mathbf{m}]|^2)$ dB, where $\mathbf{g}$ is the exact adjoint and $\hat{\mathbf{g}}$ is the estimate). These and all other times we report come from running the code on a single CPU without parallelization.

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016

The results show that our fast algorithm is highly accurate, achieving over 70dB SNR compared to the exact method when the upsampling rate is 2 (Fig. 3 left). Our method is also an order of magnitude or more faster than the exact method (Fig. 3 right). The asymptotic analysis in Section 3.3.1 suggests the exact method should grow like K^3 (because W is fixed) and the fast method should grow like $K^2 \log K$ or K^3, depending on whether the lookup or convolution dominates. The experiment roughly supports this analysis and suggests that the lookup in the fast method is negligible here. Comparison to the exact method on larger problems is impractical because it is prohibitively slow, but for reference we note that our fast algorithm can compute $\mathbf{H}^T\mathbf{g}$ on a grid of size $401 \times 401 \times 401$ from a sinogram of size 801×801 with 101 views with output in approximately 2 minutes.

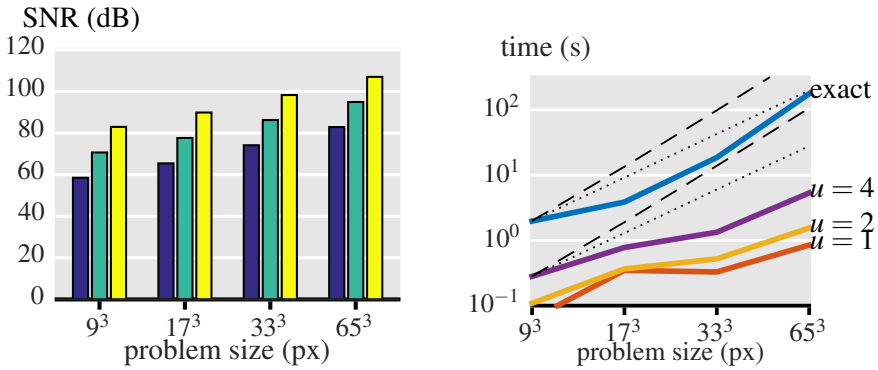


Fig. 3. Accuracy (left) and computation time (right) vs problem size for our fast algorithm for $\mathbf{H}^T\mathbf{g}$ with upsamping rates of 1, 2, and 4 (left to right bars). On the right, $K^3 + C$ (dashed) and $K^2 \log K + C$ (dotted) are plotted for comparison. Our method is a highly accurate and consistently faster than the exact implementation.

4.2.2. $\mathbf{H}^T\mathbf{H}\mathbf{c}$

We perform a similar experiment for $\mathbf{H}^T\mathbf{H}\mathbf{c}$: Using a random initialization for $\mathbf{c}$, we calculate $\mathbf{H}^T\mathbf{H}\mathbf{c}$ using two methods: both an exact implementation of and the fast implementation Eq. (26). (1) The *exact* method computes Eq. (20) by looping over $\mathbf{m}$ and $\mathbf{k}'$ followed by $\mathbf{m}$ and $\mathbf{k}$, again using a precomputed lookup table for evaluating $\mathscr{P}_\theta\{\varphi\}(y)$, and again written in C and compiled into a MATLAB MEX file. We take this implementation to be the ground truth. (2) The *fast* method computes the kernel, $\mathbf{r}$, via a discrete convolution, Eq. (27), and interpolation, then uses it to compute $\mathbf{H}^T\mathbf{H}\mathbf{c}$ via Eq. (26). It is written in MATLAB with a MEX file to handle the interpolation. The upsampling rate for the calculation of $\mathbf{r}$ is a parameter of the method.

We compare these methods across several problem sizes, where a problem size of $K \times K \times K$ corresponds to a reconstruction grid of size $K \times K \times K$ and a sinogram of size $5K \times 5K$ with 101 views.

Again, the results show that our algorithm is highly accurate, achieving greater than 70 dB SNR even with no upsampling (Fig. 4 left). It is also much faster and scales better than the exact implementation (Fig. 4 right). Again, the asymptotic analysis in Section 3.3.1 suggests the exact method should grow like K^3 and the fast method should grow like $K^2 \log K$ or K^3, depending on whether the lookup or convolution dominates. The experiment supports this analysis for the exact method, but suggests for the fast method that runtime is dominated by overhead, which is not surprising given the computations last for less than a second and involve MATLAB.

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016

The speed difference is even more pronounced in practice, since our optimization algorithm requires repeated application of $\mathbf{H}^T\mathbf{H}$ and the fast method computes a kernel once and reuses it for each computation. As is the case with the adjoint, comparison on larger data is impractical due to the poor scaling of the exact method. Again, we note for reference that computing the kernel, $\mathbf{r}$ in Eq. (27), with $\mathbf{c}$ of size $401 \times 401 \times 401$, and a sinogram of size 801×801 with 101 views takes approximately 1 minute and thereafter a single computation of $\mathbf{H}^T\mathbf{H}\mathbf{c}$ takes approximately 17 seconds.

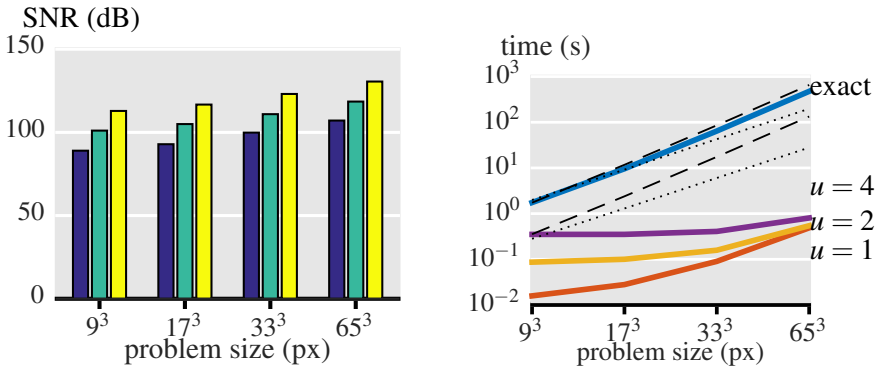


Fig. 4. Accuracy (left) and computation time (right) as functions of input/output size for our fast algorithm for $\mathbf{H}^T\mathbf{H}\mathbf{c}$ with upsamping rates of 1, 2, and 4. On the right, $K^3 + C$ (dashed) and $K^2 \log K + C$ (dotted) are plotted for comparison. Our method is highly accurate and is much faster and scales better than the exact implementation.

4.3. Fully 3D versus slice-by-slice reconstruction

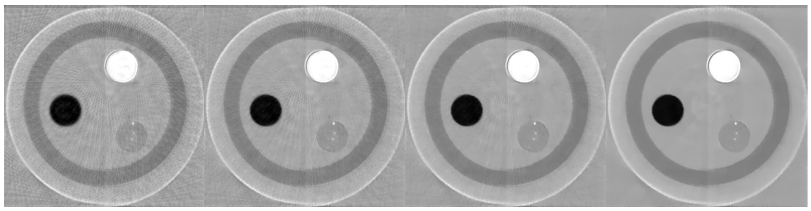
We now compare the fully 3D reconstruction algorithm presented above with a slice-by-slice version of the same.

To perform a slice-by-slice reconstruction, we set the kernel in Eq. (6) to be an isotropic KBW in the $\mathbf{x}_0\mathbf{x}_1$ plane multiplied by a box in the $\mathbf{x}_2$ direction. For each slice of the reconstruction, we average the corresponding measurements in the sinogram along the $\mathbf{y}_1$ direction, creating a new, one dimensional sinogram, which we use to compute a 2D reconstruction. This 2D reconstruction uses the algorithm described in Section 3 with dimension $d = 2$. Once the 2D reconstructions are complete, we concatenate the resulting 2D $\mathbf{c}$s along the $\mathbf{x}_2$ direction to complete the reconstruction. The main differences between the slice-by-slice and fully 3D methods are therefore that (1) the regularization in the slice-by-slice reconstruction is computed only within and not between slices and (2) the discretization kernel in the slice-by-slice reconstruction is anisotropic.

4.3.1. Shepp3D dataset comparison

We first compare the slice-by-slice and fully 3D methods using the Shepp3D dataset. To evaluate the robustness of the methods to missing projection angles, we create downsampled versions of the sinogram containing 1201, 301, 101, 51, 26, and 13 projections. For each of these sinograms and for each method, we compute a $161 \times 161 \times 101$ reconstruction for a range of 8 different regularization strengths, $\lambda_2 = 0, a2^1, a2^2, \ldots, a2^7$ with a chosen experimentally to cover the range between too little and too much regularization (see Fig. 5 for an example). We compute an SNR for each of these reconstructions using the analytical values of the phantom as the oracle and report the SNR for the unregularized and the best of the regularized reconstruc-

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016



(a) 9.60 dB (b) 11.19 dB (c) 12.42 dB (d) 13.82 dB

(e) 14.32 dB (f) 13.19 dB (g) 11.96 dB (h) 10.99 dB

Fig. 5. Example parameter sweep from low (a) to high (h) regularization strength using the physical phantom dataset.

tions. When using all 1201 views, these reconstructions take approximately 40 minutes for the slice-by-slice method and 58 minutes (7 min for $\mathbf{H}^T\mathbf{g}$, 5 min for $\mathbf{r}$, and 100×27 s $= 45$ min) for the fully 3D method.

Figure 6 shows a plot of SNR versus number of views for both methods with and without regularization. In general, decreasing the number of views reduces the SNR. Regularization greatly improves the SNR of both methods; note that this improvement comes both from noise removal and from artifact removal as the number of views decreases, see Fig. 7 for examples.

When the number of views is high, the regularized methods perform similarly well, while in the unregularized case, the fully 3D method gives an improvement of 3.5dB over the slice-by-slice. This underscores that the difference between the methods is not only the 3D regularization, but also that they use different discretization kernels. In this example, the isotropic discretization kernel of the fully 3D method offers beneficial smoothing that helps remove noise in the absence of regularization.

As the number of views decreases, the gap between the regularized methods increases, which suggests that the fully 3D regularization better handles low-view artifacts. Qualitatively, the difference is most apparent in the $\mathbf{x}_1\mathbf{x}_2$ cross section (Fig. 8), where the slice-by-slice method shows significant variability between slices. (We also note the slight improvement of the unregularized methods at low numbers of views. At such low SNR, we don't believe these differences are meaningful; qualitatively all the low-view unregularized reconstructions are equally bad.)

4.3.2. Phantom dataset comparison

We repeat the comparison using the phantom dataset. Again, we create (angularly) downsampled versions of the sinogram containing 1201, 301, 101, 51, 26, and 13 projections and perform a parameter sweep over 8 regularization strengths. The phantom dataset contains both absorption and DPC measurements, so we perform reconstructions for each separately. The output reconstructions were $417 \times 417 \times 135$ and took approximately 180 minutes for the slice-by-slice method and 300 minutes (35 min for $\mathbf{H}^T\mathbf{g}$, 31 min for $\mathbf{r}$, and 100×145 s $= 240$ min) for the fully 3D method when using all 1201 views.

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016

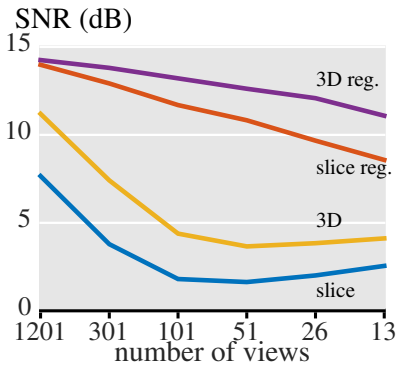
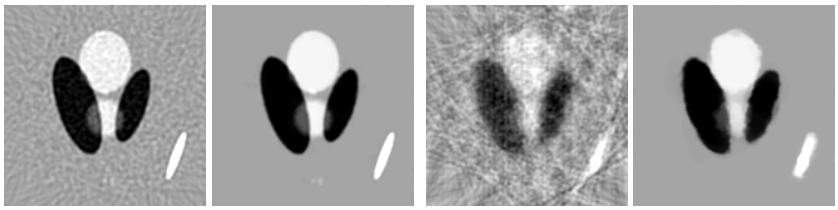


Fig. 6. Reconstruction quality versus number of views for the slice-by-slice and fully 3D reconstruction methods. Reducing the number of views negatively impacts both methods, but regularization mitigates the effect.



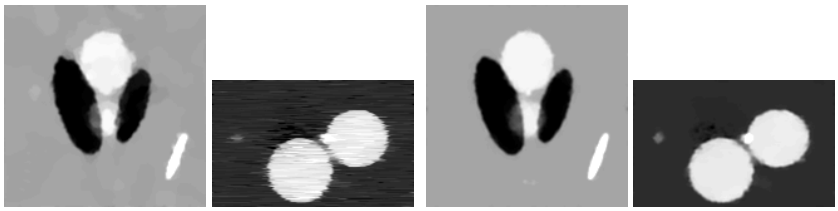
(a) 1201 views, effect of regularization (b) 13 views, effect of regularization

Fig. 7. Representative slices of the fully 3D reconstruction of the Shepp3D dataset. (a) When the number of views is large, regularization only reduces noise in the reconstruction resulting from noise in the sinogram. (b) When the number of views is small, regularization also reduces the consequent line artifacts.

Because the phantom dataset comes from a physical phantom, we do not have access to a ground truth for the reconstruction, as we did in the case of the analytical Shepp3D phantom. To solve this problem, we compute the SNR for each reconstruction using the fully 3D reconstruction with 1,201 views and no regularization as the oracle. We take this value as a measure of the stability of the methods with respect to reduced number of views rather than as ironclad measure of reconstruction quality. In the absence of a known ground truth, we argue that the best measure of the reconstruction quality is ultimately the qualitative, visual differences between the reconstructions.

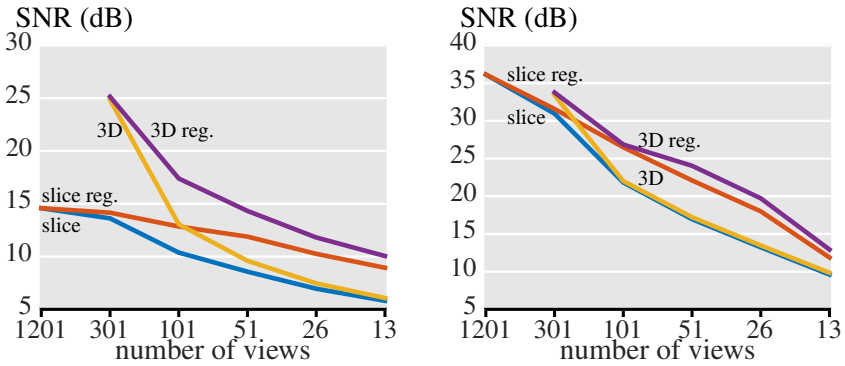
Quantitative results for this experiment are shown in Fig. 9. The same trend from the digital phantom applies here: as the number of views decreases, so does the quality of the reconstruction. Again, regularizing the reconstruction reduces this effect; this is intuitive since the physical phantom is approximately piecewise constant. And again, the fully 3D reconstruction provides a higher SNR than the slice-by-slice reconstruction for each number of views. We provide qualitative illustration of this in Figs. 10 and 11. The two reconstruction methods are similar when viewing $\mathbf{x}_0\mathbf{x}_1$ cross sections, but the $\mathbf{x}_1\mathbf{x}_2$ cross sections reveal that the 3D method enforces better consistency between the slices than the slice-by-slice method.

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016



(a) slice-by-slice (b) fully 3d

Fig. 8. Representative slices of the slice-by-slice and fully 3D reconstruction of the Shepp3D dataset from 101 views with regularization. The fully 3D achieves superior results by enforcing consistency in all three dimensions.



(a) CT (b) DPCI

Fig. 9. Phantom dataset results.

4.3.3. Rat brain dataset comparison

Finally, we compare the slice-by-slice and fully 3D methods using the rat brain dataset. We use the same experimental setup as for the phantom dataset, except with the number of views being 721, 181, 61, 31, 16, and 8, and the reconstructions having size $381 \times 381 \times 95$. Each reconstruction took approximately 205 minutes for the slice-by-slice method and 375 minutes (14 min for $\mathbf{H}^T\mathbf{g}$, 12 min for $\mathbf{r}$, and 100×209 s $=$ 348 min) for the fully 3D method when using all 721 views.

The results (Fig. 12) are similar to the phantom experiment: the fully 3D method consistently outperforms the slice-by-slice method, indicating that the fully 3D method provides a more consistent reconstruction as the number of views decreases, but this difference has a smaller magnitude. Qualitatively, there is not much difference between the slice-by-slice and fully 3D reconstruction (Fig. 13). We hypothesize that this is because the TV regularization is relatively less effective on the rat brain because it is more complex (less piecewise constant) than the other datasets and because (compared to the Shepp3D dataset) it has a small amount of measurement noise. We believe these results could be improved by taking measurements with a smaller pixel size (thereby making the reconstruction more piecewise constant) or by trying another regularizer, e.g., the Hessian Schatten-norm [25], wavelets [40], curvelets [41], or dictionary learning [42].

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016

(a) 1201 views, $\mathbf{x}_0\mathbf{x}_1$ cross section (b) 13 views, $\mathbf{x}_0\mathbf{x}_1$ cross section (c) 1201 views, $\mathbf{x}_1\mathbf{x}_2$ cross section (d) 13 views, $\mathbf{x}_1\mathbf{x}_2$ cross section

Fig. 10. Representative slices of the slice-by-slice reconstruction of the phantom dataset with regularization. When the number of views is low, there is an inconsistency between the slices apparent in the $\mathbf{x}_1\mathbf{x}_2$ cross section (d).

(a) 1201 views, $\mathbf{x}_0\mathbf{x}_1$ cross section (b) 13 views, $\mathbf{x}_0\mathbf{x}_1$ cross section (c) 1201 views, $\mathbf{x}_1\mathbf{x}_2$ cross section (d) 13 views, $\mathbf{x}_1\mathbf{x}_2$ cross section

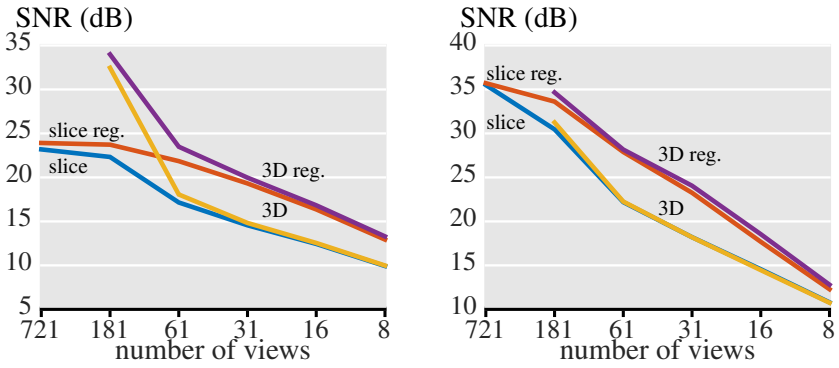
Fig. 11. Representative slices of the fully 3D reconstruction of the phantom dataset with regularization. The reconstruction is of reasonable quality even with only 13 views.

5. Conclusions

We have presented an algorithm for the X-Ray tomography reconstruction problem. Our mathematical formulation includes a generalized discretization step and applies equally to conventional CT and DPC in any number of dimensions. We provide fast, accurate algorithms for computing the X-ray adjoint and normal operators which allow large ($417 \times 417 \times 135$ with 1,201 views) reconstructions to finish in under six hours in MATLAB. We stress that, while our formulation was developed with fully 3D reconstruction in mind, it can also be used to describe the more familiar slice-by-slice approach to reconstruction. In both cases, the speed of the reconstruction benefits from our fast algorithms.

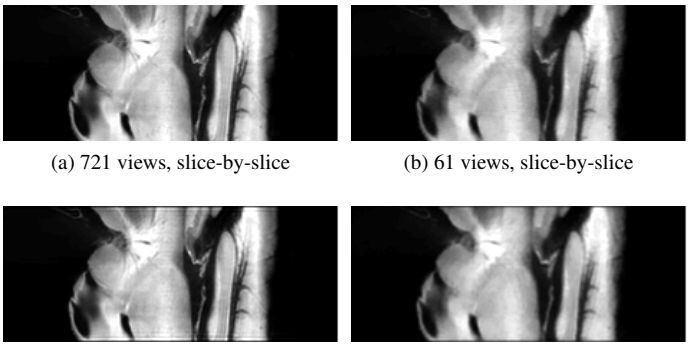
In our experiments, we compare slice-by-slice and fully 3D reconstruction. Results on a piecewise constant analytical phantom show that fully 3D reconstruction can provide more accurate reconstructions with fewer views than the slice-by-slice method. This difference comes from the fact that the fully 3D regularization uses information from vertically adjacent slices to reduce noise, while the slice-by-slice regularization does not. Results on a physical phantom show the same trend. Results on a biological sample did not show a clear qualitative difference between the methods, which we attribute both to the increased complexity of the sample and to the relatively small amount of noise in the measurements (as compared with our analytical phantom dataset). From this, we conclude that the fully 3D reconstruction provides superior

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016



(a) CT (b) DPCI

Fig. 12. Rat brain dataset results.



(a) 721 views, slice-by-slice (b) 61 views, slice-by-slice

(c) 721 views, fully 3D (d) 61 views, fully 3D

Fig. 13. Representative $\mathbf{x}_1\mathbf{x}_2$ slices of the slice-by-slice (top row) and fully 3D (bottom row) reconstructions of the rat brain DPC dataset. For this complex sample with low-noise measurements, regularizing in full 3D does not visually improve the reconstruction quality.

reconstructions when noise is high and the number of views is low, which corresponds to the low X-ray dose regime. When the desired reconstruction will not fit in memory, or when the reconstruction cannot benefit from fully 3D regularization, slice-by-slice reconstruction remains the better alternative. In either case, our formulation and fast algorithms provide a principled and efficient way to solve the reconstruction problem.

Acknowledgments

This work was funded (in part) by the Center for Biomedical Imaging of the Geneva-Lausanne Universities and EPFL, as well as by the Foundations Leenaards and Louis-Jeantet and ERC grant agreement No 267439 - FUN-SP.

#261788 Received 6 Apr 2016; revised 7 Jun 2016; accepted 8 Jun 2016; published 20 Jun 2016