https://doi.org/10.2352/ISSN.2470-1173.2018.15.COIMG-181
© 2018, Society for Imaging Science and Technology

Separable Models for cone-beam MBIR Reconstruction

***Thilo Balke*[1]*, Soumendu Majee*[1]*, Gregery T. Buzzard*[2]*, Scott Poveromo*[3]*, Patrick Howard*[4]*, Michael A. Groeber*[5]*, John McClure*[5]*, Charles A. Bouman*[1]**

[1]***School of ECE, Purdue University, West Lafayette, IN 47907 USA;***
[2]***Department of Mathematics, Purdue University, West Lafayette, IN, USA***
[3]***Northrop Grumman Corporation, El Segundo, CA, USA;***
[4]***GE Aviation, Cincinnati, OH, USA;***
[5]***Air Force Research Laboratory, Wright-Patterson AFB, OH, USA***

Abstract

Cone-beam computed tomography (CT) is an attractive tool for many kinds of non-destructive evaluation (NDE). Model-based iterative reconstruction (MBIR) has been shown to improve reconstruction quality and reduce scan time. However, the computational burden and storage of the system matrix is challenging.

In this paper we present a separable representation of the system matrix that can be completely stored in memory and accessed cache-efficiently. This is done by quantizing the voxel position for one of the separable subproblems. A parallelized algorithm, which we refer to as zipline update, is presented that speeds up the computation of the solution by about 50 to 100 times on 20 cores by updating groups of voxels together.

The quality of the reconstruction and algorithmic scalability are demonstrated on real cone-beam CT data from an NDE application. We show that the reconstruction can be done from a sparse set of projection views while reducing artifacts visible in the conventional filtered back projection (FBP) reconstruction. We present qualitative results using a Markov Random Field (MRF) prior and a Plug-and-Play denoiser.

Introduction

3D volumetric cone-beam CT is a promising tool for NDE of additively manufactured parts. The measurement setup includes an X-ray point source, an object to be scanned, and a flat panel 2D detector array. Usually the object is rotated in a range of 360° while several hundred or a few thousand projection images are taken, after which the 3D volume is conventionally computed using a backprojection technique. Cone-beam CT offers high resolution compared to, for example, ultra-sound imaging techniques and significantly faster acquisition times than fan or parallel beam CT while utilizing the X-ray source energy more efficiently.

Conventional reconstruction techniques for cone-beam CT require a large umber of views and tend to suffer from image quality artifacts caused by scatter, beam-hardening, and metal. In contrast, MBIR reconstruction techniques require fewer views [1, 2] and generally reduce modeling artifacts [3]. In MBIR, the reconstruction requires an accurate representation of the physical measurement system including noise properties, a prior model of the unknown image, and a forward projection matrix, called the system matrix. MBIR techniques come however with the burden of being computationally expensive and memory demanding.

In the case of cone-beam CT, the computational difficulty derives from the fact that different slices of the volume are highly coupled in the measurement space. As a result, parallelization of the inversion algorithm becomes challenging. Also, rebinning techniques, which are common in helical cone-beam CT, break down in general when the cone angle is too large [4].

In similar tomographic problems like 2D fan-beam or parallel-beam CT the system matrix exhibits sufficient regularity and redundancy such that a sparse, compressed representation can be precomputed and completely stored in memory. This is useful in MBIR as the projection operation is performed multiple times iteratively and thus the precomputed matrix entries can be reused. This is in general overall faster than computing them on the fly.

In this paper we present a method to compute and completely store the system matrix of a cone-beam CT problem with large cone-angle which involves a quantization of the voxel locations. This matrix data structure is particularly suited for efficiently accessing columns and groups of columns, as is common in coordinate descent optimization. We also present a parallelization using shared memory multi-threading. The forward projector is based on the distance-driven (DD) projector but can easily be modified to more accurate separable projectors without interfering with the ability to store the system matrix.

Statistical Model for image Reconstruction

A common assumption in MBIR is that the measurement and the unknown image are random vectors that are jointly distributed. This way, the reconstruction aims to use measurement noise statistics to account for uncertainty while also using assumptions about the prior distribution of the image to improve image quality and stabilize the inversion. The solution to the inverse problem is then the most likely image given an assumed probability distribution.

Let the *image*, $X = [X_1, ..., X_N]^\top$, be the vector of unknown X-ray attenuation coefficients, where each X_j represents a cuboid voxel with index j in the image volume. Using a Bayesian framework the image is assumed to have a probability density, $p_X(x)$, which is referred to as the *prior model*.

Further, let the *sinogram*, $Y = [Y_1, ..., Y_M]^\top$, be the vector of projection measurements, where each Y_i corresponds to a projection measurement at a given detector pixel and a given rotation angle. The projection measurements, Y, are computed from the photon count vector, λ, using Beer's Law

$$Y_i = -\log\left(\frac{\lambda - \bar{\lambda}_{\text{dark}}}{\bar{\lambda}_{\text{blank}} - \bar{\lambda}_{\text{dark}}}\right) \tag{1}$$

where $\bar{\lambda}_{\text{dark}}$ is the background offset and $\bar{\lambda}_{\text{blank}}$ is the normalization scan.

Due to its counting nature, $(\lambda_i | X = x)$ is assumed to be distributed as Poisson$\{\bar{\lambda}\}$. Using a 2nd order Taylor expansion of the resulting density for Y, the likelihood density can be approximated [5] as a white Gaussian noise density $p_{Y_i|X}(y_i|X = x) = \mathcal{N}(A_{i,*}x, w_i^{-1})$, where $A_{i,*}$ is the i-th row of the *system matrix*, A

$$\log p_{Y|X}(y|X = x) = -\frac{1}{2}\|Y - Ax\|_W^2 + \mathrm{const}(y), \tag{2}$$

where $W = \mathrm{diag}([w_1, \ldots, w_M])$ is the diagonal *weight matrix* with entries [6]

$$w_i \propto \frac{\lambda_i^2}{\lambda_i + \sigma_{\mathrm{noise},i}^2}. \tag{3}$$

The detector noise variance, $\sigma_{\mathrm{noise},i}^2$, can be estimated by finding the sample variance of the detector background offset.

Now, the solution of the inverse problem is formulated as the the maximizer of the *maximum a posteriori* density, $p_{X|Y}$

$$\begin{aligned}\hat{x}_{\mathrm{MAP}} &= \arg\min_{x \geq 0}\left\{-\log p_{Y|X}(y|X = x) - \log p_X(x)\right\} \quad (4)\\ &= \arg\min_{x \geq 0}\left\{\frac{1}{2}\|Y - Ax\|_W^2 - \log p_X(x)\right\}.\end{aligned}$$

In subsequent sections we will present an efficient implementation for the cone-beam CT system matrix, A, describe the choice for the prior model $p_X(x)$, and describe the optimization algorithm for equation (4).

Computation of the Forward Model

In this section we describe the computation of the forward projector matrix. First, we describe the computation of an arbitrary matrix entry then the parametrization of the complete matrix, which is made feasible due to a quantization in the voxel locations.

Computation of the System Matrix

Conventional projection methods are optimized to forward or back project the entire image or measurement space, respectively. However, to solve the optimization problem stated in (4), we use the iterative coordinate descent (ICD) algorithm, which requires the forward projection of single voxels onto the measurement space. This corresponds conceptually to accessing the entries of single columns of the A-matrix. To accomplish the repeated forward projection we use a variation of the distance-driven projector as described in [7], while storing the matrix entries in a memory and cache efficient way.

First, we focus our description on the computation of an arbitrary matrix entry. Later, we describe the parametrization of the voxel location, which enables the efficient storage of the entire matrix.

Let j be the index of an arbitrary cuboid shaped voxel with base dimensions of $\Delta_{xy} \times \Delta_{xy}$ and height of Δ_z. Further, let i be the index of an arbitrary detector of size $\Delta_{dv} \times \Delta_{dw}$, corresponding to a particular view angle β. Then, the *exact* computation of the matrix entry, $A_{i,j} = \frac{\partial y_i}{\partial x_i}$[1], would consider all possible lines passing through the source, voxel j, and detector i, with appropriate integration. In the DD method, this is simplified by (1) imposing separability of the projection profile and (2) "flattening" the voxel conceptually, thus resulting in equal-length line intersections with the voxel. With this approximation the DD projector achieves reconstructions free of visible artifacts that are caused by the projector [7]. Also, when the image resolution is not too coarse—which is generally true in MBIR applications—the DD projector is not inferior to more exact, separable projectors using trapezoid footprints [8]. As a result of the separable footprints, the 3D problem can be separated into two different 2D subproblems, illustrated in figures 1 and 2, yielding the attenuation factors $B_{i,j}$ and $C_{i,j}$, respectively:

$$A_{i,j} = B_{i,j}\, C_{i,j}. \tag{5}$$

The u, v, and w-axis comprise the Cartesian scanner coordinate system (as opposed to the image coordinate system (x, y, z)) where the u-axis is perpendicular to the detector plane, the v-axis is perpendicular to the rotation axis and the w-axis is parallel to the rotation axis.

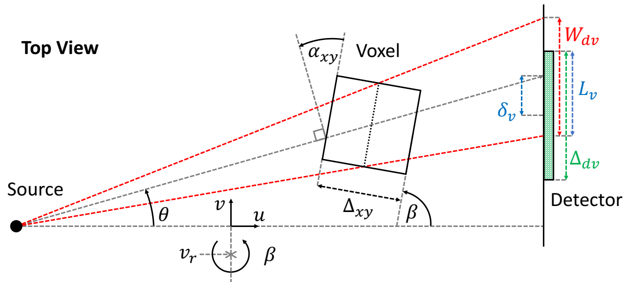

Figure 1. *Top view problem yields the $B_{i,j}$ entry. The v-axis is perpendicular to the rotation axis.*

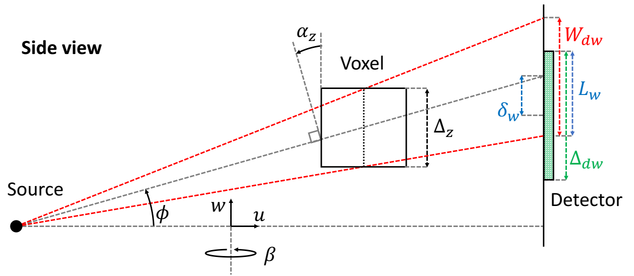

Figure 2. *Side view problem yields the $C_{i,j}$ entry. The w-axis is parallel to the rotation axis.*

In figure 1, we consider the plane that is perpendicular to the rotation axis (the u-v-plane, *table plane*), aiming to compute $B_{i,j}$. The DD projector only accounts for the rays passing through the *flattened* voxel, indicated by the red lines. All rays in that sector are assumed to have equal intersection lengths with the voxel, namely $\frac{\Delta_{xy}}{\cos\alpha_{xy}}$. The angle α_{xy} is between the center ray and the flattened voxel and the flattening direction is chosen such that $-\frac{\pi}{4} \leq \alpha_{xy} \leq \frac{\pi}{4}$. The projection profile is rectangular and has width W_{dv}. The detector response is assumed to be rectangular as well, thus having constant sensitivity, $\frac{1}{\Delta_{dv}}$, over a detector pixel width of Δ_{dv}. Thus, only the overlap length, L_v, matters for the

[1]assuming no measurement noise

computation, and L_v can be easily computed using the distance, δ_v, between the projection center and the voxel center.

The resulting equation for the attenuation coefficient, $B_{i,j}$, is

$$B_{i,j} = \left(\frac{\Delta_{xy}}{\cos\alpha_{xy}}\right)\left(\frac{1}{\Delta_{dv}}\right)(L_v), \tag{6}$$

where

$$\alpha_{xy} = [\beta - \theta]_{\frac{\pi}{2}} \tag{7}$$
$$L_v = \cap(W_{dv}, \Delta_{dv}; \delta_v), \tag{8}$$

and where we are using short-hand notation

$$[x]_{\frac{\pi}{2}} := \left(\left(x + \frac{\pi}{4}\right) \bmod \frac{\pi}{2}\right) - \frac{\pi}{4} \tag{9}$$
$$\cap(W_1, W_2; \delta) := \max\left\{\frac{W_1 + W_2}{2} - \max\left\{\frac{|W_1 - W_2|}{2}, |\delta|\right\}, 0\right\}. \tag{10}$$

In figure 2, we consider the plane that is parallel to the rotation axis (the u-w-plane), aiming to compute $C_{i,j}$. The rotation axis is parallel to the detector in this subproblem so that the voxel sides are parallel or perpendicular to the detector. We assume that the cone angle, ϕ, does not exceed $\pm\frac{\pi}{4}$, thus $\alpha_z = \phi$. The resulting equations are

$$C_{i,j} = \left(\frac{1}{\cos\alpha_z}\right)\left(\frac{1}{\Delta_{dw}}\right)(L_w), \tag{11}$$

where

$$L_w = \cap(W_{dw}, \Delta_{dw}; \delta_w). \tag{12}$$

Now that we have a way to compute the $A_{i,j}$ matrix index using equations (5), (6), and (11), we present an efficient way to parametrize and store the entire matrix.

Parametrization and storage of the System Matrix

Let $j = (j_x, j_y, j_z) \in \{0,...,N_x - 1\} \times \{0,...,N_y - 1\} \times \{0,...,N_z - 1\}$ be the 3D voxel index, where (j_x, j_y) corresponds to the horizontal x-y-plane and j_z corresponds to the vertical z-axis (= rotation axis). Then an arbitrary voxel center position, (x_v, y_v, z_v) is parametrized as

$$\begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix} = \begin{bmatrix} x_0 + \Delta_{xy}\ j_x \\ y_0 + \Delta_{xy}\ j_y \\ z_0 + \Delta_z\ \ j_z \end{bmatrix} \tag{13}$$

for some constants x_0, y_0, and z_0. The voxel position in scanner coordinates, (u_v, v_v, w_v), can be computed using a rotation matrix

$$\begin{bmatrix} u_v \\ v_v \\ w_v \end{bmatrix} = \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix} + \begin{bmatrix} 0 \\ v_r \\ 0 \end{bmatrix}, \tag{14}$$

where $\{(u, v, w) : u = 0, v = v_r, w \in \mathbb{R}\}$ is the rotation axis. The 3D measurement index, $i = (i_\beta, i_v, i_w) \in \{0,...,N_\beta - 1\} \times \{0,...,N_{dv} - 1\} \times \{0,...,N_{dw} - 1\}$, parametrizes the detector location, (u_d, v_d, w_d), and the view angle, β:

$$\begin{bmatrix} u_d \\ v_d \\ w_d \end{bmatrix} = \begin{bmatrix} u_{d0} \\ v_{d0} + i_v\Delta_{dv} \\ w_{d0} + i_w\Delta_{dw} \end{bmatrix}, \qquad \beta = \beta(i_\beta) \tag{15}$$

for some constants u_{d0}, v_{d0}, w_{d0} and a view angle list $\beta(\cdot)$. N_β is the number of views, N_{dv} is the number of detector columns, and N_{dw} is the number of detector rows.

Now, a possible—but highly inefficient—way would be to compute $B_{i,j}$ and $C_{i,j}$ for all possible choices of i and j using equations (6) and (11) and store the results in 6D arrays $B[N_x][N_y][N_z][N_\beta][N_{dv}][N_{dw}]$ and $C[N_x][N_y][N_z][N_\beta][N_{dv}][N_{dw}]$. In the following paragraphs we describe a more efficient way.

First note that most of the entries of the matrix are zero, so the matrix is sparse and structured. In a typical case the voxel sizes are $\Delta_{xy} = \frac{\Delta_{dv}}{\text{Magnification}}$ and $\Delta_z = \frac{\Delta_{dw}}{\text{Magnification}}$ and only at most a $\tilde{N}_{dv} \times \tilde{N}_{dw} \approx 3 \times 3$ window for the whole $N_{dv} \times N_{dw}$ detector pixels have a non-zero entry for a given view angle. Thus, we only have to store the non-zero entries and use pointer-like arrays to point to the location and side length of the non-zero area.

Also, these arrays would still contain many redundant entries. As a result of the separability assumption we can easily see that we have

$$B_{i,j} = B_{(i_\beta,i_v,i_w),(j_x,j_y,j_z)} = B_{(i_\beta,i_v,0),(j_x,j_y,0)} \qquad \text{for all } i_w \text{ and } j_z \tag{16}$$

and

$$C_{i,j} = C_{(i_\beta,i_v,i_w),(j_x,j_y,j_z)} = C_{(i_\beta,0,i_w),(j_x,j_y,j_z)} \qquad \text{for all } i_v \tag{17}$$

Thus, we can store the $B_{i,j}$ entries in a much smaller array, $B[N_x][N_y][N_\beta][\tilde{N}_{dv}]$, which has a feasible size. However, if we wanted to store the $C_{i,j}$ entries in an array $C[N_x][N_y][N_z][N_\beta][\tilde{N}_{dw}]$, it would still take up a great deal of memory as it has one more degree of freedom than the B array. To compress the C-matrix further, we will discretize the voxel position, as described in the next paragraphs.

First, note that the value of $C_{i,j}$ is completely determined if we know the values of u_v, w_v and w_d (see figure 2). Further, using equations (13), (14), and (15) we see that these depend on the indices i, j as follows:

$$\begin{aligned} u_v &= x_v\cos(\beta) - y_v\sin(\beta) && \leftarrow \text{only dep. on } j_x, j_y, i_\beta \quad (18) \\ w_v &= z_v && \leftarrow \text{only dependent on } j_z \\ w_d & && \leftarrow \text{only dependent on } i_w. \end{aligned}$$

The core idea behind the efficient storage is now to parametrize u_v differently. We use a fine grid of all possible voxel coordinates, u_v, as an approximation of equation (18).

$$j_u = \text{round}\left(\frac{u_v - u_0}{\Delta_u}\right) \quad \leftrightarrow \quad u_v \approx u_0 + j_u\Delta_u \tag{19}$$
$$\Delta_u = \frac{\Delta_{xy}}{\rho}, \tag{20}$$

where u_0 is the the smallest possible value for u_v, and the relative grid density, ρ, is a positive, real number. In practice we have found that $\rho \approx 10$ is more than sufficient. The new integer index, $j_u = j_u(j_x, j_y, i_\beta) \in \{0,..., N_u - 1\}$, can be precomputed using equation (20) and stored in an array, $j_u[N_x][N_y][N_\beta]$. Now, the $C_{i,j}$ values need not be computed for every (j_x, j_y, i_β) but only for every possible j_u.

The resulting array entries for the attenuation coefficients are

$$B_{(i_\beta,i_v,i_w),(j_x,j_y,j_z)} = B[j_x][j_y][i_\beta][\tilde{i}_v] \tag{21}$$

$$C_{(i_\beta,i_v,i_w),(j_x,j_y,j_z)} = C[j_u][j_z][\tilde{i}_w] \tag{22}$$

$$j_u = j_u[j_x][j_y][j_\beta] \tag{23}$$

where, again the tilde (˜) indicates the sparse dimension of the array.

With this parametrization, the C-array is often significantly smaller than the B-array since for a centered image volume, $N_u \approx \rho\sqrt{N_x^2+N_y^2}$. In the example case, when $N_x = N_y = N_z = N_\beta$ and $\tilde{N}_{dv} = \tilde{N}_{dw}$, the C-array is smaller than the B-array as long as $\rho < N_x/\sqrt{2}$. Since in practice the oversampling density, ρ, is much smaller than $N_x/\sqrt{2}$, the storage burden of the complete system matrix, A, reduces essentially to storing the 2D fan-beam projection matrix, B.

We point out that the computation of the resulting B, C, and j_u arrays was based upon the parametrization from this section and on the equations for the matrix entries themselves (equations (6) and (11)) from the previous section. However, these two steps are completely independent of one another. That is, using the mentioned parametrization scheme can be done while using any other forward projector of one's choice as long as it is separable.

Dealing with non-idealities: Implicit in our description of the imaging system the detector columns are parallel to the rotation axis. In practice this is often approximately but not exactly given. In that case we suggest preprocessing the scans that have been acquired with non-ideal conditions to emulate scans that do meet the ideal conditions regarding the detector orientation. This may be done by interpolating the projection scans with a non-homogeneous lattice according to the perspective distortion.

Prior Model

The prior distribution of the image, $p_X(x)$, is the core of the Bayesian framework. Often in MBIR the number of measurements is far smaller than the number of unknowns. Thus, without the prior model there would be infinity many images that would perfectly fit the data term, i.e. maximize the log-likelihood of equation (2). Without a suitable prior model the reconstruction will be noisy and unstable [9].

Recently, the q-Generalized Gaussian Markov random field (q-GGMRF) prior model [10] has shown favorable results. It gives the user flexibility about the regularization and edge penalty and achieves robustness through its convexity and continuous differentiability. In this work we are using this prior model, which has a density

$$\log p_X(x) = \sum_{\{s,r\}\in\mathscr{P}} b_{s,r}\,\rho(x_s - x_r) + \text{constant} \tag{24}$$

where $\mathscr{P}$ is the set of all neighboring voxel pairs with relative weights $b_{s,r}$ and the potential function, ρ, is given by

$$\rho(\Delta) = \frac{|\Delta|^p}{p\sigma_x^p}\left(\frac{\left|\frac{\Delta}{T\sigma_x}\right|^{q-p}}{1+\left|\frac{\Delta}{T\sigma_x}\right|^{q-p}}\right), \tag{25}$$

where typically the parameters are set to be $1 \le p < q \le 2$, adjusting the edge penalty and $T\sigma_x \approx \text{STD(noise)}$, adjusting the regularization strength.

Apart from demonstrating our method using the q-GGMRF prior model, we also present reconstructions using a Plug-and-Play prior [11] in the results section of this paper while using BM4D [12] as a denoiser. The methods for minimizing the slightly differing cost functions are fundamentally the same, as the quadratic approximation of the q-GGMRF potential is replaced by an actual quadratic norm of the augmented Lagrangian term. For more background we refer the reader to [11].

Computation of the solution

In this section we present the algorithmic solution of equation (4). The MAP cost function is strictly convex, resulting in a unique minimizer. Thus, in theory the choice of the optimization algorithm does not affect the reconstruction. ICD has shown favorable convergence speed [13] with comparable computational requirements to iterative methods as gradient based algorithms.

In this work we present an ICD based algorithm that efficiently computes the reconstruction. Its efficiency is in part achieved by the use of quadratic surrogate functions that substitute the non-quadratic prior term for a quadratic approximation [14, 15, 16], which enables a guaranteed cost reduction without an expensive half interval search. Using this method, the computational cost is mainly attributed to the data term. The contribution of our work is an algorithm that efficiently minimizes the approximated map cost function (4)

$$\begin{aligned} f(x;x') &= \frac{1}{2}\|Y-Ax\|_W^2 + \sum_{\{s,r\}\in\mathscr{P}} b_{s,r}\,\rho(x_s-x_r;x_s'-x_r') \\ &= \frac{1}{2}\|Y-Ax\|_W^2 + \sum_{\{s,r\}\in\mathscr{P}} \tilde{b}_{s,r}\,(x_s-x_r)^2 \end{aligned} \tag{26}$$

using an appropriate, quadratic surrogate function, $\rho(\Delta;\Delta')$.

As the modified cost function from equation (26) is quadratic, for each voxel j there exist θ_1 and θ_2 such that the coordinate descent objective is a simple quadratic

$$f(x+\Delta_{x_j}\varepsilon_j;x') = \theta_1\Delta_{x_j} + \frac{\theta_2}{2}\Delta_{x_j}^2 + \text{const} \tag{27}$$

where $e = Y - Ax$ is the *error sinogram*, which is a constantly updated auxiliary array. For details of the surrogate function, we refer the reader to [16]. We focus instead on the bottleneck of the computation of the voxel update and how it is solved efficiently. The resulting equations for θ_1 and θ_2 are

$$\begin{aligned} \theta_1 &= \underbrace{-e^\top WA_{*,j}}_{\theta_{1,Y|X}} + \sum_{r\in\partial j} 2\,\tilde{b}_{s,j}\,(x_s-x_j) &&= \theta_{1,Y|X}+\theta_{1,X} \\ \theta_2 &= \underbrace{-A_{*,j}^\top WA_{*,j}}_{\theta_{2,Y|X}} + \sum_{r\in\partial j} 2\,\tilde{b}_{s,j} &&= \theta_{2,Y|X}+\theta_{2,X}, \end{aligned} \tag{28}$$

where $A_{*,j}$ is the j-th column of A. The update pseudocode for the j-th voxel is shown in algorithm 1.

The main computational cost derives from the terms $\theta_{1,Y|X}$ and $\theta_{2,Y|X}$ and updating the error sinogram (line 5 in the algorithm).

Algorithm 1: Single Voxel ICD Updates

```
  Input: $j = (j_x, j_y, j_z)$
  Data: $e, W, A, x$
1 $\tilde{b}_{s,r} = \text{computeSurrograteCoeff.}(x)$ // equation omitted
2 $[\theta_1, \theta_2] = $ compute using equation (28)
3 $\Delta_{x_j} = \max\left\{\frac{-\theta_1}{\theta_2}, -x_j\right\}$ // Positivity Constraint
4 $x = x + \Delta_{x_j} \varepsilon_j$
5 $e = e - \Delta_{x_j} A_{*,j}$
```

Plainly using algorithm 1 does not exploit the structure of the A matrix cache-efficiently. In the next paragraphs we describe a faster algorithm that updates several voxels simultaneously. When using the term *simultaneously* we mean: compute θ_1's and θ_2's for multiple voxels for the same given data (e, W, A, x) and then update the image and error sinogram in sequence for the resulting Δ_{x_j}'s.

First, note that updating any two voxels simultaneously will not necessarily reduce the cost function of equation (4). However, consider the conditions

$$A_{*,j}^\top A_{*,j'} = 0 \qquad \text{and} \qquad x_j \notin \partial x_{j'} \tag{29}$$

for two voxels j and j'. It is easy to see that if these conditions are met then updating first j, then j' or first j' then j or both voxels simultaneously will yield the same results (compare equation (4)). This is because the first condition decouples the voxels in the forward model and the second term decouples them in the prior model. In this work the simultaneously updated voxels are a set of voxels that only differ in their j_z index and which are not adjacent to each other. If the voxel grid is not too coarse, the conditions of equation (29) are usually met. These updated voxels are referred to as *voxel line* or *z-line* or *zipline* [17, 16]. For example, the set $\text{Zipline} = \{0, 4, 8, 12, 16, \ldots, N_z - 1\}$ parametrizes a zipline when given (j_x, j_y). A fast algorithm to compute the terms $\theta_{1,Y|X}$ and $\theta_{2,Y|X}$ for all voxels in a given zipline is given in algorithm 2.

Algorithm 2: Zipline: Computation of $\theta_{1,Y|X}$'s and $\theta_{1,X}$'s

```
   Input: $(j_x, j_y)$, Zipline
   Output: Arrays: $\theta_{1,Y|X}, \theta_{2,Y|X}$
   Data: $e, W, A, x$
1  for $i_\beta = 0 : N_\beta - 1$ do
2    $j_u = j_u[j_x][j_y][i_\beta]$
3    for $\tilde{i}_v = 0 : i_{v,\text{stride}}[j_x][j_y][i_\beta] - 1$ do
4      $i_v = \tilde{i}_v + i_{v,\text{start}}[j_x][j_y][i_\beta]$
5      $B_{i,j} = B[j_x][j_y][i_\beta][\tilde{i}_v]$
6      for $m = 0 : |\text{Zipline}| - 1$ do
7        $j_z = \text{Zipline}[m]$
8        for $\tilde{i}_w = 0 : i_{w,\text{stride}}[j_u][j_z] - 1$ do
9          $i_w = \tilde{i}_w + i_{w,\text{start}}[j_u][j_z]$
10         $A_{i,j} = B_{i,j}\, C[j_u][j_z][\tilde{i}_w]$
11         $\theta_{1,Y|X}[m] \mathrel{+}= e[i_\beta][i_v][i_w]\, W[i_\beta][i_v][i_w]\, A_{i,j}$
12         $\theta_{2,Y|X}[m] \mathrel{+}= A_{i,j}\, W[i_\beta][i_v][i_w]\, A_{i,j}$
```

The update of the error sinogram is similar, but the body of the innermost loop (line 7) is $e[i_\beta][i_v][i_w] = e[i_\beta][i_v][i_w] - A_{i,j}\, \Delta_{x_j}[m]$, where again the work is done for all the voxels in the zipline at once.

To motivate why this approach is faster than single voxel updates note that the arrays are stored in row-major order, and hence consecutive row elements can be read much more efficiently than, for example, column elements. Under inspection of the loop structure, one can see that the memory access is optimized with that regard. However, the quantization of the u-coordinate positions (see equation (20)) introduces additional cache *in*efficiency. In general it is not the case that $j_u[j_x][j_y][i_\beta + k] = j_u[j_x][j_y][i_\beta] + k$. As a result, when *looping* over the i_β variable continuously, it creates discontinuous jumps in the j_u variable which again results in discontinuous (cache-inefficient) jumps in the access of the C-array elements (and corresponding index arrays $i_{w,start}$ and $i_{w,stride}$). By letting the i_β loop be the outer loop and considering multiple voxels at the same time, the j_u variable changes (jumps) less often while being able to reuse the B-matrix entries as often as possible for all the voxels in a zipline.

Parallelization: Each of the coefficients $\theta_{1,Y|X}[m]$ and $\theta_{2,Y|X}[m]$ consist of partial sums corresponding to partial view subsets. In our parallelized algorithm we use a shared-memory implementation where every thread is responsible for only a certain view subset. Conceptually every one of the P threads is executing algorithm 2 on the complete zipline, but only on N_β / P disjoint views.

Experimental Results

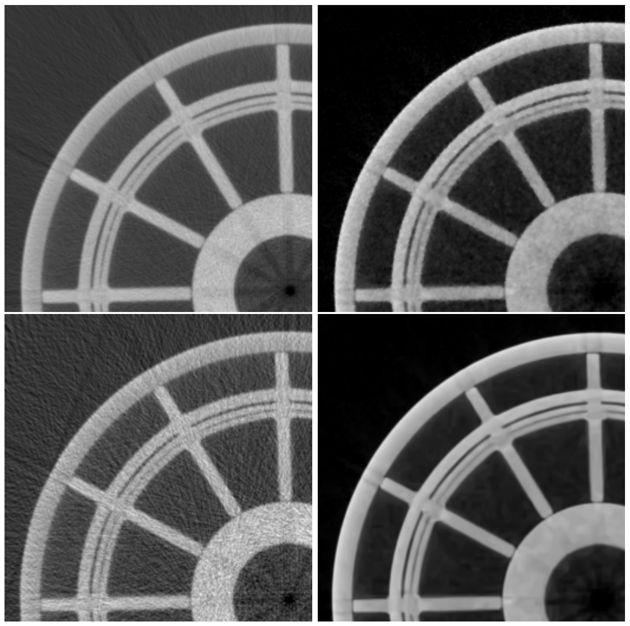

Figure 3. *Reconstruction of CoCr object. Reduced streaking.*
$\nwarrow$*: 2160-view FBP;* $\nearrow$*: 270-view q-GGMRF MBIR;*
$\swarrow$*: 270-view FBP;* $\searrow$*: 270-view BM4D Plug-and-Play MBIR.*

To demonstrate the improved reconstruction quality, we present reconstructions of a real data set of an additively manufactured CoCr part and compare the MBIR reconstruction with a FBP reconstruction using General Electrics's proprietary software. Further, we show improved performance by comparing the sequential single voxel update with the parallelized zipline update

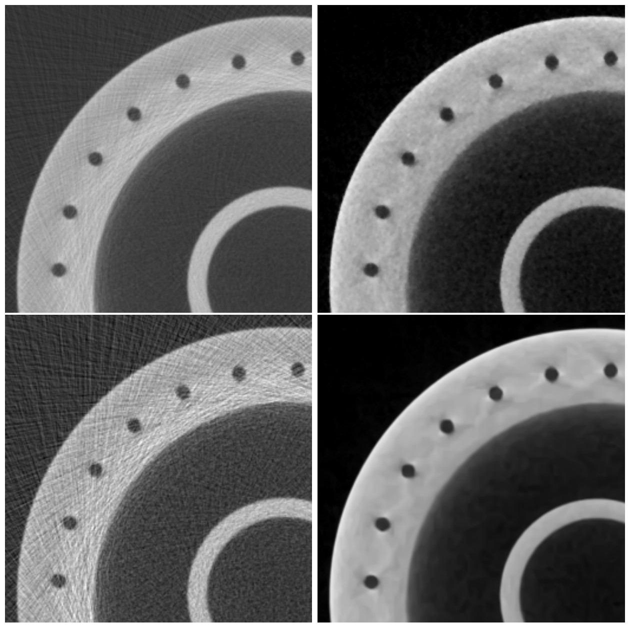
Figure 4. *Reconstruction of CoCr object. Reduced scatter artifacts. Improved detail.*
↖: 2160*-view FBP;* ↗: *270-view q-GGMRF MBIR;*
↙: 270*-view FBP;* ↘: *270-view BM4D Plug-and-Play MBIR.*

Experimental Setup

Scanner	GE Inspection Technologies *v|tome|x C 450 HS with scatter|correct*
Voltage, Current, Exposure	450 kV, 1.5 mA, 143 ms
Scatter Correction	GE proprietary
Source-detector distance	1160 mm
Magnification	1.5
Detectors	700×800, $(0.2\text{ mm})^2$
Object	CoCr, additively manufactured
Voxels	$700 \times 700 \times 870$, $(0.13\text{ mm})^3$

***Table 2*:** Experimental specifications of the reconstruction data shown in other figures.

scheme.

Table 1 shows a summary of some the experimental setup. Figures 3, 4, and 5 show slices that differ in the z-coordinate and which have been cropped. (That is, a slice corresponds to a pane through the volume of constant z-coordinate.) The reconstructions were obtained using 270 views in all cases except one of the FBP reconstructions, which was done with $8 \times 270 = 2160$ views. In general the MBIR reconstructions show improved edge contrast, reduced streaking and scatter artifacts, and improved detail with respect to the 270-view FBP. The Plug-and-Play reconstruction with the BM4D denoiser looks visually appealing due to its contour smoothness.

To evaluate the improved performance of the zipline update method, we first confirmed that the convergence rate, i.e. how many iterations it takes to compute the solution, is essentially the same as with the single voxel update method—about 10 to 20 iterations depending on regularization strength and desired accuracy. Thus, for the remainder of this analysis we focus entirely on the

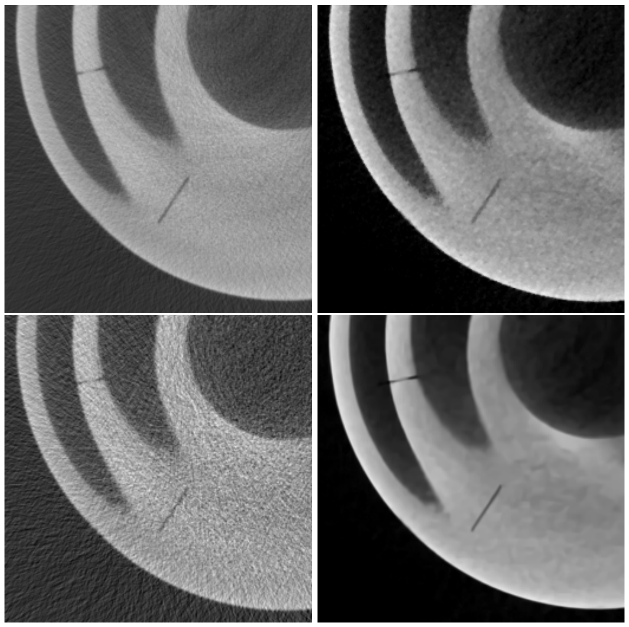
Figure 5. *Reconstruction of CoCr object. Improved detail.*
↖: 2160*-view FBP;* ↗: *270-view q-GGMRF MBIR;*
↙: 270*-view FBP;* ↘: *270-view BM4D Plug-and-Play MBIR.*

Time to compute update of all voxels (minutes)

Algorithm	Single 1-core	Zipline 1-core	Zipline 4-core	Zipline 10-core	Zipline 20-core
270 Views	414.1*	64.6	18.8	10.2	7.1
2160 Views	4508.2*	624.8	154.2	71.5	42.7

*Time extrapolated from partial update

***Table 1*:** Time per iteration in minutes.

update speed, i.e. the time it takes to update a fixed number of voxels.

For the computation of the solution we use an implementation of this algorithm using the C language and OpenMP for the shared-memory parallelization. The code is run on one compute node with dual 10-core Intel Xeon processors (20 processor cores in total).

Figure 6 shows the speedup of the zipline update with differing number of threads (= number of cores) compared to a single core zipline update scheme. Table 2 contains the corresponding iteration times while also including the iteration time of the single voxel update. We note that the zipline update is significantly faster than the single voxel update, even without multi-threading. In the single core case the zipline update is about 7 times faster than the single voxel update. Using 20 cores the zipline update is 58 times faster when using 270 views and 106 times faster when using 2160 views. The zipline update algorithm scales reasonably well in the in the range from 1 to 20 cores while reaching a parallel efficiency of about 73% for the 20-core, 2160-view case while being slightly less efficient in the 270-view case.

Conclusions

Cone-beam CT is a relatively cheap and versatile tool in NDE imaging problems while posing considerable computational burden in model based approaches. In this work we presented

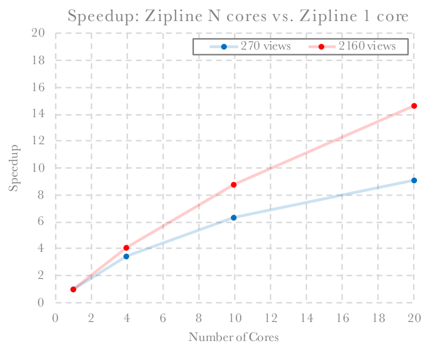

Figure 6. *Scalability of the zipline parallelization. Speedup is computed as the time per iteration using 1 core divided by the time using N cores.*

a method that makes this burden more feasible by representing the system matrix in a separable and compressed fashion and by parallelizing the algorithm using a zipline update scheme. The approach to store the complete system matrix is illustrated (but not limited to) using a distance-driven projector. That is, the method can be applied to any (more accurate) projector as long as that one is separable. Using our method, the problem of storing of the 3D cone-beam system matrix is in essence reduced to storing a 2D fan-beam projection matrix. The computational improvement is based upon the zipline update in which decoupled voxels are updated simultaneously: this feature efficiently exploits the structure of the system matrix and how it is laid out in memory, while also reducing the parallel overhead of multi-threaded implementation.

This model based approach enables reconstruction from sparse views and can thus dramatically reduce scan time.

Acknowledgements

The authors gratefully acknowledge support from the Metals Affordability Initiative (contract number FA8650-16-2-5248). We also thank Scott Poveromo, Sung Park and Michael Devito (Northrop Grumman); Patrick Howard, Scott Gold and John Janning (General Electric Aviation); John McClure and Ken LaCivita (AFRL/RXS) for both their suggestions, guidance as well as the cone-beam CT data used in our experiments.

References

[1] S. J. Kisner, E. Haneda, C. A. Bouman, S. Skatter, M. Kourinny, and S. Bedford, "Model-based ct reconstruction from sparse views," in *Second International Conference on Image Formation in X-Ray Computed Tomography*, pp. 444–447, 2012.

[2] K. A. Mohan, S. Venkatakrishnan, J. W. Gibbs, E. B. Gulsoy, X. Xiao, M. De Graef, P. W. Voorhees, and C. A. Bouman, "Timbir: A method for time-space reconstruction from interlaced views," *IEEE Transactions on Computational Imaging*, vol. 1, no. 2, pp. 96–111, 2015.

[3] P. Jin, C. A. Bouman, and K. D. Sauer, "A model-based image reconstruction algorithm with simultaneous beam hardening correction for x-ray ct.," *IEEE Trans. Computational Imaging*, vol. 1, no. 3, pp. 200–216, 2015.

[4] M. Defrise, F. Noo, and H. Kudo, "Rebinning-based algorithms for helical cone-beam ct," *Physics in Medicine & Biology*, vol. 46, no. 11, p. 2911, 2001.

[5] C. A. Bouman and K. Sauer, "A unified approach to statistical tomography using coordinate descent optimization," *IEEE Transactions on image processing*, vol. 5, no. 3, pp. 480–492, 1996.

[6] J.-B. Thibault, C. A. Bouman, K. D. Sauer, and J. Hsieh, "A recursive filter for noise reduction in statistical iterative tomographic imaging," in *Proceedings of SPIE*, vol. 6065, pp. 264–273, 2006.

[7] B. De Man and S. Basu, "Distance-driven projection and backprojection in three dimensions," *Physics in medicine and biology*, vol. 49, no. 11, p. 2463, 2004.

[8] Y. Long, J. A. Fessler, and J. M. Balter, "3d forward and backprojection for x-ray ct using separable footprints," *IEEE transactions on medical imaging*, vol. 29, no. 11, pp. 1839–1850, 2010.

[9] C. Bouman and K. Sauer, "A generalized gaussian image model for edge-preserving map estimation," *IEEE Transactions on Image Processing*, vol. 2, no. 3, pp. 296–310, 1993.

[10] J.-B. Thibault, K. D. Sauer, C. A. Bouman, and J. Hsieh, "A three-dimensional statistical approach to improved image quality for multislice helical ct," *Medical physics*, vol. 34, no. 11, pp. 4526–4544, 2007.

[11] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, pp. 945–948, IEEE, 2013.

[12] M. Maggioni, V. Katkovnik, K. Egiazarian, and A. Foi, "Nonlocal transform-domain filter for volumetric data denoising and reconstruction," *IEEE transactions on image processing*, vol. 22, no. 1, pp. 119–133, 2013.

[13] K. Sauer and C. Bouman, "A local update strategy for iterative reconstruction from projections," *IEEE Transactions on Signal Processing*, vol. 41, no. 2, pp. 534–548, 1993.

[14] J. A. Fessler, E. P. Ficaro, N. H. Clinthorne, and K. Lange, "Grouped-coordinate ascent algorithms for penalized-likelihood transmission image reconstruction," *IEEE transactions on medical imaging*, vol. 16, no. 2, pp. 166–175, 1997.

[15] A. R. De Pierro, "A modified expectation maximization algorithm for penalized likelihood estimation in emission tomography," *IEEE transactions on medical imaging*, vol. 14, no. 1, pp. 132–137, 1995.

[16] Z. Yu, J.-B. Thibault, C. A. Bouman, K. D. Sauer, and J. Hsieh, "Fast model-based x-ray ct reconstruction using spatially nonhomogeneous icd optimization," *IEEE Transactions on image processing*, vol. 20, no. 1, pp. 161–175, 2011.

[17] T. M. Benson, L. Fu, and B. K. De Man, "Zipline: A fast update scheme for reconstruction with separable system models," in *Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC), 2012 IEEE*, pp. 2343–2347, IEEE, 2012.

Author Biography

Thilo Balke is a PhD student and research assistant in the Purdue University School of Electrical and Computer Engineering. His research interests include computational imaging, fast algorithms for model based tomographic reconstruction and statistical machine learning. Thilo received his B.S. degree in electrical engineering and information technology from the Ruhr-University of Bochum, Germany, in 2014, where he graduated first in his class and received the Infineon Bachelor Award for Academic Excellence.