Non-Homogeneous Updates for the Iterative Coordinate Descent Algorithm

Zhou Yu[a], Jean-Baptiste Thibault[b], Charles A. Bouman[a], Ken D. Sauer[c], and Jiang Hsieh[b]

[a]School of Electrical Engineering, Purdue University, West Lafayette, IN 47907-0501;
[b]Applied Science Laboratory, GE Healthcare Technologies, W-1180, 3000 N Grandview Bvd, Waukesha, WI 53188;
[c]Department of Electrical Engineering, 275 Fitzpatrick, University of Notre Dame, Notre Dame, IN 46556-5637

ABSTRACT

Statistical reconstruction methods show great promise for improving resolution, and reducing noise and artifacts in helical X-ray CT. In fact, statistical reconstruction seems to be particularly valuable in maintaining reconstructed image quality when the dosage is low and the noise is therefore high. However, high computational cost and long reconstruction times remain as a barrier to the use of statistical reconstruction in practical applications. Among the various iterative methods that have been studied for statistical reconstruction, iterative coordinate descent (ICD) has been found to have relatively low overall computational requirements due to its fast convergence.

This paper presents a novel method for further speeding the convergence of the ICD algorithm, and therefore reducing the overall reconstruction time for statistical reconstruction. The method, which we call non-homogeneous iterative coordinate descent (NH-ICD) uses spatially non-homogeneous updates to speed convergence by focusing computation where it is most needed. Experimental results with real data indicate that the method speeds reconstruction by roughly a factor of two for typical 3D multi-slice geometries.

Keywords: Computed tomography, iterative reconstruction, coordinate descent

1. INTRODUCTION

Traditionally, reconstruction algorithms for Computed Tomography (CT) have used direct methods, such as the Filtered Back-Projection (FBP) algorithm. However, the development of CT technology, especially the emergence of helical CT systems with multi-slice detectors has presented challenges to direct methods due to the difficulty of accurately modeling the data acquisition process, system noise, and reconstructed image characteristics. Statistical reconstruction methods have recently been introduced to multi-slice CT in an effort to address these modeling issues, and prototype algorithms are now available for a fully 3D reconstruction of helical CT data.[1,2] Experiments indicate that statistical reconstructions methods have the potential to generate higher quality images with fewer artifacts and lower noise, even when X-ray dosage is reduced.[3]

Maximum *a posteriori* (MAP) estimation adds a statistical image model to the usual data mismatch term in order to regularize the results. Let the vector x be the three-dimensional image, and let y be the vector of tomographic measurements. The MAP estimate is given by minimizing the following cost function derived from the statistical models:

$$\hat{x} = \arg\min_{x \geq 0} \left\{ \frac{1}{2} (y - Ax)^T D (y - Ax) + U(x) \right\} \tag{1}$$

Further author information: (Send correspondence to J.B.T.)
Z.Y.: E-mail: yuz@purdue.edu, Telephone: 1 765 494 0751
J.B.T.: E-mail: jean-baptiste.thibault@med.ge.com, Telephone: 1 262 312 7404
C.A.B.: E-mail: bouman@purdue.edu, Telephone: 1 765 494 0340
K.D.S.: E-mail: sauer@nd.edu, Telephone: 1 574 631 6999
J.H.: E-mail: jiang.hsieh@med.ge.com, Telephone: 1 262 312 7635

Computational Imaging V, edited by Charles A. Bouman, Eric L. Miller, Ilya Pollak
Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 6498, 64981B, © 2007 SPIE-IS&T · 0277-786X/07/$18

The cost function first includes a quadratic log-likelihood term formed by a forward system matrix A, and a diagonal weighting matrix D which reflects variations in the credibility of the data.[4] The second term encourages local smoothness in the solution while preserving edge characteristics. We use a Markov random field (MRF) model, where $U(x)$ takes the form of the logarithm of an *a priori* density, as the sum of potential functions $\rho(x_j - x_k)$, where x_j and x_k are neighboring voxels.[5] With this or any other convex and continuously differentiable choice of prior, the resulting convex objective guarantees convergence.

A major obstacle for clinical application is the fact that these statistical reconstruction methods are very computationally demanding compared to conventional analytical algorithms. Statistical reconstruction methods generally require the use of an iterative method to solve the optimization of (1). Among the optimization algorithms compared in,[6] the ICD algorithm[7] has been found to have a relatively fast convergence behavior when it is initialized with the FBP reconstruction. The ICD algorithm decomposes the N-dimensional optimization problem into a sequence of greedy 1-dimensional voxel updates. A full iteration of the ICD algorithm then updates all the voxels in the image volume once and once only.

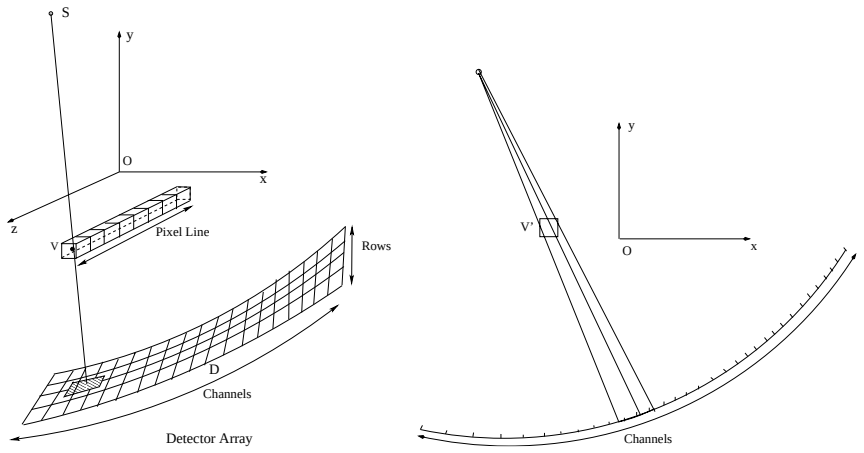
Other typical algorithms such as flavors of expectation maximization (EM),[8] conjugate gradients (CG),[9] ordered subsets (OS),[10] and other algorithms, are often used for tomographic optimization. However, ICD remains competitive against those other approaches due to its rapid convergence. In fact, ICD has a number of useful properties. It has particularly rapid convergence near edges and high frequency portions of the reconstruction; it can easily incorporate positivity's constraints and non-Gaussian prior distributions; and it naturally allows for updates that vary non-uniformly across the reconstructed image.

In this paper, we propose a novel non-homogeneous iterative coordinate descent (NH-ICD) algorithm which significantly improves the convergence speed of ICD. The NH-ICD algorithm takes advantage of the flexibility of ICD by selectively updating voxels that are furthest from convergence. Typically, the error between the FBP initialization of ICD and the converged statistical reconstruction is not uniformly distributed across the image. In fact, this initial error tends to be primarily distributed around edges and other localized regions. Therefore, the convergence of ICD can be increased by focusing computational resources on these important locations. In order to select the order of pixel updates, we formulate a pixel selection criteria (PSC) which is designed to predict the areas of greatest error. Our experimental results indicate that NH-ICD converges with roughly half the number of equivalent iterations required for adequate convergence of ICD.

2. HOMOGENEOUS ITERATIVE COORDINATE DESCENT (ICD) FOR 3D RECONSTRUCTION

In this section, we introduce the homogeneous ICD algorithm for reconstruction of 3D volumes from data obtained using a helical scan multi-slice CT system. The geometry of a helical scan multi-slice CT system is illustrated in Figure 1 where S denotes the focus of the x-ray source, D is the detector array, and the detectors are located on an arc which is centered at S. The detectors are aligned in channels and rows, as shown in Figure 1(a), and a single row of detectors forms an arc that is equidistant from S. When taking measurements in the helical scan mode, the source and detector array rotates around the patient, while the patient is simultaneously translated in the direction perpendicular to the rotation plane. The set of measurements taken at each source location is called a view. More generally, the measurement data can be indexed by its view, channel, and row, and we use the notation v, c, and r to denote the view, channel, and row indices respectively. The trajectory of S relative to the patient forms a helical path. To define the coordinate system of the reconstructed image volume, let z be the axis about which the helix rotates, and let the x-y plane be the plane that is orthogonal to z with y pointing upward. We use the notation $x_{i,j,k}$ to denote a voxel, where the indices i,j, and k corresponds to the voxel's x, y, and z coordinate.

Let f be the cost function described in (1), and $x_l^{(m)}$ denote a voxel's value at the m^{th} voxel update, where $l = (i, j, k)$ represents the index of the voxel, and m keeps track of the number of voxel updates performed. Let the function $C(m)$ determine the index of the voxel to be updated in the m^{th} update. In the conventional ICD algorithm, $C(m)$ selects each voxel once and only once per iteration; however, the order of voxel updates may



(a) (b)

Figure 1. Illustration of the geometry of helical multi-slice CT. In (a), S is the focus of the x-ray source, D is the detector array, in which detectors are aligned in channels and rows. Notice that a single row of detectors forms an arc which is equidistant from S, but a single channel falls along a straight line parallel to the z-axis. Voxels along the same (x, y) location form a pixel line. In (b), V' is the projection of a voxel onto the x-y plane, the channel response can be computed in the x-y plane.

vary with each full iteration through the voxels. When a voxel is selected, it is updated according to the rule

$$x_l^{(m)} = \begin{cases} \arg\min_{x_{C(m)} \geq 0} f(x_{l_1}^{(m-1)}, \cdots, x_{C(m)}, \cdots x_{l_N}^{(m-1)}) & \text{if } l = C(m) \\ x_l^{(m-1)} & \text{if } l \neq C(m) \end{cases} \tag{2}$$

where N denotes the total number of voxels to be reconstructed. In (2) only one voxel $x_{C(m)}$ is selected to be updated, and the values of all the other voxels are fixed in the cost function; thus, the N-dimensional problem of (1) is reduced to a sequence of 1D optimization problems as shown in (2).

To update a voxel, we must compute the voxel's projection onto the detector array for each view. More specifically, we must compute the function $P_{i,j,k}(v, c, r)$, which is the forward system model with property that

$$E[y_{v,c,r}] = \sum_{i,j,k} P_{i,j,k}(v, c, r) x_{i,j,k}$$

The function $P_{i,j,k}(v, c, r)$ is approximately separable, and can be written as

$$P_{i,j,k}(v, c, r) = P_{i,j}(v, c) P_{i,j,k}(v, r)$$

where $P_{i,j}(v, c)$ is called the channel response, and $P_{i,j,k}(v, r)$ is called the row response. Since the detectors in one channel are aligned parallel to the z axis, the channel response can be computed in the x-y plane, as shown in Figure 1(b). We refer to the line of voxels that fall along the same (i, j) in the x-y plane position as a pixel line. Notice that all the voxels on a pixel line project to the same rectangle on the x-y plane, and thus have the same channel response. Therefore the channel response only depends on the indices (i, j). For this reason, it is efficient to update voxels on a pixel line in sequence so that the calculation of channel response need only be done once for all the voxels in the pixel line.

We have investigated two methods for selecting the pixel line (i, j). One method is to select the pixel lines by raster scan order. A second method is to select the pixel lines in random order, so that each pixel is selected once per iteration, but the order of selection is randomized with a uniform distribution. The random update order can reduce the correlation between consecutive updates, and experimental results indicate that it provides faster convergence than raster order,[11] and therefore is typically used in the conventional ICD algorithm.

3. NON-HOMOGENEOUS ITERATIVE COORDINATE DESCENT (NH-ICD)

The NH-ICD algorithm proposed in this paper is a variation of the ICD algorithm which updates some voxels more often than others. In this case, $C(m)$ will sometimes repeatedly update a pixel line before other pixel lines are visited. To do this, we design a pixel selection criteria (PSC) which is greater for pixel lines which are likely to be further from convergence, and we introduce a pixel update scheduler (PUS) which uses the PSC to determine the update order of pixel lines.

3.1. Pixel Selection Criteria

To compute the PSC, we first form a 2D image $u^{(m)}$ which contains the magnitude of the last update change for the pixel line (i, j). To do this, we initialize $u_{ij}^{(0)}$ with zeros, $u_{ij}^{(0)} = 0$, and then update $u_{i,j}^{(m)}$ with each ICD update of a pixel line using the following relation:

$$u_{i,j}^{(m)} = \begin{cases} \sum_{k=1}^{K} |x_{i,j,k}^{(m)} - x_{i,j,k}^{(m-K)}| & \text{if } C(m) = (i, j, K) \\ u_{ij}^{(m-1)} & \text{if } C(m) \neq (i, j, K) \end{cases} \tag{3}$$

where $x_{i,j,k}^{(m)}$ is the voxel's value after the m^{th} update, and $x_{i,j,k}^{(m-K)}$ is the voxel's value before the pixel line is visited. Notice that after the first full iteration, all the values of $u_{ij}^{(m)}$ will have been updated once.

The values of the PSC, which we denote by the function $S_{ij}^{(m)}$, are computed from the values of $u_{ij}^{(m)}$ via the equation. The PSC is computed by applying a 2D low pass filter on the update map to reduce high frequency noise, that is,

$$S_{ij}^{(m)} = \sum_{p=-2}^{2} \sum_{q=-2}^{2} u_{ij}^{(m)} h_{i-p,j-q} \tag{4}$$

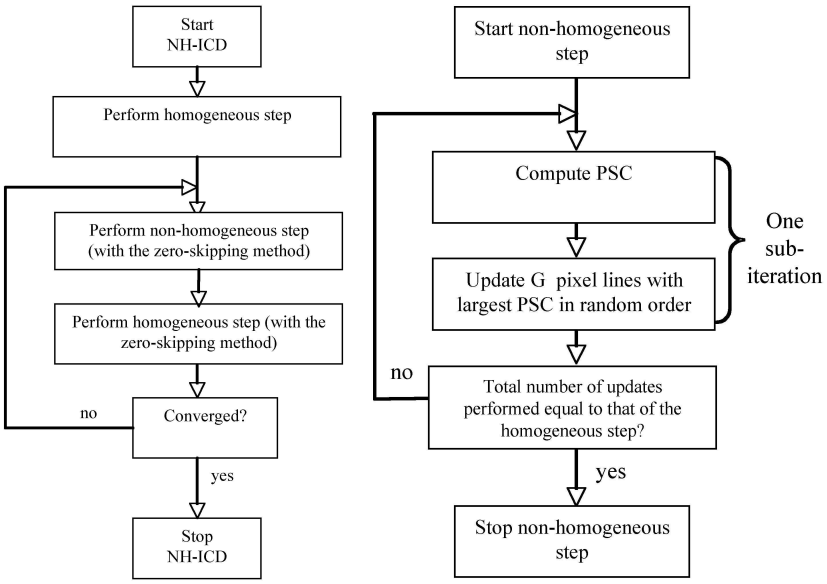
where the filter kernel h is a 5 by 5 Hamming window.

3.2. Pixel Update Scheduler

The update scheduler provides a method to select pixel lines for update based on the PSC. The update scheduler updates the pixel lines with large PSC more often, while still ensuring that every pixel line is regularly visited. To do this, the NH-ICD algorithm alternates between homogeneous and non-homogeneous steps. The homogeneous NH-ICD step is similar to a conventional ICD iteration, while the non-homogeneous step comprises several sub-iterations. In each non-homogeneous sub-iteration, a group of pixel lines are selected based on their PSC values and then updated in sequence. Figure 2(a) shows the block diagram for the NH-ICD algorithm which starts with a homogeneous step and then performs non-homogeneous and homogeneous steps alternately.

The function UpdatePixelLine$(i, j, x, y, ZeroSkippingFlag)$ is used in both homogeneous and non-homogeneous steps to update the voxels on the selected pixel line (i, j), and to compute u_{ij} using (3). The pseudo code in Figure 3 precisely specifies the function UpdatePixelLine. The input of the function is the pixel location (i, j), the image volume x, the measurement data y, and a boolean variable $ZeroSkippingFlag$. The output of the function is the updated image x and u_{ij}. The voxels on the pixel line are updated in sequence by the function UpdateVoxel(i, j, k, x, y) which computes the update value for the voxel $x_{i,j,k}$ using (2). Notice that, the NH-ICD algorithm sometimes does not update voxels when both the voxel and its neighbors are zero. We refer to this as the zero-skipping method. The zero-skipping method is used only when the boolean variable $ZeroSkippingFlags = true$. The positivity constraint in (1) is enforced to guarantee that no reconstructed voxel in the image volume can be negative since this is not physically possible. We have found that voxels that hit the positivity constraint typically tend to stay at zero during the convergence process. Therefore, most zero valued voxels are skipped to reduce computation. However, we do update voxels whose neighbors are non-zero to reduce the possibility of skipping voxels that should be non-zero.

In the homogeneous step, the method to select the pixel locations is the same as that used in the conventional ICD algorithm, that is, every pixel line is visited once per iteration, and pixels are visited in a random order.



(a) (b)

Figure 2. Block diagram for the NH-ICD algorithm. (a) illustrates the top level NH-ICD algorithm, and (b) shows the non-homogeneous step.

UpdatePixelLine$(i, j, x, y, ZeroSkippingFlag)$\{
  $K \leftarrow$ total number of voxels in the pixel line
  for $k = 1$ to K **do**
    if $\{(x_{i,j,k} = 0)$ **and** $(\forall l \in \partial(i, j, k), x_l = 0)$ **and** $(ZeroSkippingFlag =$ **true**$)\} =$ **false then**
      $x_{i,j,k} \leftarrow$UpdateVoxel(i, j, k, x, y)
    end if
  end for
  $u_{ij} \leftarrow$Update u_{ij} using (3)
  return (x, u_{ij})
\}

Figure 3. The pseudo code specification of function UpdatePixelLine(i,j). The input (i, j) is the location of pixel line to be updated. The function updates the voxels on the pixel line in sequence, and the element u_{ij} in the update map is also computed by this function. $\partial(i, j, k)$ denotes the 3D neighborhood of voxel x_{ijk}. When the zero-skipping method is used, zero valued voxels with neighbors that are zero are not updated.

To do this, as shown in Figure 4, we form a list L which contains all the pixel locations in the image, then the elements in L are randomly shuffled by the function Randomize(L). The pixel lines in L are updated in sequence using the function UpdatePixelLine. The function HomogeneousStep returns the updated the image volume x, and the 2D image u which provides information for pixel selection in the next non-homogeneous step.

Figure 2(b) shows the block diagram of the non-homogeneous step. In the non-homogeneous step, we perform several sub-iterations, and in each sub-iteration we update a selected group of pixel lines. The pseudo code in Figure 5 specifies the non-homogeneous step. Let G be the number of pixel lines to be updated in one sub-iteration, and N_p be the total number of pixel lines in the image. G is computed by $G = p * N_p$, in which p is a parameter that specifies the group size. Each sub-iteration has two steps, the pixel selection step and the update step. In the pixel selection step, first, the function ComputePSC(u) computes the PSC using (4). The function SelectPixelLine(S, G) selects G pixel lines with largest PSC values, and assign their pixel locations to

```
HomogeneousStep(x, y, u, ZeroSkippingFlag){
  N_p ← the total number of pixel lines
  L ← all the pixel locations in the reconstructed image
  L ← Randomize(L)
  for m = 1 to N_p do
    (i, j) ← L(m)
    (x, u_ij) ← UpdatePixelLine(i, j, x, y, ZeroSkippingFlag)
  end for
  return (x, u)
}
```

Figure 4. The pseudo code specification of a homogeneous step. L is a list that contains all the pixel locations, the Randomize(L) function will randomize the order of elements in L. The function HomogeneousStep returns the updated image volume x, and the 2D image u.

```
NonhomogeneousStep(x, y, u, ZeroSkippingFlag){
  N_p ← the total number of pixel lines
  G ← p * N_p
  N_nz ← CountNonzeroVoxels(x)
  N_u ← 0
  while N_u < N_nz do
    S ← ComputePSC(u)
    L_G ← SelectPixelLine(S, G)
    L_G ← Randomize(L_G)
    for m = 1 to G do
      (i, j) ← L(m)
      (x, u_ij) ← UpdatePixelLine(i, j, x, y, ZeroSkippingFlag)
      N_u ← N_u + Number of voxels updated for pixel line L_G(m)
    end for
  end while
  return (x, u)
}
```

Figure 5. The pseudo code specification of the non-homogeneous step. The code in the while loop corresponds to the sub-iteration. In each sub-iteration, the PSC is computed by the function ComputePSC(u), and then the function SelectPixelLine(S, G) selects G pixel lines with largest PSC values. The selected pixel lines are updated in random order. The function CountNonzeroVoxels(x) computes the number of voxels that will be updated in this step.

the list L_G. The inputs to SelectPixelLine are the PSC S and the parameter G, and the function returns the list L_G containing the selected pixel locations ordered by their PSC values. L_G is then random shuffled by the function Randomize(L_G) to reduce the correlation between consecutive updates. In the update step, the pixel lines in L_G are updated sequentially using the UpdatePixelLine function, and u_{ij} is computed for each pixel line that is visited.

In the next sub-iteration the updated image u will be used for pixel selection. Since the pixel lines are selected in the beginning of one sub-iteration, a pixel line can be updated only once in one sub-iteration. However, pixel lines with large PSC may be selected multiple times in consecutive sub-iterations during the non-homogeneous step.

In the non-homogeneous step, we use variable N_u to keep track of the number of voxel updates performed, and we use N_{nz} to denote the number of voxels that are updated in the homogeneous step with zero-skipping method. The function CountNonzeroVoxels(x) visits all the voxel locations and check if the voxel can be skipped

```
main() {
  ZeroSkippingFlag ← false
  (x, u) ←HomogeneousStep(x, y, u, ZeroSkippingFlag)
  ZeroSkippingFlag ← true
  repeat
    (x, u) ← NonhomogeneousStep(x, y, u, ZeroSkippingFlag)
    (x, u) ← HomogeneousStep(x, y, u, ZeroSkippingFlag)
  until Image converged to desired level
}
```

Figure 6. Top level NH-ICD algorithm

by the zero-skipping method, and returns the total number of voxels that will be updated. The non-homogeneous step will terminate when $N_u \geq N_{nz}$.

The top level NH-ICD algorithm is specified by the pseudo code in Figure 6. The algorithm starts with a homogeneous step without using the zero-skipping method so that every voxel in the image volume is updated once in the first step. After that, non-homogeneous and homogeneous steps with the zero-skipping method are performed alternately until the image is converged to the desired level.

3.3. The Interleaved form of NH-ICD

```
main() {
  Decimate pixels into subsets S_0, S_1, S_2, and S_3
  ZeroSkippingFlag ← false
  for g = 0 to 3 do
    (x, u) ← InterleavedStep(x, y, u, ZeroSkippingFlag, S_g)
  end for
  ZeroSkippingFlag ← true
  repeat
    (x, u) ← NonhomogeneousStep(x, y, u, ZeroSkippingFlag)
    (x, u) ← HomogeneousStep(x, y, u, ZeroSkippingFlag)
  until Image converged to desired level
}
```

Figure 7. Interleaved form of NH-ICD

The basic non-homogeneous update algorithm described above is identical to the conventional ICD for the first iteration because it is necessary to first initialize the values of the update map with a full iteration of homogeneous updates. This full homogeneous iteration limits the potential speedup in the convergence rate of NH-ICD. In order to overcome this limitation, we introduce an interleaved step of NH-ICD which starts with a homogeneous update of pixel subsets, and then performs a non-homogeneous sub-step with a limited number of sub-iterations.

As shown in the pseudo code in Figure 7, the interleaved NH-ICD algorithm starts with partitioning the pixel locations into 4 complementary subsets: $S_0 = \{(i,j)|i = 2p, j = 2q\}$, $S_1 = \{(i,j)|i = 2p+1, j = 2q\}$, $S_2 = \{(i,j)|i = 2p, j = 2q{+}1\}$ and $S_3 = \{(i,j)|i = 2p{+}1, j = 2q{+}1\}$, in which $0 \leq p \leq N_i/2$, $0 \leq q \leq N_j/2$, and N_i, N_j denotes the number of pixels on each coordinate. Next, the function InterleavedStep$(x, y, u, ZeroSkippingFlag, S_g)$ is called 4 times. Each time, a different subset S_g is selected. As shown in Figure 8, within the function InterleavedStep, the image is updated in two sub-steps. The first sub-step is a homogeneous sub-step that updates the pixel lines in the selected subset S_g in a random order. Next, in the non-homogeneous sub-step, pixel lines

```
InterleavedStep(x, y, u, ZeroSkippingFlag, S_g) {
        /* Perform homogeneous sub-step */
  L ← S_g
  N_g ← |S_g|
  L ← Randomize(L)
  for m = 1 to S_g do
    (i, j) ← L(m)
    (x, u_ij) ← UpdatePixelLine(i, j, x, y, ZeroSkippingFlag)
  end for
        /* Perform non-homogeneous sub-step */
  N_p ← total number of pixel lines in the image
  G ← 0.05 * N_p
  for N = 1 to 5 do
    S ← ComputePSC(u)
    L_G ← SelectPixelLine(S, G)
    L_G ← Randomize(L_G)
    for m = 1 to G do
      (i, j) ← L_G(m)
      (x, u_ij) ← UpdatePixelLine(i, j, x, y, ZeroSkippingFlag)
    end for
  end for
}
```

Figure 8. The function InterleavedStep$(x, y, u, ZeroSkippingflag, S_g)$ performs homogeneous updates on the selected subset of pixel lines, and then it performs 5 non-homogeneous sub-iterations on pixel lines with large PSC.

with large PSC are selected. The non-homogeneous sub-step is similar to a regular non-homogeneous step except that it performs only 5 sub-iterations, and each sub-iteration updates 5% of the pixel lines. Notice that, in the non-homogeneous sub-steps, only some of the pixel lines have been updated, and u has not been fully initialized. However, the filter of equation (4) fills in the values in the PSC for the pixel lines that have not been updated yet, so that any pixel line can be selected in the non-homogeneous step. After performing the interleaved steps, the NH-ICD algorithm resumes performing homogeneous and non-homogeneous steps alternately on the full image volume until convergence is achieved.

4. EXPERIMENT RESULTS

We apply the NH-ICD algorithm to the clinical chest data illustrated in Figure 9. This is a 3D reconstruction of 14 slices from axial sampling.

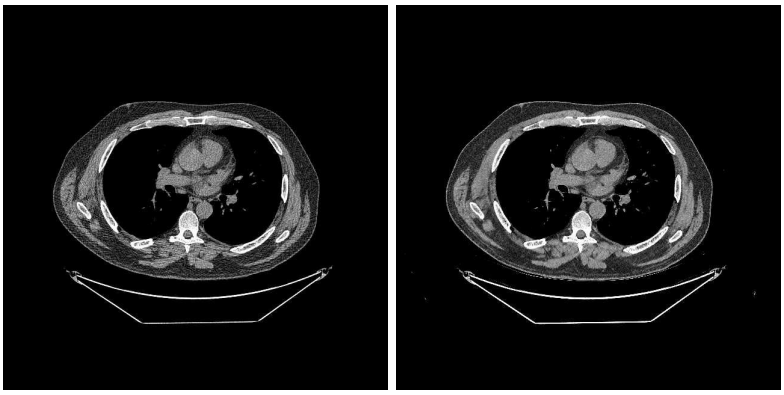
We define the residual error of a voxel to be the absolute difference between the voxel's current value and its fully converged value. At the m^{th} update, the residual error for voxel $x_{i,j,k}$ is computed as

$$e_{i,j,k}^{(m)} = x_{i,j,k}^{(m)} - x_{i,j,k}^{MAP} \tag{5}$$

where $x_{i,j,k}^{MAP}$ denotes the MAP estimate of the voxel's value. We also define the residual error for a pixel line (i, j) as the root mean squared value of all the voxels' residual error on the pixel line

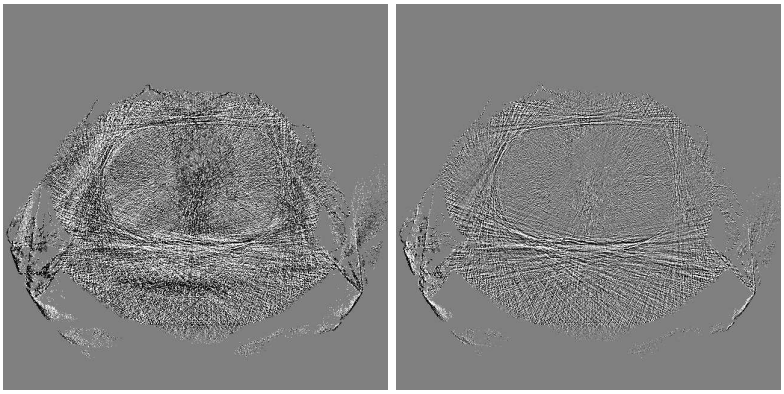
$$E_{i,j}^{(m)} = \sqrt{\frac{1}{K}\sum_{k=1}^{K} e_{i,j,k}^{(m)2}} \tag{6}$$

where K denotes the total number of voxels on the pixel line.

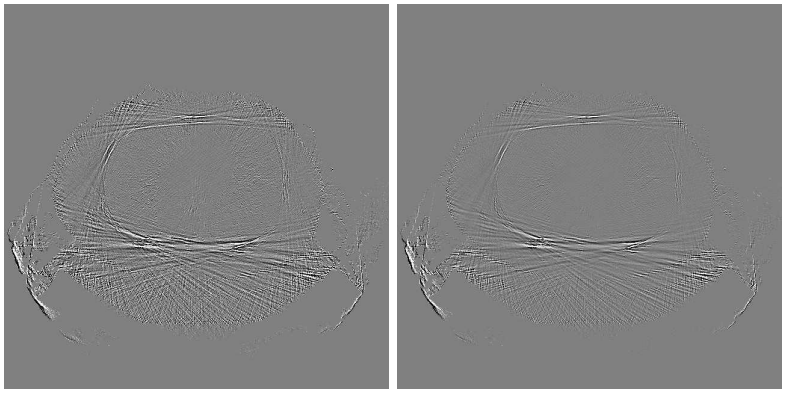


(a) (b)

Figure 9. The data set used in the experiment. (a) is the center slice of the initial FBP images, and (b) is the center slice of the reconstructed MAP images

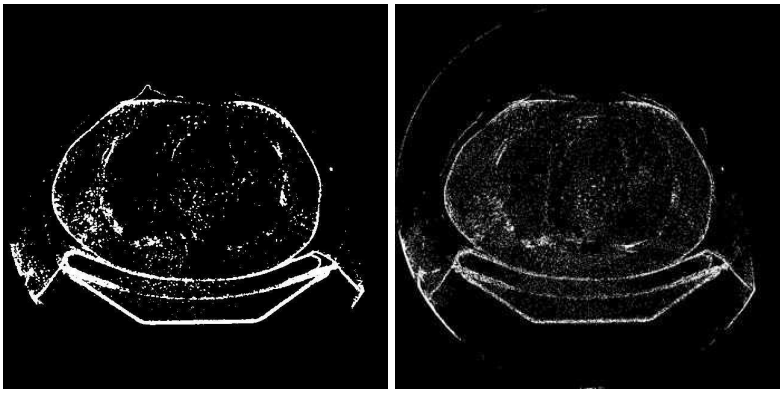


(a) (b)



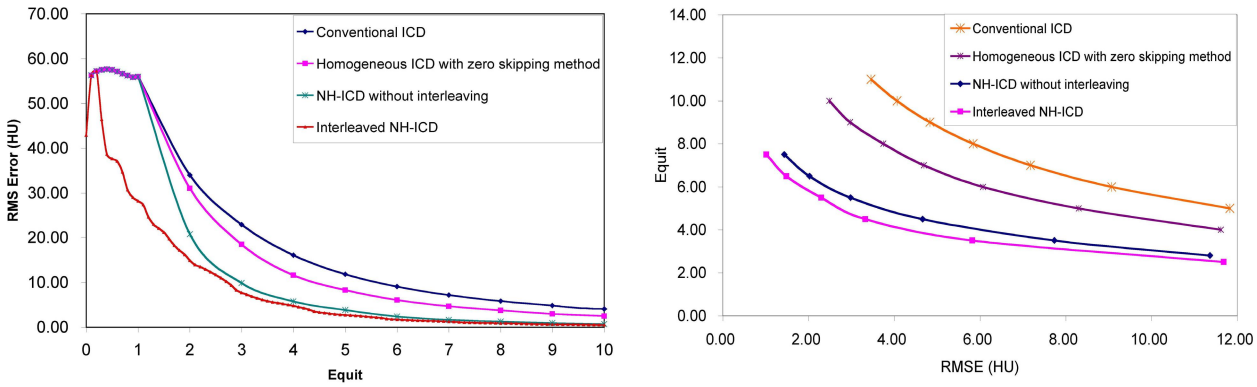
(c) (d)

Figure 10. Signed residual error images $x^{(n)} - \hat{x}^{MAP}$ of the center slice in the reconstruction volume for various number of iterations. (a) 1 iteration, (b) 3 iterations, (c) 5 iterations, and (d) 7 iteration. The evolution of the residual errors features spatial and temporal patterns providing insights into improvements of the algorithm for faster convergence. The gray scale ranges from -25 to +25 Hounsfield units.



(a) (b)

Figure 11. Correlation between the PSC and the true residual errors at the end of iteration $n = 1$. (a) shows the top 5% pixel locations with largest values of PSC, and (b) shows the top 5% pixel locations with largest residual error.



(a) (b)

Figure 12. Comparison of the convergence speed of different algorithms. (a) shows the convergence plot of RMSE of different algorithms, and (b) shows the computational cost required for each algorithm to achieve certain image quality level.

Figure 10 shows the image of residual errors of the center slice in the reconstructed volume for iterations 1, 3, 5, and 7 in the conventional ICD algorithm. The image of residual errors demonstrate that the convergence of ICD initialized from FBP images features spatially- and temporally-varying patterns.

Figure 11 shows the correlation between PSC and the pixel residual error at the end of iteration 1. Figure 11(a) shows the top 5% of the pixels with largest values of PSC. In Figure 11(b), the pixel residual errors are computed by (6), and the top 5% of the pixels with the largest residual errors are shown in the image. The correlation between the two images indicates that the PSC can be used to predict the pixel lines with large residual errors.

To compare the convergence speed of different algorithms, we must define an objective measure of computation. Due to the non-homogeneous nature of the NH-ICD algorithm, the concept of iteration is not directly applicable. Here we introduce the concept of equivalent iteration or "**equit**", as the number of voxel updates equal to the total number of voxels in the reconstruction volume. The computational cost of one equit is approximately equal to one conventional ICD iteration.

In the non-homogeneous step, we choose $p = 0.05$ and make the number of updates performed in each non-

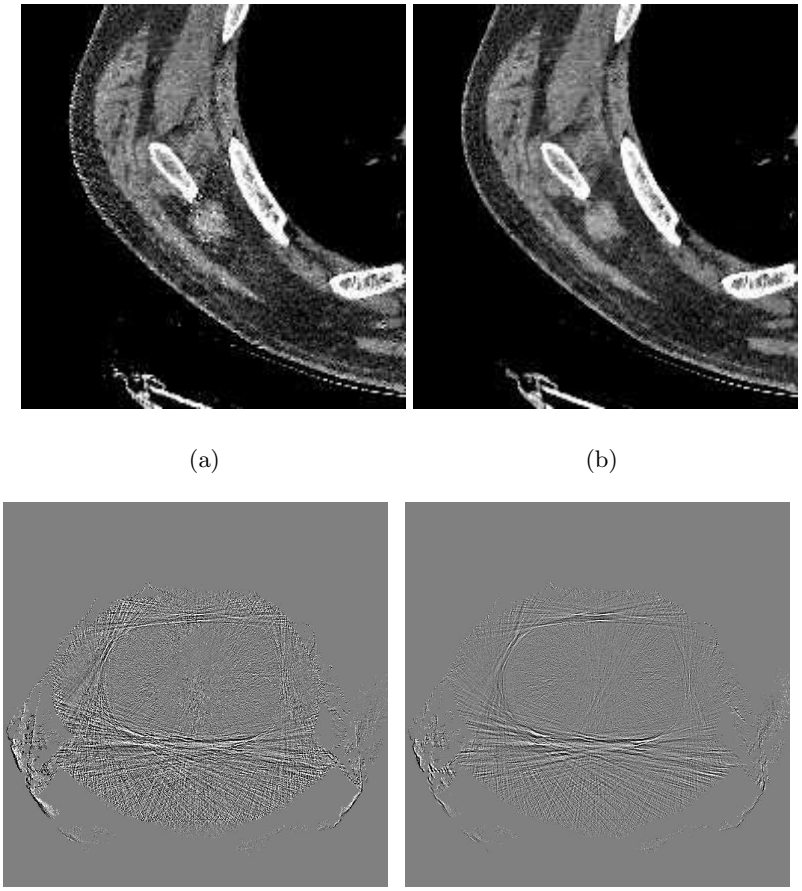


Figure 13. Comparison of image quality at fixed computational cost between NH-ICD and conventional ICD. The image quality are compared after 3 equits of update. (a) and (b) show the reconstructed image by the conventional ICD and interleaved NH-ICD respectively in a small region with a display window level of 0 HU and window width 400 HU. (c) and (d) compare the residual error in the center slice of the reconstructed image, with a display window level of 0 HU, and window width 60 HU. (c) is the result using the conventional ICD algorithm, and (d) is the result using the interleaved NH-ICD algorithm.

homogeneous step is approximately equal to that of a homogeneous step. Image quality is measured by the root mean squared value of the residual errors (RMSE) over the full image volume in Hounsfield units with the true MAP estimate by the result after 80 full iterations using the conventional ICD algorithm.

Improvements in convergence speed can be considered in two ways: either as a reduction in computational cost to reach a desired fixed image quality, or as a way to improve image quality for a fixed computational cost. Figure 12 compares the convergence plots of different algorithms applied, including the conventional ICD algorithm, the ICD algorithm with zero skipping method, the NH-ICD algorithm and the interleaved NH-ICD algorithm. Figure 12(a) shows the convergence of the RMSE as a function of equits. Notice that the interleaved NH-ICD algorithm provides the fastest convergnece speed. Especially at the first 2 equits, the RMSE of the interleaved NH-ICD algorithm is significantly smaller compared to the other algorithms, and its convergence speed is consistently faster than the other algorithms, too. Figure 12(b) shows the computation required for each algorithm to achieve a certain image quality level, where the horizontal axis is a measure of image quality using RMSE value in Hounsfield units, and the vertical axis is the computation time in equits. With the interleaved NH-ICD algorithm, convergence speed is improved by up to a factor of two. Figure 13 compares the

image quality achieved by the conventional ICD and the interleaved NH-ICD algorithm at a fixed computation time of 3 equits. Transient noise and artifacts are reduced with NH-ICD, as illustrated in the error images, providing better results for diagnosis.

5. CONCLUSION

We have presented a non-homogeneous approach to the ICD algorithm resulting in improvements in convergence speed of up to a factor of two in practice. The results are consistently robust throughout the convergence process. This improved performance may help reduce computation time to reach a desired level of image quality, or improve image quality for applications with limited computation time.

REFERENCES

1. J.-B. Thibault, K. Sauer, C. Bouman, and J. Hsieh, "Three-dimensional statistical modeling for image quality improvements in multi-slice helical CT," in *Proc. Intl. Conf. on Fully 3D Reconstruction in Radiology and Nuclear Medicine*, pp. 271–274, (Salt Lake City, UT), July 6-9 2005.
2. D. Politte, S. Yan, J. O'Sullivan, D. Snyder, and B. Whiting, "Implementation of alternating minimization algorithms for fully 3D CT imaging," in *Proc. of SPIE - The International Society of Optical Engineering*, **5674**, 2005.
3. A.Ziegler, Th.Köhler, and R.Proksa, "Noise and resolution in images reconstructed with FBP and OSC algorithms for CT," *Med. Phys.* **34**(2), pp. 585–598, 2007.
4. J.-B. Thibault, C. A. Bouman, K. D. Sauer, and J. Hsieh, "A recursive filter for noise reduction in statistical tomographic imaging," in *Proceedings of the SPIE/IS&T Symposium on Computational Imaging IV*, **6065**, (San Jose, CA), Jan. 16-18 2006.
5. J.-B. Thibault, C. A. Bouman, K. D. Sauer, and J. Hsieh, "New edge-preserving regularization for statistical reconstruction of clinical CT data," in *Proceedings of the Radiological Society of North America (RSNA)*, **SST14**, (Chicago, IL), Nov. 26 - Dec. 1 2006.
6. B. D. Man, S. Basu, J.-B. Thibault, J. Hsieh, J. Fessler, K. Sauer, and C. Bouman, "A study of different minimization approaches for iterative reconstruction in x-ray CT," in *Proc. of IEEE Nucl. Sci. Symp. and Med. Imaging Conf.*, (San Juan, Puerto Rico), October 23-29 2005.
7. C. Bouman and K. Sauer, "A unified approach to statistical tomography using coordinate descent optimization," *IEEE Trans. on Image Processing* **5**, pp. 480–492, March 1996.
8. L. Shepp and Y. Vardi, "Maximum likelihood reconstruction for emission tomography," *IEEE Trans. on Medical Imaging* **MI-1**, pp. 113–122, October 1982.
9. E. Ü. Mumcuoğlu, R. Leahy, S. Cherry, and Z. Zhou, "Fast gradient-based methods for Bayesian reconstruction of transmission and emission pet images," *IEEE Trans. on Medical Imaging* **13**, pp. 687–701, December 1994.
10. H. Hudson and R. Larkin, "Accelerated image reconstruction using ordered subsets of projection data," *IEEE Trans. on Medical Imaging* **13**, pp. 601–609, December 1994.
11. J.-B. Thibault, K. Sauer, and C. Bouman, "Newton-style optimization for emission tomographic estimation," *Journal of Electronic Imaging* **9**(3), pp. 269–282, 2000.