An Algorithm to Compute Independent Sets of Voxels for Parallelization of ICD-based Statistical Iterative Reconstruction

Sungsoo Ha and Klaus Mueller

***Abstract*—Statistical iterative reconstruction (SIR) algorithms show great potential for improving image quality in the reduced X-ray dose case. However, high computational cost and long reconstruction times remain obstacles preventing the use of SIR in practical application. Various optimization algorithms have been proposed to make the SIR algorithm parallelizable, in conjunction of improved convergence rate. Among them, identifying a set of de-coupled voxels within a coordinate descent (CD) optimization framework has shown good promise. However, so far this came at the price of additional complexity. We propose a novel algorithm that finds sets of independent voxels which can be updated simultaneously without introducing any additional complexity and within the original CD framework. We show that the cone-beam geometry gives rise to rather complex voxel distribution patterns, which however enable a higher degree of parallelism than the more regular decomposition patters suggested in the past. We also estimate the computational cost for ordered subset schemes which have advantages in the amount of parallelism that can be achieved per set, but lose performance due to the sequential execution of the sets overall.**

***Index Terms*—iterative reconstruction, coordinate descent, SIR**

I. Introduction

THE statistical iterative reconstruction (SIR) algorithm for computed tomography (CT) has been shown great potential to generate high quality images with less artifacts and noise even in reduced X-ray dosage. This capability mainly results from the statistical noise modeling that puts higher weight on reliable measurements while deemphasizing noisy measurements. SIR solves the following weighted least-squares (WLS) cost function:

$$\hat{\mathbf{x}} = \arg\min_{x \geq 0} \left\{ \frac{1}{2} (\mathbf{y} - \mathbf{A}\mathbf{x})^T \mathbf{W} (\mathbf{y} - \mathbf{A}\mathbf{x}) + R(\mathbf{x}) \right\}$$

where $\mathbf{y} = (y_1, \ldots, y_I)^{\boldsymbol{T}}$ is the vector of measured projection data, and I is the number of detector cells; $\mathbf{x} = \left(x_1, \ldots, x_J\right)^T$ is the vector of attenuation coefficients of the object subject to be reconstructed; $\mathbf{A}$ is the system matrix with the size of $I \times J$, $\mathbf{W}$ is diagonal matrix for statistical weighting, and $R(\mathbf{x})$ is regularization term.

However, the high computational cost for forward and back-projection operations still remains a challenging problem. To meet this challenge, with the consideration of modern parallel processors and properties of minimization algorithm, recent efforts have sought to find a solution between the conjugate gradient (CG)-based approach and the coordinate descent (CD)-based approach. The formal approach [1] can update all voxels simultaneously and thus is easy to parallelize, but it tends to converge very slowly; while the latter approach [2] can converge very quickly but it is hard to parallelize due to the sequential, single voxel update scheme.

Seeking a middle ground between the CG and CD approaches, the group coordinate descent (GCD) [3] and block-iterative coordinate descent (B-ICD) [4] algorithms have been developed. Both aim at finding parallel voxels within a transaxial (x-y) plane where, however, in most cases the coupling among those voxels remains high and so the acceleration is not overly dramatic. On the other hand, the axial block coordinate descent (ABCD) algorithm [5] looks for parallel voxels along the z-direction where the coupling amount is relatively lower than in x-y plane.

In this work, we take a rather different route to find sets of independent voxels for a given CT scanning geometry and system model – one that uses the GPU not only for reconstruction but also to compute a more accurate (but also more complicates) pattern of parallelizable voxels.

The remainder of this paper is as follows. In section II, we illustrate the intuition behind our method and then explain implementation details in section III. Section IV presents the results of our studies. Finally, section V ends with conclusions.

II. Motivation

A. Independency among voxels

The ICD-based statistical iterative reconstruction algorithm updates *a voxel once at a time* as follows:

$$x_j^{n+1} = \frac{\mathbf{A}_j^T \cdot \mathbf{W} \cdot (\mathbf{y} - \mathbf{A}\mathbf{x})}{\mathbf{A}_j^T \cdot \mathbf{W} \cdot \mathbf{A}_j} \qquad (1)$$

Equation 1 tells us that the correction, $\mathbf{W} \cdot (\mathbf{y} - \mathbf{A}\mathbf{x})$, is applied by the back projection with $\mathbf{A}_j$, which is a j-th column vector of the system matrix. In other words, an update of a voxel x_j is affected by a few line integrals that are corresponding to the non-zero elements of $\mathbf{A}_j$. This means that updating pixel A and pixel C in Figure 1 can be done at the same time because they

Sungsoo Ha and Klaus Mueller are with the Visual Analytics and Imaging Lab, Computer Science Department, Stony Brook University at Korea, Incheon, Korea (e-mail: {sunha, mueller}@sunykorea.ac.kr).

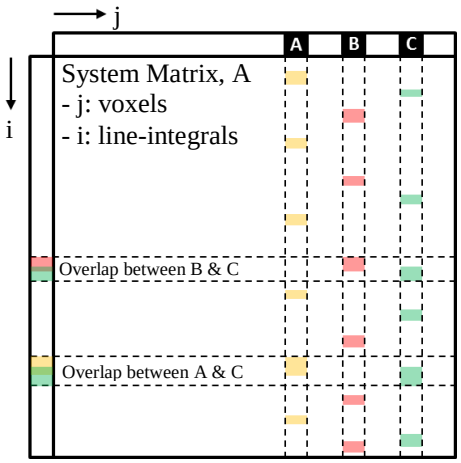

Figure 1. Example of independency among voxels

depend on independent line integrals; while pixel B contains shared line integrals with both A and B, and thus should be updated separately. We say these voxels are *independent* of each other and call a set of such voxels an *independent group*.

B. Knapsack problem

Finding such an *independent* group can be equally regarded as finding a subset of column vectors from the CT system matrix such that there are no overlaps in the positions of non-zero elements among the selected column vectors as shown in Figure 1.

Suppose a_j is a binary vector format of $\mathbf{A}_j$ where 1 indicates non-zero weight and G is a set of binary vectors. Then finding a set, G, becomes a combinatorial optimization problem as followings:

$$\min \quad \mathrm{ZERO}\left\{\bigcup_{g\in G} g\right\} \quad s.t. \quad \begin{matrix} G = \{a_k | 1 \le k \le \mathrm{J}\} \\ a_m \cap a_n = \mathbf{0} \\ \forall a_m a_n \in \mathrm{G}, \mathrm{m} \ne \mathrm{n} \end{matrix} \qquad (2)$$

Here, ZERO(·) is an operator counting the number of zero elements in a binary vector and **0** is a zero vector. This kind of problem is known as the *knapsack* problem and it is, unfortunately, a combinatorial *NP-hard* problem. Thus, instead of solving Equation 2 exactly, we solve it using a heuristic algorithm, called *first-fit decreasing (FFD) algorithm*.

With the first-fit decreasing (FFD) algorithm, we first sort all vectors, a_j, in decreasing order according to the number of non-zero elements in the vectors. After that, we find a set, G, by greedily adding all a_j in first-fit order, if and only if it satisfies the conditions in Equation 2.

III. Implementation

We now describe how the first-fit decreasing (FFD) algorithm is applied to find independent groups for a given CT geometry and CT system model. The algorithm consists of two steps: first, we identify the number of non-zero elements for each voxel and then run the FFD over the voxels.

A. Counting

Before applying the FFD to find *independent* groups, we first need to compute the total number of detector cells that interact with each voxel in all views. For example, in a view, the 8-vertices of a voxels are projected onto the detector to roughly estimate the maximum intersection area in the detector. Then, we count the number of detector cells within the area according to the forward (or system) model. With the line integral models [6][7], a ray is a zero-width single line that connects the X-ray source to the central point of a detector cell; for the area integral models [8][9], a volumetric ray is traced. Figure 3 illustrates the differences between line integral and area integral models (using 2D fan-beam geometry for ease of drawing).

B. Finding independent groups

To find an independent group, the FFD algorithm is utilized. Suppose there is a list of voxels and an indicator map. The indicator map shows what detector cells are related to the voxels in the current group. Then, FFD begins with picking a voxel from the list that has the largest count number that we computed before. The selected voxel is then projected onto the detector in all views to determine detector cells that are related to the voxel. This is a similar process than described in the previous step (see also Figure 3). Then, we check if any one of the detector cells is already related with another voxel that has been added into the current group previously. This checking is done by using the indicator map. If none of the cells have been found before, the voxel is added to the group because it is *independent* to all voxels previously added. Before inspecting the next voxel, we remove some voxels from the list that are dependent to the newly added voxel. There are two types of dependent voxels: 1) voxels on the projection path of the newly added voxel and 2) its neighbor voxels. The first type of dependent voxel is removed by tracing each ray that connects the

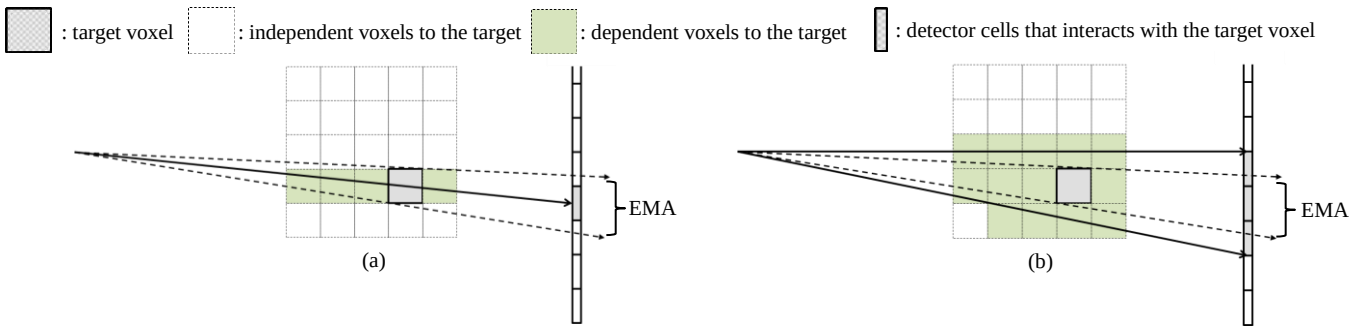

Figure 2. Independent and dependent voxels according to the system model. (a) line integral model and (b) area integral model in 2D fan-beam geometry. The estimation of maximum area (EMA) where a voxel gives some contribution in a projection will be the same in both models. However, according to the system model, the grouping result can be different.

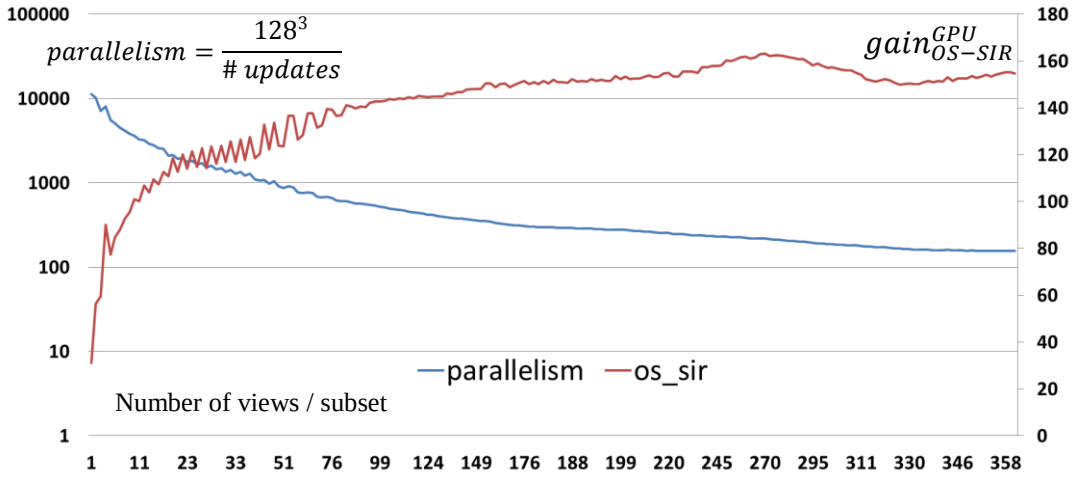

Figure 3. Relationship between expected (ideal) parallelism and estimated gain of GPU-accelerated OS-SIR. Note that parallelism is plotted over logarithmic y-axis for better view.

X-ray source to each detector cell that is related to the newly added voxel as shown in Figure 3. The second type depends on the regularization term that is going to be used in SIR. This early elimination of dependent voxels from the list helps to reduce the total running time of the FFD process. The process will be repeated until the list becomes empty.

To find all independent groups, we simply need to maintain a global voxel list. After finding an independent group, voxels included in the group will be removed from the global list. Then, the updated list will be used to find another independent group. This procedure is repeated until the global list becomes empty.

IV. Results

To test the proposed grouping algorithm, we assume a cone-beam CT geometry. The object subject to be reconstructed is set to 128×128×128 with 1×1×1 mm of voxel size. The distance from the X-ray source to rotation axis and to the center of a flat detector is set to 600 mm and 1000 mm, respectively. The dimension of the flat detector is set to 512×512 with 1×1 mm cell size. It is worth noting that all CBCT environment parameters (including detector size) are determined such that all voxels can be projected onto at least 1 detector cells in all tested views. Lastly, we assume a line integral model and a quadratic regularization term in SIR. These two assumptions are needed to make a decision what voxels belong to the independent or dependent group as explained in section III.B.

We ran the proposed algorithm varying the number of views from 1 to 360 and, in each case, the projection angles were distributed uniformly over 360 degrees. Table I shows the outcomes of the proposed method for two extreme cases, single view and 360 views in a set. Especially, with 360 views, the number of updates that is required per iteration during ICD-based SIR is reduced from $128^3 (= 20{,}971{,}152)$ to 13,569. In this case, the expected speed-up with ideal GPU implementation is 155. More importantly, the proposed grouping method finds 154 independent voxels on average that do not have any coupling with any other voxels. This means that updating those voxels simultaneously does not require any other additional computational complexity but can be done just like a single voxel update in original ICD algorithm.

TABLE I. Statistics of independent voxel grouping

# Views	# updates	Max. size of independent group	Avg. size of independent group
1	187	16,186	11,214
360	13,569	449	154

The ratio of total number of voxels to the number of updates obtained from with method is considered the theoretical parallelism we can achieve with an ideal GPU implementation. Figure 3 shows it in logarithmic scale. Suppose that we have 360 views in total and we apply the ordered-subsets (OS) algorithm [10] by varying the number of views per subset. Then, the total gain from OS with ideal GPU implementation on ICD-based SIR is estimated as following:

$$gain_{OS_{SIR}}^{GPU} = \frac{parallelism}{(360\ /\ \#\ of\ views\ per\ subset)}$$

Here, the benefit obtained from a parallel implementation of SIR is reduced by the number of subsets that must be updated sequentially. The result is shown in Figure 3. Considering the improved convergence rate with OS, around 100 views per subset could be good choice. However, in the real world, there are many constrains such as memory bandwidth between CPU and GPU, the special GPU architecture and programming model and so on. And all of these can possibly change our expectation. Future work will determine optimal computational performance of GPU-accelerated SIR [11].

The distribution patterns for the independent voxel groups are quite complex. A set of visualizations is presented in Figure 4 where we assign each group a different color from a rainbow scale. This figure shows these patterns for three volume slices – bottom, middle, and top along the z-axis – and for different numbers of views. We observe as we look at the bottom and top slices for greater view numbers – parallelizable voxels tend to gather together, creating circular shapes. This is due to the characteristics of the cone-beam – rays that are tangential to the circle do not share any voxels. We also observe that the dis-

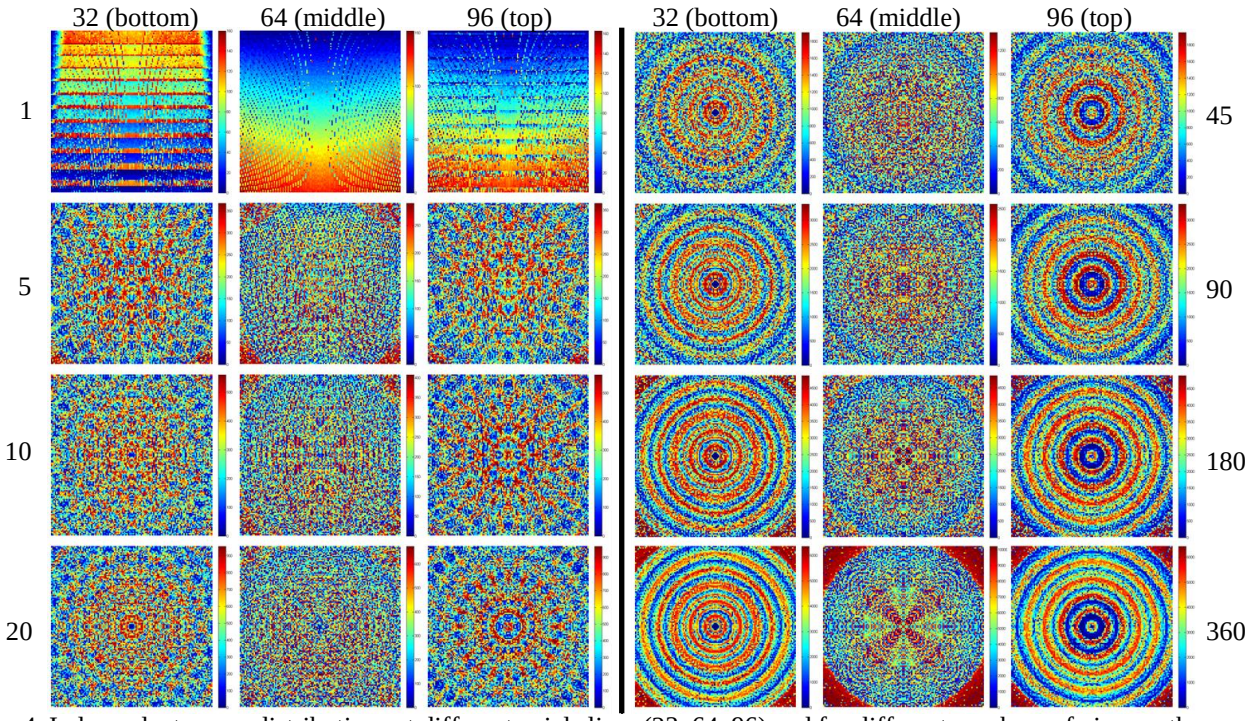

Figure 4. Independent group distributions at different axial slices (23, 64, 96) and for different numbers of views – the numbers along both sides represent the number of views. We color the voxels according to the rainbow color maps shown alongside each image to represent different groups. The groups with blue color tones are generated earlier than groups with red color tones.

tribution of independent groups on the bottom and top are reverse (the blue region in the top slice is the red region in the bottom). This is strong evidence that the proposed algorithm gathers independent voxels within an optimal distance.

V. Conclusion

We have described a new method to determine parallelizable voxels for GPU-accelerated ICD-based SIR. Not only can our method provide more parallelism than existing methods, it has several more advantages. First, since it finds completely independent voxels groups, unlike other methods it does not introduce extra complexity in the SIR procedure. Second, having accurate parallel voxel distribution as shown in Figure 4, we can devise a good strategy for a GPU implementation of SIR. Lastly, the algorithm is not limited for use in cone-beam CT geometry – it can work in any beam geometry.

As a disclaimer, the gains we can expect from the grouping results are only obtainable in the ideal case. There are many practical aspects that can reduce these expected gains in an actual GPU implementation, for example, data transfer between CPU and GPU memory, compute capability of GPU device, and so on. Our next step in this study is to apply our grouping result to an actual GPU-accelerated SIR framework. We hope that we can get sufficiently high speed-up gains that are not too far from the expected gains estimated in this paper. We will also study how the convergence rate changes with our scheme.

Acknowledgments

This research was partially supported by NSF grant IIS-11732 and the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the 'IT Consilience Creative Program (ITCCP)' (NIPA-2013-H0203-13-1001) supervised by NIPA (National IT Industry Promotion Agency).

References

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

[2] K. Sauer and C. Bouman, "A local update strategy for iterative reconstruction from projections," IEEE Trans. Sig. Proc., 41(2):534-48, February 1993.

[3] J. A. Fessler, E. P. Ficaro, N. H. Clinthorne, and K. Lange, "Grouped-coordinate ascent algorithms for penalized-likelihood transmission image reconstruction," IEEE Trans. Med. Imag., 16(2):166-75, April 1997.

[4] T. M. Benson, B. K. B D. Man, L. Fu, and J-B, "Thibault. Block-based iterative coordinate descent," In Proc. IEEE Nuc. Sci. Symp. Med. Im. Conf., 2010.

[5] J. A. Fessler and D. Kim, "Axial block coordinate descent (ABCD) algorithm for X-ray CT image reconstruction," in Proc. Intl. Mtg. on Fully 3D Image Recon. In Rad. And Nuc. Med, 2011, pp. 262-5.

[6] R.L. Siddon, "Fast calculation of the exact radiological path for a three-dimensional CT array," Med. Phys. 12(2):252-255, 1985.

[7] P.M. Joseph, "An improved algorithm for reprojecting rays through pixel images." IEEE Trans Med Imaging 1(3):192-196, 1983.

[8] B. De Man and S. Basu, "Distance-driven projection and backprojection in three dimensions," Phys. Med. Biol. 49(11), 2463-2475, 2004.

[9] Y. Long, J. A. Fessler, and J. M. Balter, "3D forward and back-projection for x-ray CT using separable footprints," IEEE Trans. Med. Imaging 29(11), 1839-1850, 2010.

[10] C. Kamphuis and F. J. Beekman, "Accelerated iterative transmission CT reconstruction using an ordered subsets convex algorithm," IEEE Trans. Med. Imag., 17(6):1001-5, December 1998.

[11] F. Xu, W. Xu, M. Jones, B. Keszthelyi, J. Sedat, D. Agard, and K. Mueller, "On the efficiency of iterative ordered subset reconstruction algorithms for acceleration on GPUs," Comput. Methods Programs Biomed. doi:10.1016/j.cmpb.2009.09.003 2009.