9-12-2017

Equations' Derivations In The Three-Dimensional Super-Voxel Calculations

Xiao Wang
Harvard Medical School/Boston Children's Hospital, Xiao.wang2@childrens.harvard.edu

Charles A. Bouman
Purdue University, bouman@purdue.edu

Samuel Midkiff
Purdue University, smidkiff@purdue.edu

Equations' Derivations In The Three-Dimensional Super-Voxel Calculations

Xiao Wang, Charles Bouman, and Samuel Midkiff

CONTENTS

LIST OF FIGURES

X. Wang was with Harvard Medical School / Boston Children's Hospital. C. Bouman and S. Midkiff was with the Department of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, 47907.

Equations' Derivations In The Three-Dimensional Super-Voxel Calculations

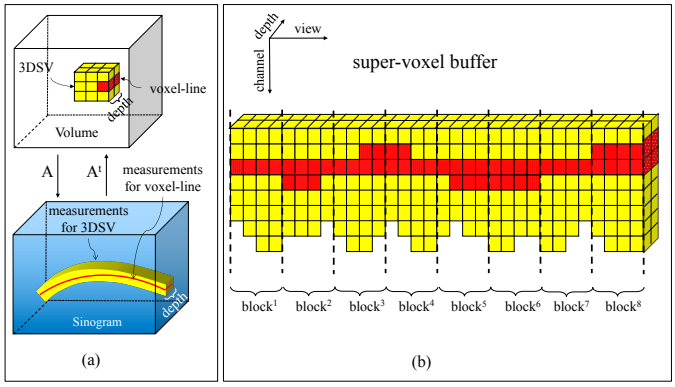

Fig. 1: (a) Shows a 3DSV of size $3 \times 3 \times 2$. Measurements for the 3DSV follow a sinusoidal band in the sinogram. (b) A super-voxel buffer. Notice that measurements for the red voxel-line trace curves up and down in the super-voxel buffer with a small amplitude.

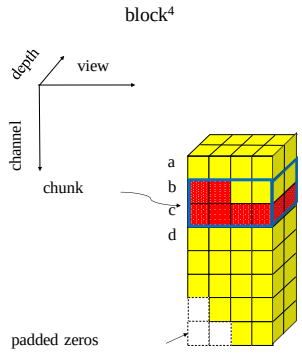

Fig. 2: $block^4$ of the super-voxel buffer with padded zeros and chunk design.

***Abstract*—Model-Based Iterative Reconstruction (MBIR) is a widely-explored fully 3D Computed Tomography (CT) image reconstruction technique that has a large impact on the image reconstruction community. The slow computation speed for MBIR, however, is a bottleneck for scientific advancements in fields that use imaging, such as materials. A recently proposed algorithm, *Non-Uniform Parallel Super-Voxel* (NU-PSV), utilizes the concept of *Three-Dimensional Super-Voxel* (3DSV) and *Block-Transposed Buffer* (BTB) [1]. Experiments in the past show that the NU-PSV algorithm significantly improves the computation speed for MBIR by regularizing data access pattern, reducing cache misses, enabling more parallelism and speeding up algorithmic convergence. This technical report serves as an auxiliary appendix to publication [1]. In this technical report, we demonstrate the theoretical calculations related to a BTB.**

I. Introduction

This section provides definitions for *Three-Dimensional Super-Voxel* (3DSV), *block*, *chunk* and *Block Transposed Buffer* (BTB). Most content of this section has appeared in [1].

We define a 3DSV as a group of voxels in the shape of a rectangular cuboid in the volume. Figure 1(a) shows a 3DSV of size $3 \times 3 \times 2$, where the width and height are 3 and the depth is 2.

Figure 1(a) also illustrates the parallel-beam Computed Tomography (CT) measurements associated with a 3DSV as a yellow sinusoidal band in the sinogram. The measurements for a single voxel-line within the 3DSV are shown as a red trace within the sinusoidal band. Notice that both the sinusoidal band and the red voxel-line trace are three-dimensional with depth 2.

To further improve cache locality and enable hardware prefetching, the 3DSV's measurements can be copied from the sinusoidal band to a memory buffer, called the *super-voxel buffer*. Figure 1(b) shows the layout of a super-voxel buffer for the 3DSV in Figure 1(a). Since the sinusoidal band is three-dimensional, the super-voxel buffer also has three dimensions (channel, view, and depth), with measurements stored in memory in the order of channel, view and depth, i.e., adjacent channel entries are adjacent in memory.

Within the super-voxel buffer, all measurements for a voxel-line, shaded in red in Figure 1(b), are accessed along a sinusoidal trace with depth 2, and with a much smaller amplitude than in the sinogram. Nevertheless, the combination of remaining trace amplitude, varying trace and super-voxel buffer width (as in Figure 1(b)) still lead to irregular data access.

Before we present the technical solution, we start by defining a series of data structures. We define a *block* to be a fixed number of contiguous views within a super-voxel buffer, and denote the i^{th} block by $block^i$. Figure 1(b) shows a super-voxel buffer consisting of 8 blocks, with each block composed of 4 views. In addition, we define a *chunk* as a rectangular cuboid, circumscribing the sinusoidal voxel-line trace in a block. Figure 2 illustrates $block^4$ of the super-voxel buffer, and the chunk contained within $block^4$ is outlined with a bold blue line. To have a constant trace width, all the measurements in a chunk are accessed when processing a

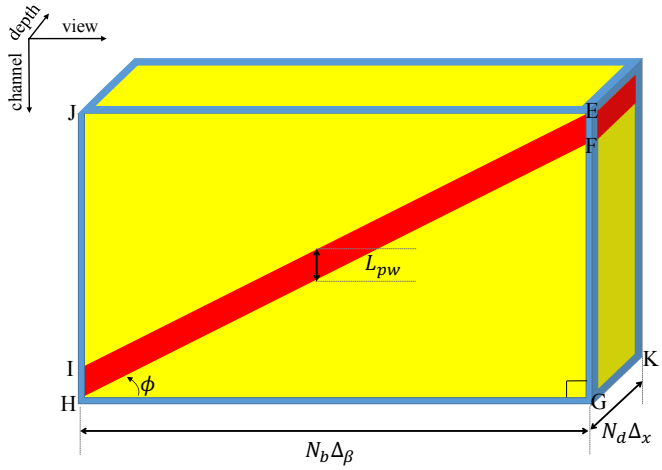

Fig. 3: Demonstrates the dimensions of a chunk. The voxel-line trace is colored in red and the redundant measurements are colored in yellow.

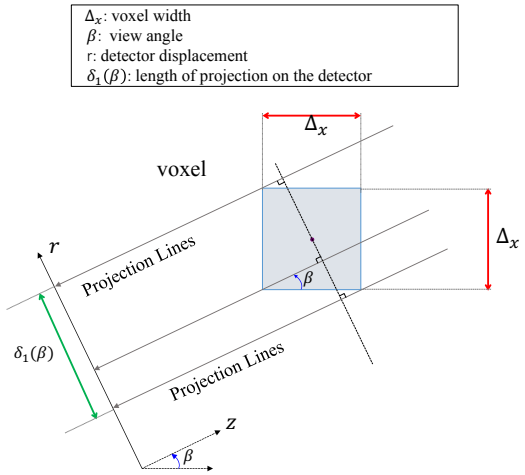

Fig. 4: Shows a voxel modeled as a square. The length of projection, $\delta_1(\beta)$, is shown as the green bar.

voxel line, even though only a portion of them are needed. For convenience, we call the measurements required for a voxel-line update *essential measurements*, shown in red in Figure 2. The unneeded measurements for the voxel-line update are called *redundant measurements*, shown in yellow in Figure 2.

In addition to the chunk design, each block is padded with zeros so that the buffer width, namely the number of channels of a block, is constant. Since the voxel trace and the buffer widths are both constant, measurements in a chunk have completely regular data access.

To further improve SIMD performance, we define the *Block Transposed Buffer* (BTB) as a block-wise *(i.e., block-by-block)* counter-clockwise rotation of 90° about the depth direction of the super-voxel buffer, i.e. a concatenation of transposed blocks. After transposition, the axes of channel and view are swapped. All measurements within the block are laid out along the view direction in memory.

II. Equations' Derivations

In this section, we show the theoretical calculation for the average number of regular memory accesses in a chunk.

Figure 3 is a simplified figure that shows a chunk in a BTB, where the voxel-line trace (essential measurements) is colored in red and redundant measurements are colored in yellow. The chunk length, HG, equals to $N_b\Delta_\beta$, where N_b is the block size, β is the view angle, ranging from 0 to π, and Δ_β is the view angle spacing; the chunk width, GK, is $N_d\Delta_x$, where N_d is the depth of the 3DSV and Δ_x is the voxel width; the chunk height, EG, is $L_{pw} + m(\phi)N_b\Delta_\beta$, where L_{pw} is the average voxel trace width, ϕ is the angle between the voxel trace and the view direction, $m(\phi)$ is the average voxel trace absolute slope. Relating to Figure 3, L_{pw} equals to the length of EF, $m(\phi)N_b\Delta_\beta$ equals to the length of FG, Then, $L_{pw} + m(\phi)N_b\Delta_\beta$ equals to the chunk height, EG. With the dimensions of the chunk, the volume of the chunk, denoted as V_c, can be computed as:

$$V_c = N_b\Delta_\beta N_d\Delta_x \left(L_{pw} + m(\phi)N_b\Delta_\beta\right) \tag{1}$$

Then, the average number of regular memory accesses in a chunk, denoted as N_{run}, can then be computed as:

$$N_{run} = \frac{V_c}{\Delta_\beta\Delta_x\Delta_d} \tag{2}$$

where Δ_d is the detector channel spacing, $\Delta_\beta\Delta_x\Delta_d$ is the size of a single entry in the BTB, and $\frac{V_c}{\Delta_\beta\Delta_x\Delta_d}$ represents the number of element entries in a chunk. By plugging the expression of V_c to Equation (2), we can then get:

$$N_{run} = \frac{N_d N_b \left(L_{pw} + m(\phi)N_b\Delta_\beta\right)}{\Delta_d}. \tag{3}$$

In this equation, N_d and N_b are known constants, chosen to optimize computing performance. Δ_β and Δ_d are also known constants about the dataset. L_{pw} and $m(\phi)$, however, are unknown parameters. To compute Equation (3), we must first compute these two parameters.

Computing parameter L_{pw} To compute L_{pw}, we model a voxel to be a square with voxel width Δ_x, as shown in Figure 4. In addition, we denote $\delta_1(\beta)$, shown as a green bar in Figure 4, as the length of projection on the X-ray detector at view angle β. $\delta_1(\beta)$ can be computed as in [2]:

$$\delta_1(\beta) = \begin{cases} \sqrt{2}\Delta_x \cos(\frac{\pi}{4} - \beta), & \text{if } \beta \in [0, \frac{\pi}{2}) \\ \sqrt{2}\Delta_x \sin(\beta - \frac{\pi}{4}), & \text{if } \beta \in [\frac{\pi}{2}, \pi) \end{cases} \tag{4}$$

Unfortunately, Equation (4) is not ideal because $\delta_1(\beta)$ must be approximated to be a multiple of Δ_d in real applications.

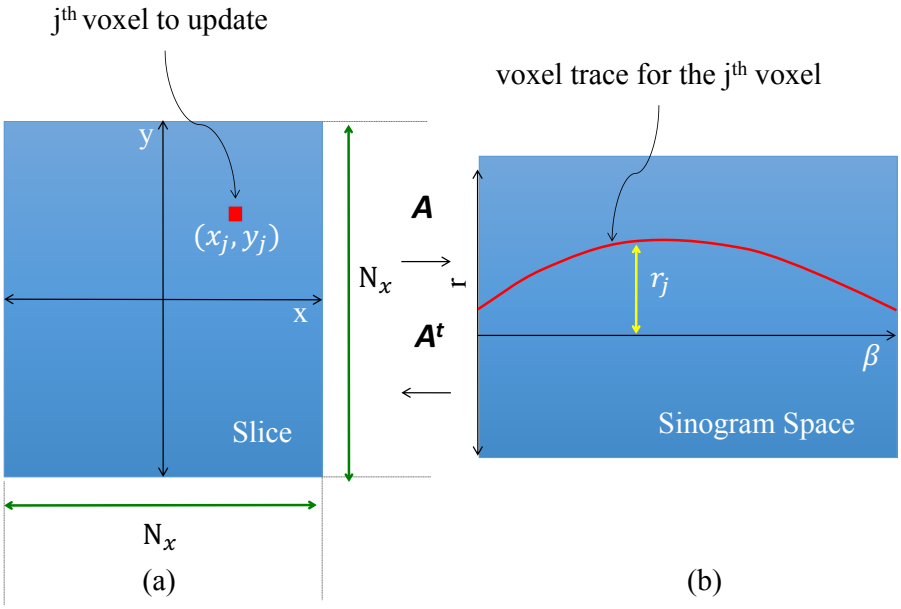

Fig. 5: (a) The red square in the slice is the j^{th} voxel, whose coordinate is (x_j, y_j). (b) The red trace is the measurements for the j^{th} voxel. The yellow bar r_j represents the voxel trace amplitude in the sinogram space.

To offset this error, a constant term Δ_d is added to Equation (4) and the new equation becomes:

$$\delta_1(\beta) \approx \begin{cases} \sqrt{2}\Delta_x \cos(\frac{\pi}{4} - \beta) + \Delta_d, & \text{if } \beta \in [0, \frac{\pi}{2}) \\ \sqrt{2}\Delta_x \sin(\beta - \frac{\pi}{4}) + \Delta_d, & \text{if } \beta \in [\frac{\pi}{2}, \pi) \end{cases} \tag{5}$$

With Equation (5), L_{pw} can then be computed to be the average value for $\delta_1(\beta)$, shown as:

$$L_{pw} = \frac{\int_0^{\pi} \delta_1(\beta)\, d\beta}{\pi} \approx \frac{4\Delta_x}{\pi} + \Delta_d \tag{6}$$

Computing parameter $m(\phi)$ For the j^{th} voxel in a slice, illustrated as a red square in Figure 5(a), we denote its coordinate as (x_j, y_j). In addition, we denote its voxel trace amplitude in the sinogram at view angle β as $r_j(\beta)$, shown as a yellow bar in Figure 5(b). Analytically, $r_j(\beta)$ can be expressed as [2]:

$$r_j(\beta) = y_j \cos\beta - x_j \sin\beta \tag{7}$$

and the voxel trace slope in the sinogram at view angle β is then:

$$r_j^{'}(\beta) = -y_j \sin\beta - x_j \cos\beta \tag{8}$$

Therefore, the average absolute slope for a voxel trace in the sinogram space, denoted as $\tilde{m}$, can be computed as:

$$\tilde{m} = \frac{\int_{\frac{-N_x}{2}}^{\frac{N_x}{2}} \int_{\frac{-N_x}{2}}^{\frac{N_x}{2}} \int_0^{\pi} |r_j^{'}(\beta)|\, d\beta\, dx_j\, dy_j}{N_x N_x \pi} \tag{9}$$

Where N_x is the slice dimension. To simplify Equation (9), we use polar coordinate and we let $x_j = -\gamma\cos\beta$, $y_j = -\gamma\sin\beta$ and $\gamma = \sqrt{x_j^2 + y_j^2}$. Therefore,

$$\begin{aligned} \tilde{m} &= \frac{8\int_0^{\frac{\pi}{4}} \int_0^{\frac{N_x}{2\cos\beta}} 2\gamma^2\, d\gamma\, d\beta}{N_x N_x \pi} \\ &= \frac{N_x}{3\pi}\left(\sqrt{2} + \ln(1+\sqrt{2})\right) \end{aligned} \tag{10}$$

Similar as before, Equation (10) must compensate for errors. Therefore, a constant term is added to Equation (10) and $\tilde{m}$ is approximated to be:

$$\tilde{m} \approx \frac{N_x}{3\pi}\left(1 + \sqrt{2} + \ln(1+\sqrt{2})\right) \tag{11}$$

After measurements are copied from the sinogram space to a BTB, all voxel traces are flattened with a much smaller amplitude and slope. To calculate the voxel trace average absolute slope in the BTB, $m(\phi)$, a 3DSV can be viewed as a slice, whose length and height is $N_{wh}\Delta_x$, where N_{wh} is the number of voxels along the width and height of the 3DSV. Therefore, $m(\phi)$ can be calculated by replacing N_x with $N_{wh}\Delta_x$ in Equation (11):

$$m(\phi) \approx \frac{N_{wh}\Delta_x}{3\pi}\left(1 + \sqrt{2} + \ln(1+\sqrt{2})\right) \tag{12}$$

After plugging Equations (6) and (12) into Equation (3), we can then get:

$$N_{run} = \left(\frac{N_{wh}\Delta_x N_b \Delta_\beta}{3\pi\Delta_d} \left(1 + \sqrt{2} + \ln(1+\sqrt{2})\right) + \frac{4\Delta_x}{\pi\Delta_d} + 1 \right) N_d N_b \tag{13}$$

Since Δ_β can also be computed as $\frac{\pi}{N_v}$ [2], the full analytical expression for N_{run} becomes:

$$N_{run} = \left(\frac{N_{wh}\Delta_x N_b}{3\Delta_d N_v} \left(1 + \sqrt{2} + \ln(1+\sqrt{2})\right) + \frac{4\Delta_x}{\pi\Delta_d} + 1 \right) N_d N_b \tag{14}$$

If we let constant $C_1 = \frac{\Delta_x}{3\Delta_d N_v} \left(1 + \sqrt{2} + \ln(1+\sqrt{2})\right)$ and constant $C_2 = \frac{4\Delta_x}{\pi\Delta_d} + 1$, then N_{run} can be simply stated as follows:

$$N_{run} = (N_{wh} C_1 N_b^2 + C_2 N_b) N_d \tag{15}$$

where $N_{wh} C_1 N_b^2 N_d$ is the number of regular memory access for redundant measurements and $C_2 N_b N_d$ is the number of regular memory access for essential measurements. In this equation, we can note that N_{run} is proportional to N_{wh} and N_d, and is proportional to the square of N_b.

The percentage of essential measurements, E_c, can then be computed as the ratio of essential measurement entries to N_{run}:

$$E_c = \frac{C_2 N_b N_d}{N_{run}} = \frac{C_2}{N_{wh} C_1 N_b + C_2} \tag{16}$$

In this equation, we can note that E_c is inversely proportional to N_{wh} and N_b.

III. Conclusions

This technical report shows the theoretical calculation for the average number of regular memory accesses in a chunk, as well as the percentage of essential measurements. In addition, this technical report also shows what factors contribute to the efficiency of calculations. Therefore, scientists can gain insights on how to optimize NU-PSV's algorithmic performance through optimizing its parameters.

References

[1] X. Wang, A. Sabne, P. Sakdhnagool, S. Kisner, C. A. Bouman, and S. P. Midkiff, "Massively parallel 3d image reconstruction," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC'17)*, Nov 2017.

[2] X. Wang, "High Performance Tomography," Ph.D. dissertation, School of Electrical and Computer Engineering, Purdue University, the United States, 2017.