FAST VOXEL LINE UPDATE FOR TIME-SPACE IMAGE RECONSTRUCTION

Xiao Wang[†] *K. Aditya Mohan*[†] *Sherman J. Kisner*[⋆] *Charles Bouman*[†] *Samuel Midkiff*[†]

[†] School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907, USA
[⋆] High Performance Imaging LLC, West Lafayette, IN 47906, USA

ABSTRACT

Model based iterative reconstruction (MBIR) algorithms have been used to greatly improve image quality and temporal resolution in synchrotron based time-space Computed Tomography (CT). Among the various optimization methods that have been used for MBIR, iterative coordinate descent (ICD) has relatively lower computational requirements because it converges fast. In spite of that, long execution time remains a barrier for the widespread use of MBIR. In this paper, we present a new data structure, called *VL-Buffer*, for time-space reconstruction that significantly improves the cache locality while retaining good parallel performance. Experimental results show an average speedup of 40% using VL-Buffer.

***Index Terms*—** Time-Space Image Reconstruction, Multicore, High Performance Computing

1. INTRODUCTION

Synchrotron based X-rays are used for 3D imaging of material samples in a wide range of disciplines, including biology [1] and materials science [2]. In synchrotron based X-ray CT tomography, there are two general categories of reconstruction methods: direct methods such as filtered back projection (FBP) and iterative methods such as MBIR [3]. MBIR results in better reconstruction quality and fewer artifacts than FBP [4]. However, MBIR has a much higher computational cost than direct methods [3], and this high computation requirement has been a barrier to the widespread use of MBIR.

There are two approaches to implement the MBIR optimization: simultaneous methods and coordinate descent methods. Simultaneous methods [5, 6, 7, 8] work by projecting and back-projecting the entire image to the sinogram space. The advantage of simultaneous methods is that they can update all voxels simultaneously which facilitates parallelism, but the disadvantage is that they have relatively slower convergence [5]. Techniques, such as preconditioning [7] and ordered subsets [9], are used to speed up the convergence. However, preconditioning methods can be sensitive to geometry and ordered subset methods generally slow down global convergence.

Among iterative methods, ICD [10, 11, 12] has been shown to have rapid and robust convergence for a wide variety of geometries, applications and image models. Instead of hundreds of iterations required in the simultaneous methods [5], ICD [12] typically converges in 3 to 6 iterations [12, 10]. However, while ICD has rapid convergence, it requires operations that are more difficult to parallelize [13, 14, 15]. Moreover, ICD exhibits poor cache locality because the data layout requires memory to be accessed in sinusoidal patterns [16, 17], as shown in Sec.2.1.

Typically, high performance CT image reconstruction can be summarized into two competing challenges: (1) increasing the cache locality, and (2) increasing the parallelism [17]. In this paper, we call voxels that are far away from each other as "loosely coupled" and neighboring voxels as "strongly coupled". Loosely coupled voxels have fewer measurements in common and fewer dependencies in the voxel updates, enabling good parallel performance. However, loosely coupled voxels also suffer from poor cache locality. On the other hand, strongly coupled voxels have higher cache locality and more measurements in common. But they suffer from poor parallel performance due to the data dependencies. Therefore, the best solution is to find certain voxels that allow increased cache locality without negating parallel performance. In this paper, we propose a new data structure, the voxel line memory buffer (*VL-Buffer*), to meet both goals.

We make the following contributions:

1. We describe the performance issues inherent in voxel lines.
2. We propose the idea of the VL-Buffer to increase cache locality and prefetching. In addition, VL-Buffer reads non-coalesced measurements in a coalesced way.
3. We show experimental results that VL-Buffer leads to an average 40% speedup on each core. In addition, VL-Buffer does not worsen parallel performance.

In Sec. 2.1, we provide background information and we review the concept of *voxel line*. In Sec. 2.2, we present the mathematical formulation of MBIR. In Sec. 3, we describe how the VL-Buffer enables fast voxel line updates. Finally

Xiao Wang, Sherman Kisner, Charles Bouman and Samuel Midkiff were supported by the U.S. department of Homeland Security under SBIR Award D15PC00110 and by the Indiana Economic Development Corporation (IEDC). K. A. Mohan was supported by AFOSR/MURI grant FA9550-12-1-0458.

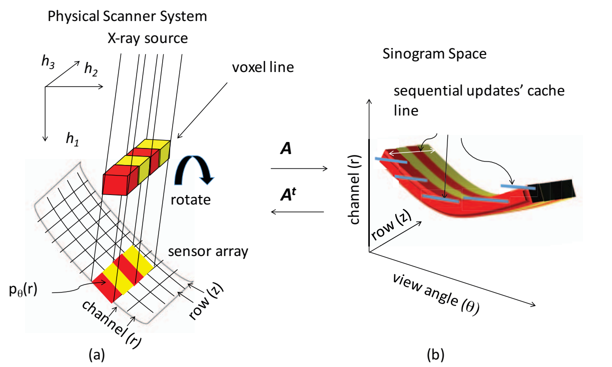

Fig. 1: (a) Illustrates the forward projections of the red voxels and the yellow voxels on a synchrotron based X-ray CT scanner. (b) Shows how the measurement data collected from the CT scanner is organized into the sinogram.

in Sec. 4, we implement the VL-Buffer in MBIR to reconstruct a real data set collected by synchrotron based X-ray CT scanner.

2. BACKGROUND

2.1. Synchrotron Based X-ray CT System

Fig. 1(a) illustrates how a typical parallel beam synchrotron based X-ray CT system operates. We let h_1, h_2 and h_3 denote the coordinate vectors of a right-handed coordinate system, where vectors h_1 and h_2 are perpendicular to the axis of the rotation. Vector h_3 points along the axis of rotation. The object to be imaged is mounted on a rotating stage. As the object rotates about h_3, the CT system takes a set of measurements at each view angle, θ, from all the elements of the detector array. These measurements from each array element, r (or channel), are used to estimate $p_\theta(r)$, the integral density of the object along the path from the X-ray source to the detector.

Typically, the data collected from the scanner is organized into a sinogram indexed by the view θ, channel r and row z (or slices), as illustrated in Fig. 1(b). As we project the source X-rays through individual voxels, the intersecting detector channels, r, at different views, θ, trace out sine wave patterns. For example, the red voxel and the yellow voxel in Fig. 1 have sine wave patterns in the sinogram, shown as the thin red line and the thin yellow line.

To update a voxel value by the ICD algorithm, it is necessary to access this voxel's corresponding measurements in the sinogram following these sine wave patterns. Modern processors access main memory by first transferring blocks of local memory onto cache lines. These cache lines, shown as short blue lines in Fig. 1(b), will only partially overlap the red line or yellow line of memory entries of the sinogram. This means that most of the space in the cache is used to hold data not needed for the current voxel update.

One way to avoid the cache line issue is to update voxels along h_3 axis. We call this set of voxels a voxel line and these voxels share the same geometry calculations [12]. By allowing voxels on a voxel line to update in a sequential manner, a cache line, shown as a short white line in Fig. 1(b), can hold more data because the access pattern along the row is linearized.

Nevertheless, when a voxel line is updated in parallel, only non-neighboring voxels can be updated simultaneously. Although different voxels' traces, belonging to the same voxel line, do not have any intersection in the sinogram space, there are still dependencies in calculating the prior function if voxels are neighbors (see Sec. 2.2). For example, in Fig. 1, red and yellow voxels on the voxel line have the same sinusoidal path in the sinogram space and their voxel traces have no intersections. Neighboring red and yellow voxels, however, can not be updated in parallel. As the consequence, voxel updates in parallel need separated cache line for separated voxels and each cache line fits in less useful data.

Another issue is that the sinusoidal path access along view direction is still sinusoidal even though a cache line reads in linearized data. This makes predicting needed measurements in the near future impossible for the hardware prefetcher.

2.2. Mathematical Model and Objective Function

To better understand the key novelties of this paper, the underlying mathematical and algorithmic concepts of MBIR must first be briefly reviewed. MBIR is based on the numerical solution to an optimization problem described by

$$\hat{u} = \arg\min_{u \geq 0} \left\{ \frac{1}{2}(v - Au)^T D(v - Au) + S(u) \right\} \quad (1)$$

where we consider the image u as a vector of size N whose elements are called voxels. The data v is a vector of size M equal to the total number of measurements for all voxels. A is the $M \times N$ forward system matrix of the scanner geometry, D is a diagonal weighting matrix of size $M \times M$ containing the inverse variance of the scanner noise, and $S(u)$ is the regularizing prior function which depends upon voxels only. The i^{th} diagonal entry of the matrix D, denoted by d_i, is proportional to the photon rate, while inversely proportional to an estimate of the variance in the measurement v_i.

To solve the above optimization problem, the ICD algorithm updates each voxel in sequence to minimize the overall cost function, while keeping the remaining voxels fixed. Formally, the update of the selected voxel u_j is given by

$$\hat{u}_j = \arg\min_{u_j \geq 0} \left\{ \frac{1}{2}(v - Au)^T D(v - Au) + S(u) \right\} \quad (2)$$

To simplify its computation and potentially speed up the ICD algorithm, we use a variable $e = v - Au$ to replace the term $v - Au$. In addition, we also need the first derivative and the second derivative of the cost function with respect to u_j, denoted by θ_1 and θ_2 respectively, to compute $\hat{u}_j$. They can be expressed by using the following equations:

$$\theta_1 = -\sum_{i=1}^{M} d_i A_{ij} e_i, \quad \theta_2 = \sum_{i=1}^{M} d_i A_{ij}^2 \tag{3}$$

where e_i is the i^{th} element in the error term. By using Eqn. (3), we can further simplify Eqn. (2) as:

$$u_j \leftarrow \operatorname*{argmin}_{r \geq 0} \left\{ \theta_1 r + \frac{\theta_2 (r - \tilde{u}_j)^2}{2} + f(r, u_{\partial j}) \right\} \tag{4}$$

Where $\tilde{u}_j$ is the j^{th} voxel's value before the update and $f(r, u_{\partial j})$ is a function of the 26 neighbors of the voxel u_j in three dimensional space.

After the j^{th} voxel is updated, we then update the error term required for the ICD algorithm in the following way:

$$e \leftarrow e + A_{*j}(u_j - \tilde{u}_j) \tag{5}$$

3. EXTENSION

A good single core performance is essential for high performance computing. In this section, we discuss how a voxel line and VL-Buffers can meet the competing goals of good cache locality and good parallel performance.

A voxel line is a special group of voxels that is both strongly coupled and loosely coupled, meeting both goals of cache locality and parallelism. A voxel line is strongly coupled because voxels are grouped together along h_3 axis, sharing the same access pattern. A voxel line is loosely coupled because its voxel traces belong to different rows in the sinogram with no or little intersection among them.[1] Previous research [4], has shown that multicores can work in parallel efficiently on non-neighboring voxels of a voxel line.

Contrary to the well-studied parallelism on a voxel line, also known as inter-slice parallelism [17], the voxel line's

[1]The actual number of intersections depends on the voxel size and the projector model.

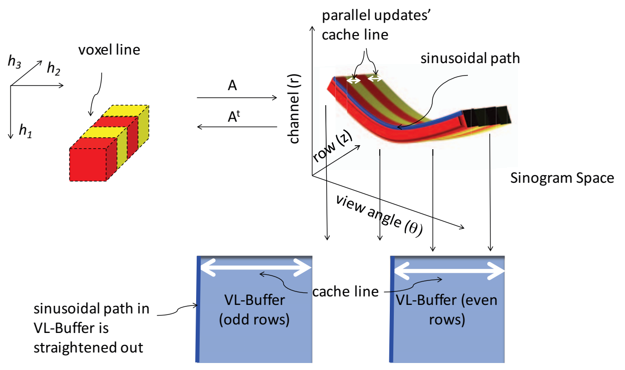

Fig. 2: Measurement data in a voxel line is copied into VL-Buffer so that a cache line (shown in white arrows) can fit more useful data and sinusoidal voxel traces (shown in blue line) are straightened out. In addition, non-coalesced measurements are read in a coalesced way into VL-Buffers.

cache locality advantage has never been studied because of the issues mentioned in Sec. 2.1. To recapture the discussion, there are two inherent issues. First of all, the cache line might not be used efficiently if multiple computing cores update a voxel line in parallel. As shown in Fig. 1, non-neighboring voxels can be updated in parallel. Nevertheless, since voxels are not neighbors, there have to be two independent cache lines reading their measurements and each cache line holds much unneeded data.

The other issue is that the sinusoidal path for voxel traces makes computer hardware prefetching impossible. A computer prefetcher can predict data needed in the near future and prefetch these data into cache ahead of time if the data access pattern follows a linearized pattern. Although the data access pattern in the same view in the sinogram space follows a linearized pattern, the data access pattern across different views follows a sinusoidal path, making prefetching impossible and leading to a high cache miss rate.

To address the above issues, we introduce a new data structure, called the VL-Buffer. In creating VL-Buffers for each computing core, the memory accesses in the voxel line in Fig. 2 are copied to two localized memory spaces, namely VL-Buffers, shown as aquamarine color rectangles in the same figure. One VL-Buffer stores measurements for the voxel traces of odd number rows and the other stores measurements for the voxel traces of even number rows. Therefore, each row of a VL-Buffer consists of linearized and coalesced data in a view angle, θ, of odd number rows or even number rows.

In Fig. 2, white arrows show the cache line in the sinogram space and in the VL-Buffer. We can see that by creating VL-Buffers, non-coalesced measurements of the sinogram becomes coalesced. In addition, red voxels' measurements (or yellow voxels') need only one cache line and more useful measurements will fit into this cache line. Thus the spatial locality significantly improves. In addition, the VL-Buffer lays out in a way that the sinusoidal path can follow a complete straight line pattern that is ideal for hardware prefetching. In Fig. 2, the blue line in the sinogram space shows the sinusoidal path in the sinogram space. This sinusoidal path is straightened out in the VL-Buffer and thus increases computer hardware prefetching.

A major difficulty of VL-Buffers comes from the update of the error term in Eqn. 5. We can see from this equation that ICD requires an update to the sinogram space after each voxel update. It means that we need to copy an updated VL-Buffer to the full sinogram space after each voxel update. This overhead in copying can be significant when the number of cores is large. Since the measurements among different voxels on the same voxel line have no intersection at all, Eqn. 5 can be postponed until the entire voxel line is updated. After that, all measurements in the VL-Buffers are purged out and the VL-Buffers are ready for other voxel lines. By doing this, a full sinogram is updated exactly once for a voxel line. At the same time, all of the measurement data needed for a voxel line

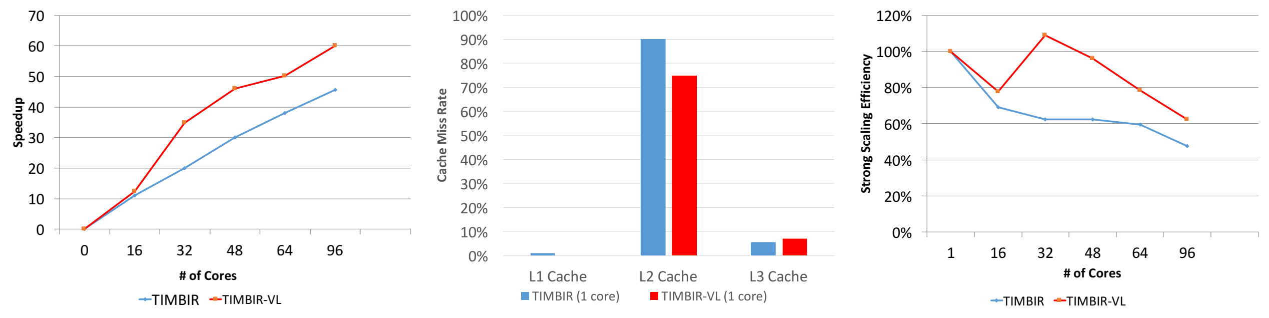

Fig. 3: (a) The blue curve shows the baseline TIMBIR speedup at different numbers of cores. The red curve shows TIMBIR-VL speedup at different numbers of cores. Notice that we achieve an average speed up of 40% by using the VL-Buffers. (b) Illustrates the data cache miss rate for baseline TIMBIR and TIMBIR-VL. (c) TIMBIR and TIMBIR-VL's strong scaling parallel efficiency at different numbers of cores.

are accessed from the localized VL-Buffers. Intuitively, this mechanism allows better cache locality and lower memory copy overhead because we collect all local changes in VL-Buffers and apply a global change to the full sinogram.

4. EXPERIMENTS

In this section, we will compare MBIR reconstructions with and without the VL-Buffer data structure. To compare the reconstruction speed, we define the baseline TIMBIR method as the single core conventional time-interlaced model-based iterative reconstruction (TIMBIR) [4], and TIMBIR-VL as the equivalent but with the addition of the VL-Buffers.

To demonstrate the performance gains achieved by VL-Buffer in a real physical system, we reconstruct an Al-Cu alloy in 4D with 16 sub-frames in the interlaced view sampling, the same data set used in [18]. The detector width is 1600 pixels in the cross-axial direction and 2080 pixels along the axial direction with a pixel resolution of $0.65\ \mu m \times 0.65\ \mu m$. In addition, each voxel line has 24 voxels in the axial direction and each slice has 2080×2080 voxels in the cross-axial direction with a voxel resolution of $0.65 \times 0.65 \times 0.65\ \mu m^3$. Therefore, the reconstructed image volume has size $24 \times 2080 \times 2080$ voxels. Each slice in this data set has (1) 2000 views interlaced between 0 and 180 degrees, and (2) 2080 channels uniformly sampled over the region of interest. The exposure time of the detector is set to 4 ms. The regularization parameters are chosen to provide the best visual reconstruction quality.

In prior extensive experimentation, we have found that a root-mean-squared error (RMSE) of less than 10 Hounsfield Unit (HU), with respect to a fully converged volume, consistently results in a high quality reconstruction with little or no visible convergence artifacts. Therefore, all reconstructions are converged to reach less than 10 HU of RMSE. All computing performance data in this section was collected on multiple standard 2.6 GHz clock rate Intel Processors Xeon-E5 2660 v2 with 8 cores in each processor. Each core has a L1 data cache of size 32 KB and a shared L2 data cache of 256 KB. Each core also has a shared L3 cache of 20 MB.

Fig. 3(a) shows the speedup of TIMBIR and TIMBIR-VL over the baseline TIMBIR at different number of cores. TIMBIR-VL has a performance increase of 12.6% over TIMBIR at 16 cores. This is a direct result of VL-Buffer design to reduce cache misses and prefetching misses. Overall, TIMBIR-VL has a better performance when the number of cores is large. At 96 cores, TIMBIR reaches a speedup of $45X$ while TIMBIR-VL reaches a speedup of $60X$, which is a performance increase of 31.5%. As explained before in Sec. 3, TIMBIR-VL's efficiency will be more prominent with large number of cores because VL-Buffers allows more non-coalesced measurements to be read coalescedly.

Fig. 3(b) shows the cache miss rate of TIMBIR (1 core) and TIMBIR-VL (1 core) at different levels of data cache. The L2 cache miss rate decreases from 90% to 75%. However the L3 cache miss rate mildly increases from 5.6% to 7% because of the memory copy operations in using VL-Buffer. In addition, the prefetching also contributes to the decrease of L2 cache misses. The L2 prefetching hit rate increases from 0% to 6% when using VL-Buffer.

The parallel performance is also a point of interest in discussion. Fig. 3(c) illustrates the strong scaling parallel efficiency in using TIMBIR and TIMBIR-VL. In general, VL-Buffer does not worsen the parallel efficiency. At 96 cores, TIMBIR has a parallel efficiency of 48% while TIMBIR-VL has a parallel efficiency of 63%. It is also interesting to note that TIMBIR-VL has a super-linear speedup at 32 cores because of the reduced cache misses. However when the number of cores further increases, the synchronization overhead becomes more prominent.

5. CONCLUSIONS

While MBIR provides high quality reconstructions, it is considered impractical in some applications because of its long running time. Voxel line updates have been demonstrated to allow efficient parallel operations and significantly reduce running time. In spite of that, each core's performance remains low. In this work, we have introduced VL-Buffer to use cache much more efficiently. Our experimental results have shown a speedup of 40% on average by using VL-Buffer.

6. REFERENCES

[1] R. Mizutani and Y. Suzuki, “X-Ray Microtomography in Biology,” *Micron*, vol. 43, no. 2–3, pp. 104 – 115, 2012.

[2] H. A. Bale, A. Haboub, A. A. MacDowell, J. R. Nasiatka, D. Y. Parkinson, B. N. Cox, D. B. Marshall, and R. O. Ritchie, “Real-Time Quantitative Imaging of Failure Events in Materials under Load at Temperatures above 1,600° C,” *Nature Materials*, vol. 12, no. 1, 2013.

[3] J. Hsieh, B. Nett, Z. Yu, K. Sauer, J.-B. Thibault, and C. Bouman, “Recent Advances in CT Image Reconstruction,” *Current Radiology Reports*, 2012.

[4] K. A. Mohan, S. V. Venkatakrishnan, J. W. Gibbs, E. B. Gulsoy, X. Xiao, M. D. Graef, P. W. Voorhees, and C. A. Bouman, “TIMBER: A Method for Time-Space Reconstruction from Interlaced Views,” *IEEE Transactions on Computational Imaging*, vol. 1, no. 2, pp. 96–111, June 2015.

[5] S. Degirmenci, D. G. Politte, C. Bosch, N. Tricha, and J. A. O’Sullivan, “Acceleration of Iterative Image Reconstruction for X-Ray Imaging for Security Applications,” in *Proceedings of SPIE-IS&T Electronic Imaging*, vol. 9401, 2015.

[6] N. Clinthorne, T. S. Pan, P. C. Chiao, W. L. Rogers, and J. A. Stamos, “Preconditioning Methods for Improved Convergence Rates in Iterative Reconstructions,” *IEEE Transactions on Medical Imaging*, vol. 12(1), 1993.

[7] J. Fessler and S. D. Booth, “Conjugate-Gradient Preconditioning Methods for Shift-variant PET Image Reconstruction,” *IEEE Transactions on Image Processing*, vol. 8(5), 1999.

[8] J. Shewchuk, *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*. Pittsburgh, PA: Carnegie Mellon University, 1994.

[9] C. Kamphuis and F. J. Beekman, “Accelerated Iterative Transmission CT Reconstruction Using an Ordered Subsets Convex Algorithm,” *IEEE Transactions on Medical Imaging*, vol. 17(6), 1998.

[10] K. Sauer and C. Bouman, “A Local Update Strategy for Iterative Reconstruction from Projections,” *IEEE Transactions on Signal Processing*, vol. 41(2), 1993.

[11] B. D. Man, S. Basu, J.-B. Thibault, J. Hsieh, J. A. Fessler, C. A. Bouman, and K. Sauer, “A Study of Different Minimization Approaches for Iterative Reconstruction in X-ray CT,” in *IEEE Nuclear Science Symposium*, vol. 5, 2005, pp. 2708–2710.

[12] 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(1), 2011.

[13] J. A. Fessler, E. Ficaro, N. Clinthorne, and K. Lange, “Grouped-Coordinate Ascent Algorithms for Penalized-Likelihood Transmission Image Reconstruction,” *IEEE Transactions on Medical Imaging*, vol. 16(2), 1997.

[14] J. Zheng, S. S. Saquib, K. Sauer, and C. A. Bouman, “Parallelizable Bayesian Tomography Algorithms with Rapid, Guaranteed Convergence,” *IEEE Transactions on Image Processing*, vol. 9(10), 2000.

[15] J. A. Fessler and D. Kim, “Axial Block Coordinate Descent (ABCD) Algorithm for X-ray CT Image Reconstruction,” in *11th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*, 2011.

[16] X. Wang, C. A. Bouman, and S. P. Midkiff, “High Performance Model Based Image Reconstruction,” in *2015 ACM/IEEE Conference on Supercomputing*, November 2015. [Online]. Available: http://sc15.supercomputing.org/sites/all/themes/SC15images/src_poster/poster_files/spost107s2-file1.pdf

[17] X. Wang, A. Sabne, S. J. Kisner, A. Raghunathan, C. A. Bouman, and S. P. Midkiff, “High Performance Model Based Image Reconstruction,” in *21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, March 2016.

[18] J. W. Gibbs, K. A. Mohan, E. B. Gulsoy, A. J. Shahani, X. Xiao, C. Bouman, M. D. Graef, and P. Voorhees, “The Three-Dimensional Morphology of Growing Dendrites,” *Scientific Reports*, 2015.