Title:
Chunked Voxel Models for Realtime Haptic Collision and Material Removal in CT-Derived Anatomy

Authors:
Vlad Popa, vpopa2@uwo.ca, Western University, Canada
O. Remus Tutunea-Fatan, rtutunea@uwo.ca, Western University, Canada
Reza Faieghi, reza.faieghi@torontomu.ca, Toronto Metropolitan University, Canada

Keywords:
Voxel-Based Simulation, Haptics, CT Reconstruction, Chunked Voxel Grid, Collision Detection, Material Removal, GPU computing, OpenCL

DOI: 10.14733/cadconfP.2026.130-134

Introduction:

Haptics-enabled surgical simulators require stable, low-latency collision response between the surgical tool and anatomical geometry. Many of these systems operate with servo-loop update rates near 1 kHz, which imposes strict bounds on collision detection, force update, and any material-removal computations performed in the loop. In voxel-based simulators, collision detection and material removal can be implemented robustly by testing occupancy or density over a discretized volume; nonetheless, real-time operation is often constrained by the memory footprint of volumetric representations and by the cost of repeatedly scanning large voxel buffers during localized contact events [9, 10]. These constraints become more pronounced as CT volumes increase in extent, anisotropy, or required resolution.

Dense voxel storage as a single 3D matrix (or 3D texture) is particularly inefficient for CT-derived anatomy, where the occupied volume may be a small fraction of the axis-aligned bounding box. Conversely, hierarchical sparse structures (octrees) can reduce memory but may introduce traversal overhead and more complex update logic when frequent localized material removal is required. Therefore, sparse-at-chunk-level representations are of interest for interactive volumetric editing tasks—such as virtual surgical reaming and related localized cutting operations—where contact is spatially limited but repeated updates are required [1, 3].

Main Idea:

A chunked voxel representation is presented for CT-derived anatomical models in which the global voxel field is partitioned into fixed-size cubic chunks and memory is allocated only for active chunks. The contributions are threefold: 1) a sparse-at-chunk-level voxel data structure that preserves O(1) addressing via direct chunk indexing; 2) a GPU collision/material-removal workflow in which voxel scans are restricted to a candidate set of intersecting chunks selected by a conservative tool–bone intersection estimate; and 3) a quantitative characterization of memory and collision-event timing across CT and porous volumes, including chunk-size tradeoffs that yield practical guidelines for real-time interaction. Active memory reductions of up to ~50% were observed for non-uniform CT anatomy (scapula), while collision-event times remained in the sub-millisecond to millisecond range when chunk dimensions were matched to localized contact.

To accomplish the aforementioned objective, CT image data are converted to a rectilinear voxel grid and then transferred to a cubic voxel grid suitable for chunking. A chunk grid is constructed by padding the cubic voxel grid to the nearest multiple of the chunk dimension. Chunks are marked active if any voxel in the chunk contains data of interest; inactive chunks are not allocated in the active voxel buffer (Fig. 1). This reduces active memory when the anatomy is non-uniform within its bounding volume (Tabs

1 and 2). Collision checks are restricted to the candidate set of tool–bone intersecting chunks. Within each chunk, local voxel indices are computed analytically from global voxel coordinates (integer division and modulo), avoiding tree traversal. Material removal is implemented by clearing intersected voxels (and associated metadata) within affected chunks, maintaining real-time updates suitable for a haptics loop [2].

Fig. 1: Active (red) vs. inactive (gray) chunks in a scapula chunk grid.

The approach was evaluated on five volumetric datasets (scapula, glenoid, two trabecular cores, cellular foam) spanning CT-derived anatomy and porous structures. Voxel sizes ranged from isotropic micro-CT (32 µm) to anisotropic clinical CT (0.47 × 0.47 × 1.0 mm). Metrics included allocated memory (dense vs. chunked), collided chunks per event, processing time per chunk, and total collision-event time. Tests were run on a Windows workstation (Intel i5-class CPU, NVIDIA GTX 1070-class GPU). As indicated in Tabs. 1–2, substantial memory savings were achieved for non-uniform anatomy (scapula), with active memory reduced by up to ~50% relative to the dense grid.

Model	*Voxel size*	*Voxel grid dimensions*	*Voxel Count*	*Active Memory (2B/Voxel)*	*Text file size on disk*
Cellular foam 1	32µm isotropic	314 × 320 × 604	60,689,920	121.4 MB	764.1 MB
Trabecular 2	32µm Isotropic	314 × 316 × 360	35,720,640	71.4 MB	1.4 GB
Trabecular 1	32µm isotropic	156 × 156 × 313	7,617,168	15.2 MB	101.4 MB
Scapula	(0.47, 0.47, 1) mm	256 × 234 × 201	12,040,704	24.1 MB	34.3 MB
Glenoid	(0.47, 0.47, 1) mm	49 × 81× 41	162,729	0.3 MB	3.7 MB

Tab. 1: General voxel grid characteristics of the CT-derived models used for evaluation.

Model	*Voxel grid dims after chunk gen*	*Chunk grid dimensions*	*Chunk count*	*Active chunk count*	*Raw memory (2B/Voxel)*	*Active memory (2B/Voxel)*
Cellular foam	320 × 320 × 608	10 × 10 × 19	1900	1669	124.5 MB	109.4 MB
Trabecular 2	320 × 320 × 384	10 × 10 × 12	1200	1060	78.6 MB	69.5 MB
Trabecular 1	160 × 160 × 320	5 × 5 × 10	250	250	16.4 MB	16.4 MB
Scapula	256 × 256 × 448	8 × 8 × 14	896	202	58.7 MB	13.2 MB
Glenoid	64 × 96 × 96	2 × 3 × 3	18	18	1.2 MB	1.2 MB

Tab. 2: Chunk grid characteristics of the CT-derived models.

By contrast, for nearly cubic and uniformly occupied volumes – such as the glenoid and one trabecular core – savings were not guaranteed. In these cases, chunk-aligned padding increases the effective bounding volume, while the scarcity of inactive chunks leaves little opportunity for sparsity at the chunk level. Consequently, the chunk representation approaches dense storage in both allocation and access costs [2]. This behavior indicates that the benefit of chunking is strongly tied to the distribution of

occupied voxels within the global bounding box and to the fraction of chunks that can remain inactive. Figure 2 summarizes active-voxel memory across models and data structures.

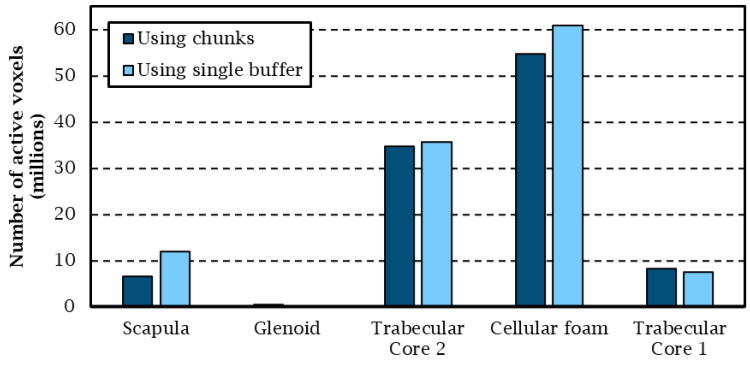

Fig. 2: Active voxels stored in memory across test models and data structures.

Static collision tests were also performed with representative reaming and peg–hole tools to characterize the effect of chunk dimensions on computation time. The two static collision configurations used for benchmarking are shown in Fig. 3. Three chunk sizes were tested (16^3, 32^3, and 64^3 voxels per chunk), each corresponding to different subdivisions of the same interaction region. Tabs. 3 and 4 report the number of collided chunks, processing time per chunk, and total collision-event time. The 32^3 configuration consistently provided robust performance across tools, balancing subdivision (voxels scanned per chunk) with the overhead of processing many chunks. The 64^3 configuration was consistently slower because each collided chunk requires scanning a much larger number of voxels even when only a small portion of the chunk intersects the tool.

a) b)

Fig. 3: Static collision configurations used for chunk-size performance testing with: a) reamer and b) peg–hole cylinder tools.

Chunk dimensions	Chunks collided	Processing time per chunk (ms)	Total processing time (ms)
16^3	84	0.012	1.024
32^3	18	0.055	0.970
64^3	5	0.405	2.023

Tab. 3: Static collision event timings for various chunk dimensions using the reamer tool.

A chunked voxel representation enables sparse-at-chunk-level storage and localized GPU collision/material-removal updates suitable for real-time haptics loops [2]. By allocating voxel buffers only for active chunks, memory usage is driven primarily by the occupied regions of the anatomy rather than by the full axis-aligned bounding volume, while direct chunk indexing preserves constant-time access during repeated updates. Significant memory reductions were demonstrated for non-uniform CT-derived anatomy, and collision-event times remained near the sub-millisecond-to-millisecond range when chunk sizes were selected to match localized contact (Tab. 5). These results indicate that practical real-time performance can be achieved when the contact footprint remains spatially limited and when chunk dimensions balance per-chunk scan cost against per-event overhead.

Chunk dimensions	Chunks collided	Processing time per chunk (ms)	Total processing time (ms)
16^3	80	0.012	0.964
32^3	18	0.065	1.172
64^3	8	0.405	3.234

Tab. 4: Static collision event timings for various chunk dimensions using the peg–hole cylinder tool.

Tool voxel size (mm)	Chunks collided	Total event time (ms)	Time per chunk (ms)	Tool voxel count
1.03	18	0.982	0.0545	33,212
0.753	18	0.975	0.0542	75,000
0.53	18	0.967	0.0538	232,286

Tab. 5: Static collision event timings at chunk dimensions of 32^3 using the reamer tool while varying tool voxel-sizes.

Discussion:

Across tested scenarios, collision-event computation ranged from sub-millisecond to a few milliseconds depending on chunk size and tool geometry. For the reamer configuration, total event times of approximately 1.0 ms were observed for both 16^3 and 32^3 chunks, whereas 64^3 chunks increased event time beyond 2 ms. For the peg–hole configuration, 16^3 chunks provided the lowest total event time (< 1 ms), while 64^3 again resulted in multi-millisecond events. These results support a practical guideline: 32^3 chunks are recommended for general-purpose tools with larger contact patches, while 16^3 may be advantageous for tools with small surface area and elongated contact regions.

Beyond collision timing, predictable access patterns are provided that are favorable for GPU execution. Local voxel indices are computed analytically from global coordinates (integer division/modulo), avoiding hierarchical traversal and reducing branch divergence relative to octree-based representations. Along these lines, constant-time addressing supports repeated material-removal updates without re-voxelization or rebuilding global acceleration structures. However, a different trade-off is targeted relative to adaptive octree meshes or linked-voxel lattices: updates are localized to a small set of affected chunks while maintaining uniform resolution within each chunk, matching localized contact regimes where only a small fraction of the anatomy is engaged at any time [3, 5, 9, 10].

Two practical guidelines follow from the presented benchmarking: i) 32^3 chunks provide robust performance across tool types by balancing voxels scanned per chunk with per-event overhead (Tabs. 3–4); and ii) smaller chunks can be advantageous for slender contacts, whereas larger chunks incur unnecessary scans of empty voxels even under sparse contact. Visualization of the evolving voxel field can be implemented without frequent global remeshing by rendering voxels directly on the GPU; ray–box intersection methods support efficient rendering of large dynamic voxel scenes and can be integrated with chunk-level updates when required [4]. A cost decomposition is useful for interpreting the reported timings. For each collision event, total time can be viewed as the sum of: i) chunk selection and dispatch, ii) per-chunk voxel scans in GPU kernels, and iii) update/write-back of modified voxel densities. Once candidates are restricted to the localized contact region, per-chunk scans dominate and this explains sensitivity to chunk dimensions (Tabs. 3–4 and [2]).

From a CAD perspective, the chunk representation can be interpreted as a uniform-resolution volumetric analogue of region-based refinement: resolution is fixed within chunks while sparsity is achieved via activation only where geometry exists, enabling localized operations (drilling, reaming, voxel-based Boolean subtraction) without global remeshing [1]. The limitations are primarily associated with padding and activation granularity. When occupied anatomy is nearly uniform in its bounding box, or when chunks are too large relative to the contact patch, padding and full-chunk scans reduce efficiency. By contrast, overly small chunks increase per-event overhead and bookkeeping. These regimes are identifiable from active-chunk distributions and expected contact footprint [2, 9, 10].

Conclusions:

A chunked voxel representation enables sparse-at-chunk-level storage and localized GPU collision/material-removal updates for CT-derived anatomical models. The approach reduces active memory in non-uniform anatomy, limits collision work to the subset of affected chunks, and achieves sub-millisecond collision-event times in typical localized-contact regimes. The reported chunk-size tradeoffs provide practical guidance for selecting chunk dimensions to meet real-time interaction requirements in haptics-enabled surgical simulation. Future work will extend the present benchmarking and analysis to interaction stability under sustained haptic feedback, toolpath-driven removal trajectories, and broader volumetric editing benchmarks, while retaining the same chunked representation and evaluation framework.

Acknowledgements:

The authors would like to acknowledge the financial support provided in part by Natural Sciences and Engineering Research Council (NSERC) of Canada and Canadian Institutes of Health Research (CIHR) that was received under the framework of the Collaborative Health Research Projects (CHRP) program.

Vlad Popa, https://orcid.org/0009-0005-5482-3306
O. Remus Tutunea-Fatan, https://orcid.org/0000-0002-1016-5103
Reza Faieghi, https://orcid.org/0000-0003-0075-2969

References:

[1] Aleksandrov, M.; et al.: Voxelisation Algorithms and Data Structures: A Review, 2021. https://pmc.ncbi.nlm.nih.gov/articles/PMC8707769
[2] Faieghi, M.; Tutunea-Fatan, O.R.; Eagleson, R.: Fast and scalable GPU-based collision detection for voxelized models, Computer-Aided Design and Applications, 15(6), 2018, 852-862. https://doi.org/10.1080/16864360.2018.1477729
[3] Jia, S.; et al.: A real-time deformable cutting method combining a uniform grid of linked voxels and an octree of linked voxels, Multimedia Tools and Applications, 2025. https://doi.org/10.1007/s11042-025-21037-0
[4] Majercik, A.; Crassin, C.; Shirley, P.; McGuire, M.: A Ray-Box Intersection Algorithm and Efficient Dynamic Voxel Rendering, Journal of Computer Graphics Techniques, 7(3), 2018, 66-81. https://jcgt.org/published/0007/03/04
[5] Qi, D.; et al.: Divided Voxels: An efficient algorithm for interactive cutting based on an adaptive octree simulation mesh, 2020. https://pmc.ncbi.nlm.nih.gov/articles/PMC8133368
[6] Renz, M.; Preusche, C.; Hirzinger, G.: Stable haptic interaction with virtual environments using the voxmap-pointshell algorithm, Proceedings of Eurohaptics, 2001.
[7] Sagardia, M.; Hulin, T.; Preusche, C.; Hirzinger, G.: Improvements of the voxmap-pointshell algorithm, Proceedings of the Internationales Wissenschaftliches Kolloquium, 2008.
[8] Scarpino, M.: OpenCL in Action: How to Accelerate Graphics and Computations, Manning, 2011.
[9] Seiler, M.; et al.: Robust interactive cutting based on an octree simulation mesh, The Visual Computer, 27(6-8), 2011, 519-529. https://doi.org/10.1007/s00371-011-0562-2
[10] Yau, H.; Tsou, L.; Tsai, M.: Octree-based virtual dental training system, Computer-Aided Design and Applications, 3(1-4), 2006, 415-424. https://doi.org/10.1080/16864360.2006.10738471