Fast Generation of Cartesian Meshes from Micro-Computed Tomography Data

Mohammadreza Faieghi[1], Nikolas K. Knowles[2], O. Remus Tutunea-Fatan[3] and Louis M. Ferreira[3]

[1]Biomedical Engineering, Western University, Canada, mfaieghi@uwo.ca
[2]Biomedical Engineering, Western University, Canada, nknowle@uwo.ca
[3]Mechanics and Materials Engineering, Western University, Canada, rtutunea@eng.uwo.ca
[4]Mechanics and Materials Engineering, Western University, Canada, louis.ferreira@sjhc.london.on.ca

Corresponding author: O. Remus Tutunea-Fatan, rtutunea@eng.uwo.ca

ABSTRACT

Micro-finite element models (μFEMs) are one of the critical components of the microscale analyses that are typically performed on trabecular bone. These models are often derived from on micro computed tomography (μCT) data and tend to encompass an extremely large number of elements that in turn require significant processing time and power. To address the increased computational demands, the main goal of the current study was to devise an algorithm capable to manage the large μCT data in order to construct Cartesian μFEMs. For this purpose, the developed technique relies on the projection of μCT voxels to a structured grid and were designed to involve fast integer operations and hashing techniques for fast mesh constructions. The numerical tests performed on common computer hardware revealed that only 55.16 seconds are required to discretize more than 36.2M voxels. Furthermore, the linear time complexity of the developed algorithm ensures that its efficiency will be preserved even in case of larger datasets that tend to be prevalent in micro-structural biomechanical analysis.

Keywords: Micro-finite element model (μFEM), Micro computed tomography (μCT), Hexahedral mesh, Cartesian Mesh, Voxelization
DOI: https://doi.org/10.14733/cadaps.2019.161-171

1 INTRODUCTION

Finite element analysis (FEA) plays a critical role in the accurate assessment of local stress-strain distribution to develop in geometrically complex structures [2]. Although FEA had been initially proposed to solve problems in traditional structural fields - such as automobile and aircraft industries [11] - it has continuously evolved and eventually extended into many other engineering disciplines, including biomechanics where FEA serves as a standard tool used in the structural analysis of osseous and other tissues [12], [5].

One of the common FEA problems is related to the generation of the finite element models (FEMs) to capture as accurately as possible the geometrical, loading and material properties of a given biomechanical structure. Quite often, the generation of FEMs for osseous tissue begins with the acquisition of computed tomography (CT) data that is capable to simultaneously capture the geometry and at least some of the material characteristics of the scanned bone [6]. Without going into extensive details, it will be briefly reminded here that the non-destructive CT technology generates cross-sectional images of the analyzed bone that can be subsequently used to reconstruct 3D volumetric replicas. The relatively recent advent of micro computed tomography (μCT) has enabled digital bone

reconstruction with micron resolutions to be then integrated into downstream micro-finite element models (μFEMs) [13], a critical tool in the investigation of the internal porous structure of the osseous tissue, typical known as the trabecular/subchondral bone [16].

Since μCT data could be made up of tens of millions of voxels that correspond in turn to tens of millions of elements, one of the common challenges of μFEM is constituted by the ability to discretize extensive models in a minimal amount of time. Furthermore, since hexahedral FEA elements are characterized by a higher numerical performance compared to their tetrahedral counterparts, they tend to be preferred by most analysts [1], [26]. However, commercial FEA software makes the construction of hexahedral meshes either time-consuming or restricted to relatively simple geometries. To address this, a number of studies [14], [17], [30] have proposed various solutions. However, their main emphasis has been modeling accuracy whereas running time was not reported. Because of that, their efficiency remains uncertain.

A particular type of hexahedral mesh - known as Cartesian mesh - can be generated through a direct conversion of CT voxels into eight-node hexahedral elements [19],[24]. This type of mesh provides the advantage of a reduced time in both mesh generation and solving phases, but this commonly happens at the cost of a coarse representation of the bone surface [1]. However, at the micron resolution levels that are characteristic to μFEMs, this type of discretizations represent a viable option, particularly since the small size of the elements tends to diminish the – otherwise prominent – surface appearance artifacts. To date, several studies have proved that Cartesian meshes could accurately predict the mechanical properties of the trabecular bone as measured through physical experiments [8], [9], [18], [20], [22], [23], [27], [28].

The generation of a Cartesian mesh from μCT data poses a number of challenges that are primarily derived from the large size of the μCT data. As it can be inferred, processing of this amount of data is a time-consuming operation, even if performed with powerful hardware. Moreover, the available tools rely on unoptimized codes that are subjected to random crashes caused by poor/obsolete memory management routines. As such, robust numerical techniques are mandatory in this context since the number of decimal points to be handled exacerbates all known flaws associated with the generic and widespread floating-point operations.

Along these lines, the present study proposes an efficient algorithm that was designed to address the known challenge of rapid construction of Cartesian meshes from μCT data. The core idea of the algorithm is that since μCT data consist of uniform voxels, it could be stored in a uniform 3D grid. If this grid is determined/known, then fast and robust integer operations within the grid coordinates can be used to manage the necessary computations. As discussed above, since the small dimensions of the μCT voxels are accurate enough for μFEMs, the algorithm was restricted to output only cubes/voxels as simple hexahedral elements, and this should result in significant reductions of the computational time. Another prominent feature of the algorithm is related to the use of the hashing techniques for indexing of the nodes and materials in the μFEM [10], [21]. The implementation of the hashing techniques allows the search operations to run in constant time and this yields a desirable linear time complexity of the algorithm, thus making it extremely efficient/suitable for large data sets. More details on the proposed numerical implementation are provided in the upcoming sections.

2 ALGORITHM OVERVIEW

In order to test the viability of the proposed approach, the developed μCT-to-mesh generation algorithm has been intentionally tailored to the needs/input of a specific commercial package (Abaqus) that tends to be routinely used in the FEA of the biomechanical structures. However, it should be relatively easy to understand that this type of particularization does not significantly restricts the generality of the method to be described further. Moreover, at the highest conceptual level, the developed algorithm outputs geometrical, topological and material characteristics associated with finite element mesh.

According to the specific mesh importing needs of Abaqus, the algorithm was designed to output an Abaqus-specific input file which includes four distinct blocks of data: i) 3D coordinates of mesh nodes/vertices, ii) indices of the nodes; they are required for elements formation, (3) element-sets; each set is represented by a group of elements characterized by identical material properties and (4) indices of the element-sets; they are used to assign material properties to each of the element-sets.

The major phases involved in the developed μFEM generation method are outlined in Fig. 1. First, μCT images were exported as 16-bit DICOM files to be then loaded into the commercial Mimics software. The high-frequency noise of the raw images was removed by means of a discrete Gaussian filter. As recommended in [4], a specimen-specific gray-value threshold was used for trabecular bone in order to best preserve its architecture. The image segmentation was performed via region growing with embedded “six-connectivity”. This approach ensures the face connectivity of hexahedral elements and avoids the generation of nonmanifold geometries.

Microscale finite element studies for bones are often performed within the elastic regime with Poisson's ratio equal to 0.3 and Young's modulus derived from gray-values intensity, or simply gray-values [7]. Therefore, the gray-value of each voxel must be passed to the mesh generation algorithm, along with its spatial data. Mimics allows the export of this information into a text file in which every row contains the centroid coordinates $\left(x_i, y_i, z_i\right) \in \mathbb{R}^3$ and the gray-value $r_i \in \mathbb{R}$ of the i-th voxel, where $1 \le i \le n$ and n is the of number post-processed μCT voxels.

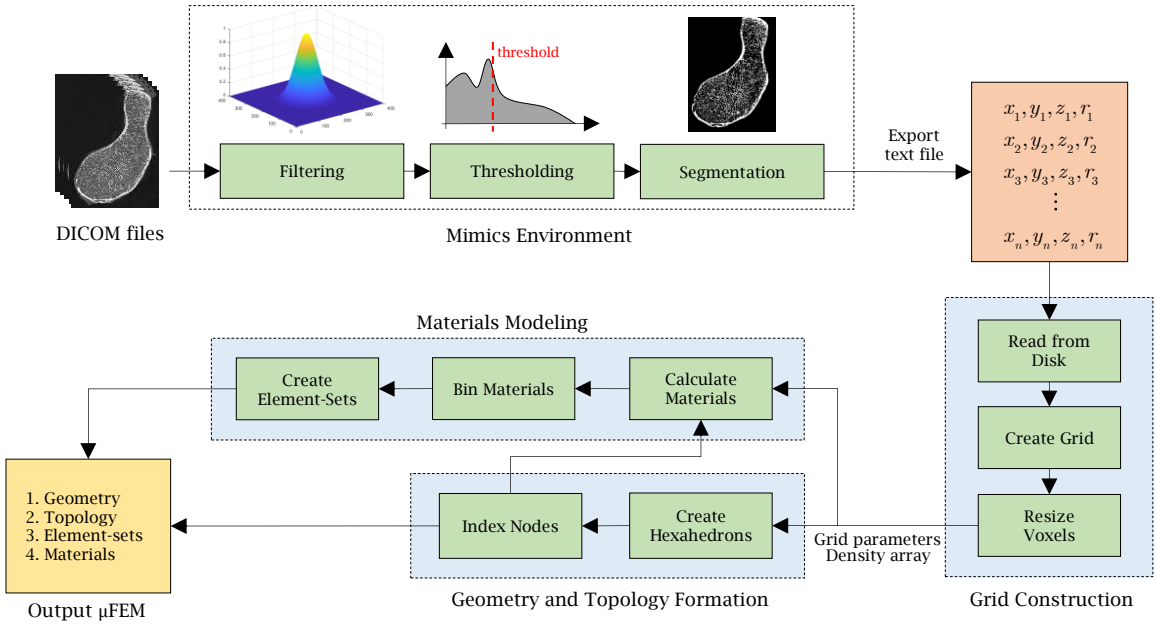

Figure 1: Flowchart of the developed algorithm.

The choice of Mimics for data preparation was merely determined by its availability. Alternatively, it is also possible to read the voxel data directly from DICOM files and then feed it into the algorithm. However, since μCT images often require manual segmentation in order to extract the region of interest, the need for a powerful image processing tool remains valid [29].

The developed μFEM generation algorithm is comprised of three main components. More specifically, after the input text file is loaded, the algorithm constructs a structured 3D grid that embeds all μCT voxels. This allows for the implicit identification of each voxel by means of efficient integer operations. This grid is then passed to the *Geometry and Topology Formation* block, in which an explicit representation of the mesh is generated. Then, *Material Processing* block constructs distinct element-sets and assigns each element-set with a material property that is inferred from the gray-values of each voxel. This completes the mesh generation process such that the resulting information can be then transferred to Abaqus for the remainder of the FEA steps.

3 GRID GENERATION AND CONSTRUCTION

A structured uniform grid can be uniquely identified by means of its voxel size, minimum corner and the number of voxels to be stored in each direction (sometimes termed “grid dimension”). Evidently, the size of the voxel ($\mathbf{s} \in \mathbb{R}^3$) is *a priori* known by the resolution of the μCT. By contrast, the last two parameters from the aforementioned list of three can be easily inferred from the centroid of the voxels, an operation to take place directly during the file stream. The minimum $\mathbf{c}_{\min}$ and maximum $\mathbf{c}_{\max}$ corners of the grid can be easily derived from the all voxels centroids. Then,

the grid dimension is computed by the element-wise calculation of $\mathbf{d} = \frac{\mathbf{c}_{\max} - \mathbf{c}_{\min}}{\mathbf{s}} \in \mathbb{Z}^3$. As a result, the total number of voxels within a grid $\mathcal{G}(\mathbf{s}, \mathbf{c}_{\min}, \mathbf{d})$ can be expressed as $n_{\max} = d_x d_y d_z$.

Once the grid is identified, the algorithm proceeds with dynamic allocation of a linear array of integers whose length is $n_{\max}$. This array, hereafter referred to as the density array, provides a memory-efficient way to describe the spatial occupancy of the grid. In particular, each element of the array stores the gray-value of an individual voxel whereas the indices of these elements encode the coordinates. As a result, each voxel can be easily identified by means of a few elementary integer operations. Specifically, a voxel $\mathcal{V}$ with integer grid coordinates $(u, v, w) \in \mathbb{Z}^3$, ranging from $\mathbf{o}_3$ to $(d_x - 1, d_y - 1, d_z - 1)$, maps to the i-th element given by

$$i = u + v\, d_x + w\, d_x\, d_y . \tag{3.1}$$

Conversely, for a given index i, the voxel coordinates can be calculated by

$$w = \left\lfloor \frac{i}{d_x d_y} \right\rfloor, \; v = \left\lfloor \frac{i - w d_x d_y}{d_x} \right\rfloor \text{ and } u = i - w\, d_x\, d_y - v\, d_x . \tag{3.2}$$

Therefore, the center coordinates of a voxel can be readily computed by

$$\mathbf{c} = \mathbf{c}_{\min} + < \mathbf{s}, (u, v, w) > \tag{3.3}$$

Where $\mathbf{c} \in \mathbb{Z}^3$ denotes center coordinates and $<,>$ represents the dot product. As the above equations only involve integers, the use of the density array minimizes the need for floating-point operations that – given the number of decimal places associated with µCT data – are known to be both error-prone as well as slow.

For many FEA applications, one hexahedral element per µCT voxel represents an acceptable resolution. Nevertheless, if finer meshes are needed, the uniform grid was designed to be passed to the voxel up-sampling step where sub-voxel mesh resolutions can be generated. To ensure the conservation of the model volume, the voxel size of the grid has to be a divisor of the µCT voxel size: $\mathbf{s} = k\, \mathbf{s}', k \in \mathbb{Z}$. Then, the minimum corner of the new grid is placed at the minimum corner of the previous grid, but a new density array is constructed with a length of $d'_x d'_y d'_z$, where $\mathbf{d}' = k\mathbf{d}$. The gray-values of the new voxels are obtained through the linear interpolation of the native µCT gray-values via iterations performed over the new density array. Finally, the new density array along with new grid parameters are passed to the next processing phase to be outlined further.

4 GENERATION OF GEOMETRY AND TOPOLOGY

As a general rule, Cartesian mesh is comprised of eight-node hexahedral elements. Node coordinates can be computed by means of the centroid coordinates of the corresponding µCT voxels. For this purpose, the algorithm loops over the density array, calculates the centroid coordinates of occupied voxels via Eq. (3.2) and (3.3), and generates eight nodes in an order shown in Fig. 2. The resulting nodes are stored in an array of $8n$ length where n is the number of hexahedral elements and the elements spanning from $8i$ to $8i + 7$ correspond to the i-th hexahedral element. This array captures unequivocally and concurrently both the geometry and topology of the mesh. The main advantage of this approach resides in that the knowledge of the adjacent nodes is not required since nodes are constructed independently for each of the elements. This is potentially beneficial for out-of-core as well as parallel extensions of the algorithm to become valuable additions in the management of large models. However, this approach will inevitably lead to duplicate nodes that are unacceptable for many FEA solvers. Therefore, the main purpose of the node indexing step is to ensure the uniqueness of the each of the nodes by also ensuring their connectivity.

Evidently, one of the simplest way to remove duplicates would rely on nested loops that would be set to continuously iterate over the array in order to identify the identical/duplicate elements to be subsequently eliminated. However, since the average time complexity for array searching is linear, the worst-case time complexity associated with this method would end up being quadratic and thereby rendering a poor performance in the present large dataset context. To accelerate this particular phase, the developed algorithm incorporates a more efficient searching technique called hash mapping.

$$\mathbf{n}_1 = \mathbf{c} + \langle (-1,-1,+1), 0.5*\mathbf{s} \rangle \quad \mathbf{n}_5 = \mathbf{c} + \langle (-1,+1,+1), 0.5*\mathbf{s} \rangle$$
$$\mathbf{n}_2 = \mathbf{c} + \langle (+1,-1,+1), 0.5*\mathbf{s} \rangle \quad \mathbf{n}_6 = \mathbf{c} + \langle (+1,+1,+1), 0.5*\mathbf{s} \rangle$$
$$\mathbf{n}_3 = \mathbf{c} + \langle (+1,-1,-1), 0.5*\mathbf{s} \rangle \quad \mathbf{n}_7 = \mathbf{c} + \langle (+1,+1,-1), 0.5*\mathbf{s} \rangle$$
$$\mathbf{n}_4 = \mathbf{c} + \langle (-1,-1,-1), 0.5*\mathbf{s} \rangle \quad \mathbf{n}_8 = \mathbf{c} + \langle (-1,+1,-1), 0.5*\mathbf{s} \rangle$$

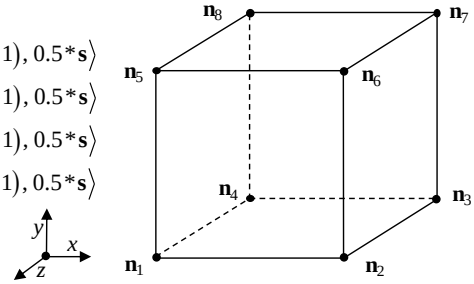

Figure 2: Topology of the hexahedral element of the Cartesian mesh.

In brief, a hash map is a data container that stores every node coordinate while pairing it with a key value. These key values are computed by a hash function assigned to the hash map. While a linear array will inevitably stores its elements in a sequential, the hash map places the elements based on their keys. As a result, searches performed on hash maps will always yield a constant time complexity since the algorithm will only iterate once over each node. At each iteration, all duplicate nodes are identified and removed from the hash table. This leads to an overall linear time complexity that in turn translates into significant computing time decreases when compared to the conventional nested-loop approach (Tab. 1). The resulting nodes and indices will form the explicit representation of the final μFEM.

Number of μCT voxels	*Total number of nodes*	*Number of duplicate nodes*	*Runtime (ms)*	
			Nested loops	*Hash mapping*
500	4,000	3,147	2.696	0.212
1,000	8,000	7,167	10.476	0.435
5,000	40,000	33,735	162.167	1.795
15,000	120,000	101,123	15,92.704	6.175
40,000	320,000	270,113	10,986.821	15.416

Table 1: Comparative assessment of hash mapping efficiency.

5 MATERIAL MODEL

While gray-values can provide precise information about the density of an object, additional processing is required in order convert them into meaningful material properties. While presently there is no generally accepted mapping between gray-values and bone elasticity, most conversion methods advocate for the need of an user-defined function $m = f(r)$ that maps the gray-value r into a particular material property m [25]. For trabecular structures, this function might be defined as a linear mapping $f(r) = ar + b$ where a and b are constants [3]. This function is implemented in the *Calculate Material* block and essentially converts the μCT gray-value into a corresponding Young's modulus. Then, in the optional material binning step, the computed moduli are categorized into bins of user-defined widths. Subsequently, materials belonging to the same bin are substituted by the center of their bin. This process decimates the number of materials derived from μCT in order to reduce the complexity of the resulting μFEM in an attempt to speed up the FEA computations. The final material models are then listed in the output μFEM, however, they are to be linked first to the mesh elements that were produced in the anterior step.

To do that, a second hash map technique was utilized to identify all elements characterized by identical material properties to be then grouped as distinct element sets. Following this, the elasticity of each set was linked to the computed list of materials by means of indices. As a result, material characteristics of all mesh elements could be generated.

6 RESULTS AND DISCUSSION

Several μCT samples were used to test the performance of the proposed numerical technique. C++ language was used as the programming platform and computing time was measured by means of the chrono timer that is available in C++ standard library [15]. The hardware used for the tests included a standard Core-i7 6700K CPU equipped with 16 GB RAM. The models used for the tests include three μCT datasets as well as a clinical CT image of scapula bone. Details on these models are provided in Tab. 2, below.

Model	*Voxel Size*	*Voxel Grid Dimension*	*Number of Occupied Voxels*	*Text File Size on Disk*
Trabecular Specimen	32μm isotropic	$281\times372\times353$	1.7 M	84.8 MB
Cellular foam	32μm isotropic	$422\times629\times652$	12.4 M	746 MB
Glenoid	64μm isotropic	$1021\times548\times742$	36.2 M	1.68 GB
Scapula	(0.47, 0.47, 1) mm	$311\times284\times169$	558 K	33.5 MB

Table 2: Characteristics of the models used for testing.

6.1 Running Time Breakdown – Fixed Voxel Size

Initially, the resolution of the hexahedral FE mesh was set to match that of the acquired μCT scan. The material properties of the larger samples (*e.g.*, cellular foam and cadaveric glenoid) were binned with a bucket size of 10. Tab. 3 shows the breakdown of the running time for different steps of the algorithm. To eliminate confounding errors, I/O times were not considered. Given the comparison results in Tab. 1 and the number of μCT voxels in the studied samples, it is easy to infer that the use of hash tables is significantly advantageous for the overall performance of the algorithm, even though indexing operations continue to remain one of the major bottlenecks.

Phase	*Runtime (ms)*			
	Trabecular specimen (1.7M voxels)	*Cellular foam (12.4M voxels)*	*Glenoid (36.2M voxels)*	*Scapula (558K voxels)*
Create Voxel Grid	25.057	189.135	641.875	17.358
Create Hexahedral	196.085	1,383.37	4631.19	69.359
Nodes Indexing	1,354.804	12,888.198	37,013.259	343.325
Calculate Material	15.743	147.407	500.394	9.335
Material Binning	N/A	35.414	97.415	N/A
Material Indexing	387.306	3,301.615	12,279.429	109.241
Sum	1,978.995	17,945.139	55,163.562	548.798

Table 3: Breakdown of runtime time for different phases of the proposed algorithm.

Peak memory usage for each model is reported in Tab. 4. While the algorithm did not run out of memory in none of the analyzed cases, it is expected that the larger models will require excessive computing memory. On the other hand, since the construction of each hexahedral element is independent from the rest of the elements, it is practically possible to use out-of-core implementations in order to accommodate meshing larger models. However, it is reasonable to expect that the slower access to auxiliary/external-to-CPU memory will negatively impact the overall computing time. Figure 3 depicts the algorithm-generated μ-FEMs.

Model	*Peak memory usage*
Trabecular Specimen	359.4 MB
Cellular foam	3.65 GB
Glenoid	10.83 GB
Scapula	213 MB

Tab. 4: Peak memory usage for the tested models

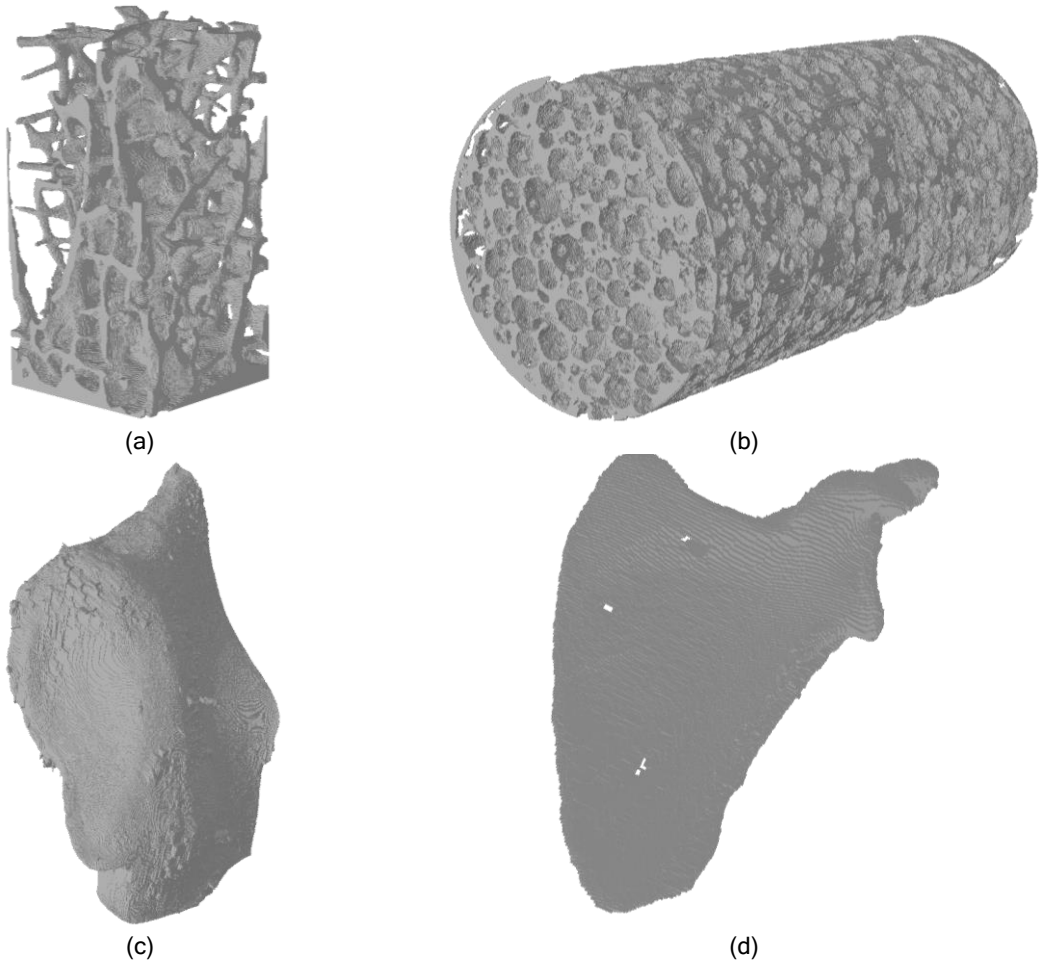

Figure 3: Generated μ-FEM for: (a) trabecular, (b) cellular foam, (c) glenoid, and (d) scapula samples.

6.2 Running Time Break Down - Voxel Up Sampling

To investigate the effect of voxel resizing on algorithm runtime, a new μFEM was generated by fragmenting each μCT voxel into 8 smaller voxels. This results in an isotropic resolution of 16 μm. The gray-values of the new voxels were obtained through linear interpolation of the gray-values obtained from the native μCT voxels. As shown in Tab. 5, the voxel resizing step - tested on the trabecular specimen - needed only an additional 4.54 s in order to up-sample more than 1.7M voxels. However, the overall computing time of the algorithm has experienced a significant increase due to the considerably smaller size of the mesh that was generated at this time. Nevertheless, the total runtime of 20.74 s remains remarkable, particularly when considering that the total size of the mesh is in excess of 13.6M hexahedrons/FE elements. The results suggest that the parameter that significantly affects the algorithm performance is represented by the model size, i.e., the sheer number of μCT voxels to be processed/converted into hexahedral elements.

Phase	*Runtime (ms)*
Create Voxel Grid	23.163
Resizing Voxels	4,542.848
Create Hexahedral	1,470.698
Nodes Indexing	11,330.332

Calculate Material	122.155
Material Indexing	3,255.170
Sum	20,744.366

Table 5: Breakdown of the running time for trabecular specimen undergoing up-sampling.

6.3 Time Complexity Analysis

To further investigate the functional relationship between the model size n and the algorithm runtime t, various decimations of the glenoid model were tested such that their log-log dependence could be graphically represented. Figure 4 reveals that for large datasets, the slope of the plot approaches unity:

$$\log\left(\frac{t_2}{t_1}\right) = \log\left(\frac{n_2}{n_1}\right) \rightarrow \frac{t_2}{t_1} = \frac{n_2}{n_1}, \tag{3.4}$$

This practically implies that runtime increases linearly with the size of the input, a finding that echoes well the linear time complexity that mentioned in the previous sections. Figure 4 also seems to suggest

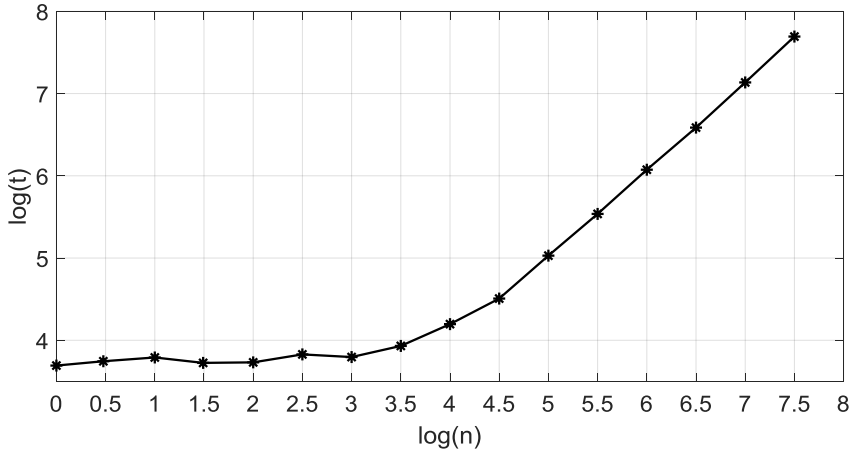

Figure 4: Relationship between the model size n and the algorithm runtime t measured in milliseconds.

that small datasets are characterized by a constant time complexity which might be nothing but a consequence of the dominant initialization overhead on the overall small runtime associated with small datasets.

7 DISCUSSION

As revealed by several numerical tests, the algorithm is capable to reduce the mesh generation time to under minute values for models comprised with as much as 36M voxels. Perhaps even more importantly, all tests were run on mid/low end/inexpensive computer hardware. Evidently, this level of performance is primarily owed to the implementation of grid-based approaches as well as hashing techniques that have enabled the concurrent achievement of a number of desirable software characteristics such as: memory efficiency, minimal number of floating point operations, fast and robust computations, implicit definitions of μCT voxels, facile up-sample of the voxels, etc. Above all, the linear time complexity behavior is set to guarantee the performance of the algorithm even for very large datasets.

Another important feature of the proposed method resides in its flexibility to be further enhanced in different directions. For example, since hexahedral elements are constructed independently per μCT voxel, construction of the mesh geometry can be effectively accelerated by a multi-threaded implementation using either CPU or GPU parallel computing platforms. It is also easy to utilize out-of-core methods in order to manage large models with minimal memory needs. Furthermore, the algorithm allows the incorporation of different material mapping methods in order to better replicate the structural characteristics of the model. The proposed approach also lends well to

direct conversions of CT data into finite element models. Although the automatic segmentation of CT images remains a challenge, the integration of image processing techniques could reduce data preprocessing time with further positive implications on the productivity of the downstream FEA analysis.

One of the possible limitations of the proposed method is represented by the fact that the generated mesh is constrained to have equally shaped and sized elements. Even though this type of limitation would be widely preferred over the conventional tetrahedral mesh alternative, one of the possible extensions of the current work could be focused on the generation of adaptively-sized hexahedral elements (possibly dimensioned through geometry curvature tracking algorithms), particularly since this would translate into a higher modeling accuracy of the generated μFEM. However, it can be anticipated that this possible enhancement direction is far from being trivial, particularly in order to address the occurrence of mismatched/hanging mesh nodes/vertices. Evidently, all these future investigational scenarios would lead to the generation of "true" (i.e., nonuniform/uneven) hexahedral meshes as opposed to their present Cartesian aspect.

8 CONCLUSION

The primary objective of the present study was to develop a fast and simple algorithm to facilitate the tedious process of deriving μFEMs from μCT data. The developed algorithm leverages several simple yet effective data structures and methods, primarily in the form of grid-based and hash mapping techniques, to speed up mesh generation processes that would otherwise crash the commercially available FEA software used for similar purposes. The developed algorithm is insensitive to geometrical complexity of the models and can generate hexahedral instead of tetrahedral elements. Of note, all the aforementioned features are practically lacking from any of the commercial FEA platforms that were scrutinized/reviewed prior to the start of this work. The algorithm can be regarded as a step forward towards the achievement of a superior level of performance/productivity during the structural FEA of biomechanical osseous models derived from μCT data.

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.

Mohammadreza Faieghi, http://orcid.org/0000-0003-0075-2969
Nikolas K. Knowles, http://orcid.org/0000-0001-8461-8104
O. Remus Tutunea-Fatan, http://orcid.org/0000-0002-1016-5103
Louis M. Ferreira, http://orcid.org/0000-0001-9881-9177

REFERENCES

[1] Baker, T. J.: Mesh generation: Art or science?, Progress in Aerospace Sciences, 41(1), 2005, 29-63. http://dx.doi.org/https://doi.org/10.1016/j.paerosci.2005.02.002
[2] Bathe, K.-J. r.: Finite element procedures, Prentice Hall, Englewood Cliffs, N.J., USA, 1996.
[3] Bourne, B. C.; van der Meulen, M. C.: Finite element models predict cancellous apparent modulus when tissue modulus is scaled from specimen CT-attenuation, Journal of Biomechanics, 37(5), 2004, 613-621. https://doi.org/10.1016/j.jbiomech.2003.10.002
[4] Bouxsein, M. L.; Boyd, S. K.; Christiansen, B. A.; Guldberg, R. E.; Jepsen, K. J.; Müller, R.: Guidelines for assessment of bone microstructure in rodents using micro-computed tomography, Journal of Bone and Mineral Research, 25(7), 2010, 1468-1486. https://doi.org/10.1002/jbmr.141
[5] Burkhart, T. A.; Andrews, D. M.; Dunning, C. E.: Finite element modeling mesh quality, energy balance and validation methods: A review with recommendations associated with the modeling of bone tissue, Journal of Biomechanics, 46(9), 2013, 1477-1488. https://doi.org/10.1016/j.jbiomech.2013.03.022
[6] Cattaneo, P.; Dalstra, M.; Frich, L. H.: A three-dimensional finite element model from computed tomography data: a semi-automated method, Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine, 215(2), 2001, 203-212. https://doi.org/10.1243/0954411011533760
[7] Chevalier, Y.; Pahr, D.; Allmer, H.; Charlebois, M.; Zysset, P.: Validation of a voxel-based FE method for prediction of the uniaxial apparent modulus of human trabecular bone using macroscopic mechanical tests

and nanoindentation, Journal of Biomechanics, 40(15), 2007, 3333-3340. https://doi.org/10.1016/j.jbiomech.2007.05.004

[8] Christen, D.; Melton, L. J.; Zwahlen, A.; Amin, S.; Khosla, S.; Müller, R.: Improved Fracture Risk Assessment Based on Nonlinear Micro-Finite Element Simulations From HRpQCT Images at the Distal Radius, Journal of Bone and Mineral Research, 28(12), 2013, 2601-2608. https://doi.org/10.1002/jbmr.1996

[9] Costa, M. C.; Tozzi, G.; Cristofolini, L.; Danesi, V.; Viceconti, M.; Dall'Ara, E.: Micro Finite Element models of the vertebral body: Validation of local displacement predictions, PloS one, 12(7), 2017, e0180151. https://doi.org/10.1371/journal.pone.0180151

[10] Ericson, C.: Real-time collision detection, CRC Press, 2004.

[11] Fagan, M. J.: Finite element analysis : theory and practice, Longman Scientific & Technical, London, UK, 1992.

[12] Geng, J.-P.; Tan, K. B.; Liu, G.-R.: Application of finite element analysis in implant dentistry: a review of the literature, Journal of Prosthetic Dentistry, 85(6), 2001, 585-598. https://doi.org/10.1067/mpr.2001.115251

[13] Holdsworth, D. W.; Thornton, M. M.: Micro-CT in small animal and specimen imaging, Trends in Biotechnology, 20(8), 2002, S34-S39. https://doi.org/10.1016/S0167-7799(02)02004-8

[14] Ji, S.; Ford, J. C.; Greenwald, R. M.; Beckwith, J. G.; Paulsen, K. D.; Flashman, L. A.; McAllister, T. W.: Automated subject-specific, hexahedral mesh generation via image registration, Finite Elements in Analysis and Design, 47(10), 2011, 1178-1185. https://doi.org/10.1016/j.finel.2011.05.007

[15] Josuttis, N. M.: The C++ standard library: a tutorial and reference, Addison-Wesley, 2012.

[16] Keaveny, T. M.; Morgan, E. F.; Niebur, G. L.; Yeh, O. C.: Biomechanics of trabecular bone, Annual Review of Biomedical Engineering, 3(1), 2001, 307-333. https://doi.org/10.1146/annurev.bioeng.3.1.307

[17] Nielson, G. M., Dual Marching Cubes, in Proceedings of the conference on Visualization '042004, IEEE Computer Society, 489-496. https://doi.org/10.1109/VISUAL.2004.28

[18] Pistoia, W.; Van Rietbergen, B.; Lochmüller, E.-M.; Lill, C.; Eckstein, F.; Rüegsegger, P.: Estimation of distal radius failure load with micro-finite element analysis models based on three-dimensional peripheral quantitative computed tomography images, Bone, 30(6), 2002, 842-848. https://doi.org/10.1016/S8756-3282(02)00736-6

[19] Polgar, K.; Viceconti, M.; Connor, J.: A comparison between automatically generated linear and parabolic tetrahedra when used to mesh a human femur, Proceedings of the Institution of Mechanical Engineers, Part H: Journal of Engineering in Medicine, 215(1), 2001, 85-94. https://doi.org/10.1243/0954411011533562

[20] Ramezanzadehkoldeh, M.; Skallerud, B. H.: MicroCT-based finite element models as a tool for virtual testing of cortical bone, Medical Engineering and Physics, 46(2017, 12-20. https://doi.org/10.1016/j.medengphy.2017.04.011

[21] Rao, S.: Sams Teach Yourself C++ in One Hour a Day, Sams Publishing, 2012.

[22] Shefelbine, S. J.; Simon, U.; Claes, L.; Gold, A.; Gabet, Y.; Bab, I.; Müller, R.; Augat, P.: Prediction of fracture callus mechanical properties using micro-CT images and voxel-based finite element analysis, Bone, 36(3), 2005, 480-488. https://doi.org/10.1016/j.bone.2004.11.007

[23] Torcasio, A.; Zhang, X.; Van Oosterwyck, H.; Duyck, J.; van Lenthe, G. H.: Use of micro-CT-based finite element analysis to accurately quantify peri-implant bone strains: a validation in rat tibiae, Biomechanics and Modeling in Mechanobiology, 11(5), 2012, 743-750. https://doi.org/10.1007/s10237-011-0347-6

[24] van Rietbergen, B.: Micro-FE analyses of bone: state of the art, in Noninvasive assessment of trabecular bone architecture and the competence of bone, 2001, Springer, 21-30. https://doi.org/10.1007/978-1-4615-0651-5_3

[25] Wagner, D. W.; Lindsey, D. P.; Beaupre, G. S.: Deriving tissue density and elastic modulus from microCT bone scans, Bone, 49(5), 2011, 931-938. https://doi.org/10.1016/j.bone.2011.07.021

[26] Wang, E.; Nelson, T.; Rauch, R.: Back to elements-tetrahedra vs. hexahedra, In: Proceedings of Proceedings of the 2004 International ANSYS Conference, ANSYS Pennsylvania, 2004,

[27] Wolfram, U.; Wilke, H.-J.; Zysset, P. K.: Valid μ finite element models of vertebral trabecular bone can be obtained using tissue properties measured with nanoindentation under wet conditions, Journal of Biomechanics, 43(9), 2010, 1731-1737. https://doi.org/10.1016/j.jbiomech.2010.02.026

[28] Yeni, Y. N.; Fyhrie, D. P.: Finite element calculated uniaxial apparent stiffness is a consistent predictor of uniaxial apparent strength in human vertebral cancellous bone tested with different boundary conditions, Journal of Biomechanics, 34(12), 2001, 1649-1654. https://doi.org/10.1016/S0021-9290(01)00155-5

[29] Zaitoun, N. M.; Aqel, M. J.: Survey on image segmentation techniques, Procedia Computer Science, 65(2015, 797-806. https://doi.org/10.1016/j.procs.2015.09.027

[30] Zhang, Y.; Bajaj, C.: Adaptive and quality quadrilateral/hexahedral meshing from volumetric data, Computer Methods in Applied Mechanics and Engineering, 195(9-12), 2006, 942-960. https://doi.org/10.1016/j.cma.2005.02.016