sensors

Article

Effects of Different Parameter Settings for 3D Data Smoothing and Mesh Simplification on Near Real-Time 3D Reconstruction of High Resolution Bioceramic Bone Void Filling Medical Images

Daniel Jie Yuan Chin [1], Ahmad Sufril Azlan Mohamed [1,*], Khairul Anuar Shariff [2,3], Mohd Nadhir Ab Wahab [1] and Kunio Ishikawa [4]

[1] School of Computer Sciences, Universiti Sains Malaysia, Gelugor 11800, Penang, Malaysia; danielcjy96@student.usm.my (D.J.Y.C.); mohdnadhir@usm.my (M.N.A.W.)
[2] School of Materials and Mineral Resources Engineering, Universiti Sains Malaysia, Engineering Campus, Nibong Tebal 14300, Penang, Malaysia; biokhairul@usm.my
[3] Dental Materials Science and Technology Division, Faculty of Dental Medicine, Airlangga University, JI. Prof. Dr. Moestopo No. 47, Surabaya 60132, East Java, Indonesia
[4] Department of Biomaterials, Faculty of Dental Science, Kyushu University, 3-1-1 Maidashi, Higashi-ku, Fukuoka 812-8582, Japan; ishikawa@dent.kyushu-u.ac.jp
* Correspondence: sufril@usm.my

check for updates

Citation: Chin, D.J.Y.; Mohamed, A.S.A.; Shariff, K.A.; Ab Wahab, M.N.; Ishikawa, K. Effects of Different Parameter Settings for 3D Data Smoothing and Mesh Simplification on Near Real-Time 3D Reconstruction of High Resolution Bioceramic Bone Void Filling Medical Images. *Sensors* **2021**, *21*, 7955. https://doi.org/10.3390/s21237955

Academic Editor: Manuchehr Soleimani

Received: 24 October 2021
Accepted: 25 November 2021
Published: 29 November 2021

Abstract: Three-dimensional reconstruction plays a vital role in assisting doctors and surgeons in diagnosing the healing progress of bone defects. Common three-dimensional reconstruction methods include surface and volume rendering. As the focus is on the shape of the bone, this study omits the volume rendering methods. Many improvements have been made to surface rendering methods like Marching Cubes and Marching Tetrahedra, but not many on working towards real-time or near real-time surface rendering for large medical images and studying the effects of different parameter settings for the improvements. Hence, this study attempts near real-time surface rendering for large medical images. Different parameter values are experimented on to study their effect on reconstruction accuracy, reconstruction and rendering time, and the number of vertices and faces. The proposed improvement involving three-dimensional data smoothing with convolution kernel Gaussian size 5 and mesh simplification reduction factor of 0.1 is the best parameter value combination for achieving a good balance between high reconstruction accuracy, low total execution time, and a low number of vertices and faces. It has successfully increased reconstruction accuracy by 0.0235%, decreased the total execution time by 69.81%, and decreased the number of vertices and faces by 86.57% and 86.61%, respectively.

Keywords: 3D reconstruction; 3D data smoothing; mesh simplification; high resolution micro-CT images

1. Introduction

Bone void fillers are used to treat bone voids caused by various reasons. Traditionally, doctors and surgeons examine bone defects implanted with a bone void filler by scanning it through a computed tomography (CT) scanner, which results in a stack of two-dimensional (2D) CT images. However, it is difficult to judge the healing progress of the bone defect as there are noises and artefacts present in the 2D CT bone defect images, which obscure the details of the bone defect. Additionally, it is difficult to visualize the bone defect as a whole with the 2D CT image stack. Therefore, by visualizing the 2D CT image stack as a three-dimensional (3D) view, 3D models can help doctors increase the quality of experience and diagnosis accuracy [1,2]. The process of visualizing the 2D CT image stack as a 3D view is called 3D reconstruction.

With the advancement in 3D technologies, more and more domains are adopting 3D reconstruction technologies. As this study emphasizes 3D reconstruction using micro-CT, images which are medical images, 3D reconstruction in the medical imaging area is studied. For example, the annulus fibrosus (AF) in the intervertebral disc (IVD) is reconstructed, visualized, and tracked from diffusion tensor imaging (DTI) images [3]. Furthermore, 3D models generated through 3D reconstruction are also used for measurement purposes in which the measurements are useful in surgical planning [4,5]. Moreover, measurements obtained from the reconstructed 3D models are also beneficial in virtual simulation for preoperative plans and personalized surgery [6–8]. 3D reconstruction algorithms are generally grouped into two categories: surface rendering and volume rendering [9,10].

Both methods have different advantages and disadvantages. For instance, if the main focus of the visualization is on the surface and shape of the bone, then surface rendering is more suitable than volume rendering. If the main focus of the visualization is on the internal structures, then volume rendering is more suitable than surface rendering. Therefore, as the main focus is on visualizing the surface of the bone defect, the study will only be focusing on surface rendering methods.

Surface rendering is a 3D reconstruction method that deposits isosurface comprising triangular patches on the segmented region of interest (ROI) per medical image slice. This results in a 3D model made up of isosurface stacks. Standard surface rendering techniques discussed in recent years are the Marching Cubes algorithm and the Marching Tetrahedra algorithm. The Marching Cubes algorithm is a popular surface rendering method applied in the medical field, mainly because of its implementation simplicity and relatively fast reconstruction speed. Marching Cubes uses a cube as a unit when forming the isosurface. The cube configurations formed during the triangulation step can be generally categorized into 15 unique patterns identified in the original Marching Cubes algorithm [11]. However, ambiguity issues will occur during the triangulation step, which leads to the formation of "holes" on isosurfaces.

Despite this, the original Marching Cubes algorithm is widely applied in recent publications for reconstruction purposes. For instance, it is used in the construction of polyhedral element meshes [12]. It is also used for lumbar intervertebral disk herniation diagnosis [2]. Improvements over the original Marching Cubes algorithm are also made in recent years. For example, Wang et al. [13] proposed the application of Laplacian smoothing to overcome the ambiguity issue by eliminating zero probability during isosurface extraction, edge collapse method using Quadric Error Metric [14], and polygon merging method through polygon normal regression in the Marching Cubes algorithm to improve the display and interaction speed. A variation of the Marching Cubes comprising 33 unique cube configurations, called the Marching Cubes 33, is introduced by Chernyaev [15] to cover the majority of the complex trilinear function of topology cases that leads to an ambiguity issue, which is further extended by Custodio et al. [16] by grouping the cube configurations into either a simple leaves triangulation, tunnel triangulation, or interior point leaves triangulation. Wi et al. [17] proposed another improved Marching Cubes algorithm that applies Laplacian smoothing to overcome the ambiguity issue, the edge collapse method using Quadric Error Metric [14], and Taubin smoothing to improve mesh quality. Masala et al. [18] proposed an improved Marching Cubes algorithm by extending the original 15 cube configurations to 21 cube configurations to cover some of the complex topology cases identified by the authors. Wang et al. [19] also proposed an improved Marching Cubes algorithm by extending the 15 cube configurations to 24 cube configurations, which covers more complex triangulation cases, eliminating more "holes".

On the other hand, Marching Tetrahedra is a variant of Marching Cubes that uses a tetrahedron as a unit instead of a cube for isosurface [20]. One significant advantage of Marching Tetrahedra over Marching Cubes is that there is no ambiguity issue. However, a considerable number of vertices and faces are generated to represent the isosurface. There are also improvements in the Marching Tetrahedra algorithm in recent years. For example, Lu and Chen [21] used bisection iterations to shorten the time needed to calculate the intersection points and proposed a more straightforward redundant vertices elimination strategy as a post-reconstruction step. Bagley et al. [22] proposed an improved Marching

Tetrahedra algorithm by applying a simple bisection technique to discover the edge-cut locations, warped and retetrahedralized. Guo et al. [23] proposed another improved Marching Tetrahedra algorithm which involves removing wrongly connected tetrahedrons by looking into the neighborhood relation among the tetrahedrons in all directions. Finally, Ren et al. [24] proposed a more complex improved Marching Tetrahedra algorithm that uses the second-order tetrahedral element C3D10 as a unit during surface reconstruction.

Although many of the improvements focused on increasing the reconstruction accuracy and reducing the reconstruction time for relatively small image datasets, they are not tested with large image datasets, and not many papers made an in-depth study on the effects of the different parameter settings of the improvement methods towards the reconstruction accuracy, reconstruction time, and optimization of vertices and faces. Hence, this study focuses on three objectives:

1. To compare Marching Cubes and Marching Tetrahedra in terms of reconstruction accuracy for large image dataset;
2. To optimize the better surface rendering technique towards near real-time rendering and higher reconstruction accuracy, lower reconstruction and rendering time, and a lower number of vertices and faces by experimenting on different parameter value combinations;
3. To study the effects of the improvements' different parameter values on the reconstruction accuracy, reconstruction time, rendering time, and the number of vertices and faces.

2. Materials and Methods

A high-resolution micro-CT image dataset is collected from Dr. Khairul Anuar Shariff from the School of Materials and Mineral Resources Engineering, Universiti Sains Malaysia, Engineering Campus. The image dataset consists of 971 high-resolution micro-CT image slices scanned with Skyscan 1076. Every image slice sits at 1400 dots per inch (DPI) for horizontal and vertical resolution. The images are already sorted by their image slice number; hence, they do not need to be rearranged based on their position. The main reason for choosing this image dataset is that it can be reconstructed as a whole unit without running out of memory.

The overall flowchart is as illustrated in Figure 1. All the implementations are made in the matrix laboratory (MATLAB). Before the reconstruction takes place, the images need pre-processing so that the reconstruction algorithms can support them. Firstly, the ROI is labeled using the Canny edge detector with a threshold value of 0.5, which will also label some of the artefacts. The next pre-processing step is thresholding with a threshold value of 0.3, which removes some artefacts. Lastly, area filtering is applied with only one connected component retrieved to further reduce the artefacts. Because the thresholding and area filtering steps will also remove parts of the ROI, the percentage of ROI retained in the image before being concatenated into 3D volumetric data is kept above 90% for all images.

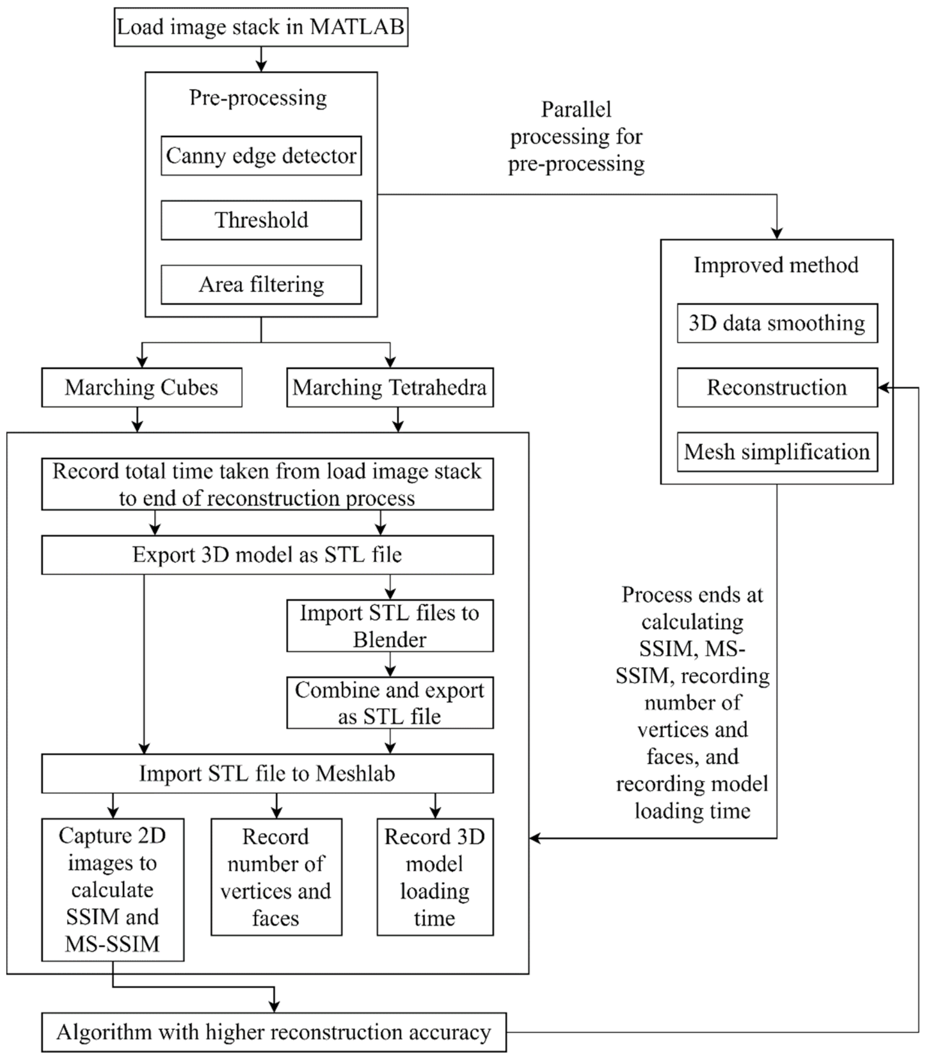
Load image stack in MATLAB
Pre-processing
Canny edge detector
Threshold
Area filtering
Parallel processing for pre-processing
Improved method
3D data smoothing
Reconstruction
Mesh simplification
Marching Cubes
Marching Tetrahedra
Record total time taken from load image stack to end of reconstruction process
Export 3D model as STL file
Import STL files to Blender
Combine and export as STL file
Import STL file to Meshlab
Capture 2D images to calculate SSIM and MS-SSIM
Record number of vertices and faces
Record 3D model loading time
Process ends at calculating SSIM, MS-SSIM, recording number of vertices and faces, and recording model loading time
Algorithm with higher reconstruction accuracy

Figure 1. Overall flowchart of methodology.

After the pre-processing step, the reconstruction of 3D volumetric data into 3D models takes place for the comparison study. The Marching Cubes code used in this paper is the implementation by Hammer [25]. A 3D model is generated after the reconstruction process, which is then exported out as a Standard Tessellation Language (STL) file using Sven's implementation [26]. The Marching Tetrahedra code implemented in this paper uses Hammer's Marching Cubes implementation as a codebase, vertices and faces orientation, and lookup table changes are also made so that it runs as Marching Tetrahedra instead of as Marching Cubes. The code changes refer to the source code written in C by Bourke [27]. The Marching Tetrahedra replicated in this study is the six tetrahedron variant; hence, after the reconstruction process, each tetrahedron configuration is exported as one individual STL file, totaling six STL files. They are then imported into Blender to combine them into one complete model before exporting it as another single STL file. This results in two 3D models, one from Marching Cubes and another one from Marching Tetrahedra. After successfully importing into Meshlab, the 2D images of the 3D models are captured as black and white images. The captured 2D images are then used to calculate the reconstruction accuracy with the 2D image of the bone defect provided in the dataset as ground truth.

This study uses the structural similarity index (SSIM) and multiscale structural similarity index (MS-SSIM) to evaluate the reconstruction accuracy. Both metrics measure the structural similarity over the 2D images, with the main difference in MS-SSIM being that it measures over different image scales through different image sub-sampling processes with low-pass filters [28]. Values of both metrics in MATLAB implementation are already normalized; therefore, no further action is needed. The higher the SSIM and MS-SSIM values, the higher the reconstruction accuracy. Therefore, the algorithm with higher SSIM and MS-SSIM values is the base for improvement, referred to as algorithm A in this chapter.

The improvement comprises applying two additional steps on top of algorithm A: the 3D data smoothing step and the mesh simplification step. The 3D data smoothing step is applied to the 3D volumetric data before the reconstruction step by algorithm A. This step removes noises in the 3D data, which may improve the 3D reconstruction accuracy and solve the ambiguity issue because noises often lead to wrong triangulations. The 3D data smoothing step applied in this paper is a MATLAB built-in function called *smooth3* [29], which convolves the 3D volumetric data with a convolution kernel at a defined size. In this paper, different convolution kernels, namely box and Gaussian, at different convolution kernel sizes (3, 5, and 11) are tested to investigate the effect of different convolution kernels and their sizes on the reconstruction accuracy. Convolution kernel sizes 3 and 5 are standard kernel sizes. Another commonly used kernel size is 10, but because *smooth3* only accepts odd-numbered convolution kernel size, size 11 is selected.

The mesh simplification step, which is applied after the reconstruction step using algorithm A, is a commonly applied method in reducing the number of vertices and faces whilst retaining the overall shape of the 3D model. This method is useful, especially when the 3D model is 3D printed, significantly reducing the slicing time. The mesh simplification step applied in this paper is also a MATLAB built-in function called *reducepatch* [30]. Unfortunately, there is no proper documentation on this particular MATLAB function, but based on the output of the 3D model, it is speculated that an adaptive remeshing based on a criterion or an error metric is applied. This MATLAB function accepts three parameters: a list of vertices, a list of faces, and a reduction factor between 0.0 and 1.0. The lists of vertices and faces are obtained through the reconstruction step. The reduction factor depicts what percentage of vertices and faces should be left after the reduction step. For example, a reduction factor of 0.1 means only 10% of the vertices and faces are left after the reduction step. In this paper, different reduction factors, starting with 0.1 and ending at 0.9, incrementing by 0.1 for each test case, are experimented on to study the effect of different reduction factors on the reconstruction accuracy.

After the mesh simplification step, the 3D models are exported as STL files and imported into Meshlab to capture the 2D black and white images. These images will be used in calculating the SSIM and MS-SSIM. In this paper, the total number of parameter value combinations tested are 2 different convolution kernels multiplied by 3 different convolution kernel sizes multiplied by 9 reduction factors, which is 54.

In an attempt towards making the reconstruction process for high-resolution medical images near real-time, the whole experiment is made using a laptop with 12 gigabytes (GB) random-access memory (RAM), 6 central processing unit (CPU) cores, and a NVIDIA Geforce RTX2060 graphics processing unit (GPU). MATLAB code changes are also made accordingly to support GPU and parallel processing. Firstly, all the arrays used in the code are defined as *gpuArray* objects stored in the GPU. This allows the code to run in the GPU. It is to be noted that not all functions in MATLAB support *gpuArray* objects and that MATLAB only supports the CUDA-enabled NVIDIA GPU [31]. Secondly, the looping over the image stacks during the pre-processing step is changed from *for* to *parfor* to support parallel processing [32]. The for-loop iterations are executed over a parallel pool of six workers defined by MATLAB during code execution. Lastly, although very minimal code changes were made, the reconstruction function used is the vectorized version of the original algorithm A. MATLAB is optimized to perform operations over array-based objects like matrices and vectors [33]; thus, any loops that can be vectorized will perform much faster than the loop itself. However, only loops that involve basic arithmetic operation over the matrix or vector can be vectorized. The speed-up due to the changes stated above can be evaluated by recording the reconstruction time and rendering time for both the original and improved algorithm A.

The reconstruction time refers to the total time taken from loading the image stack until the reconstruction ends. The reconstruction time does not cover the time to export the 3D models as STL files and is recorded in seconds. It is tabulated as the average over five runs. The rendering time refers to the total time taken to load the 3D model in 3D software, which refers to the time taken to load the 3D model as STL files in Meshlab. It is also recorded in seconds and tabulated as the average over five runs.

In addition to the reconstruction accuracy, the reconstruction time, and rendering time, the number of vertices and faces are also recorded for 3D models reconstructed with original and improved algorithm A. They are recorded by loading the 3D models in Meshlab. Finally, graphs are plotted to study the effect of 3D data smoothing and mesh simplification on the reconstruction accuracy, reconstruction time, rendering time, and the number of vertices and faces.

3. Results

For the table results in this chapter, MC refers to Marching Cubes, MT refers to Marching Tetrahedra, S refers to convolution kernel size for *smooth3*, and RF refers to the reduction factor for *reducepatch*.

3.1. Marching Cubes vs. Marching Tetrahedra

According to the results in Table 1, the 3D model reconstructed with Marching Cubes has a higher SSIM and MS-SSIM value than the 3D model reconstructed with Marching Tetrahedra, albeit by a minimal margin. This means Marching Cubes has higher reconstruction accuracy than Marching Tetrahedra. Reconstruction time, rendering time, and the number of vertices and faces are not compared here as the reconstruction accuracy is given the highest emphasis in this comparison study. Marching Cubes will be used as a codebase for improvement.

Table 1. SSIM and MS-SSIM for Marching Cubes and Marching Tetrahedra.

Reconstruction Method	SSIM (%)	MS-SSIM (%)
Marching Cubes	87.68	82.32
Marching Tetrahedra	87.66	82.27

3.2. Proposed Improved Marching Cubes with Different Parameter Value Combinations

The reconstructed 3D models with different parameter value combinations are as illustrated in Figure 2. Visually, as there is no difference between smoothing with box convolution kernel and Gaussian convolution kernel, all the images in Figure 2 represent both convolution kernels. Based on the results tabulated in Table A1, two-parameter value combinations are tied for the highest SSIM and MS-SSIM, which are Marching Cubes with a convolution kernel box size 11 and a reduction factor of 0.1, and Marching Cubes with a convolution kernel Gaussian size 11 and a reduction factor of 0.1, with both having a SSIM value of 87.77 and MS-SSIM value of 82.48. Between these two combinations, the one with the lower reconstruction and rendering time is Marching Cubes with a convolution kernel Gaussian size 11 and a reduction factor of 0.1. Regardless of the convolution kernel, as long as the size and reduction factors are the same, they will have the same SSIM, MS-SSIM, and number of vertices and faces.

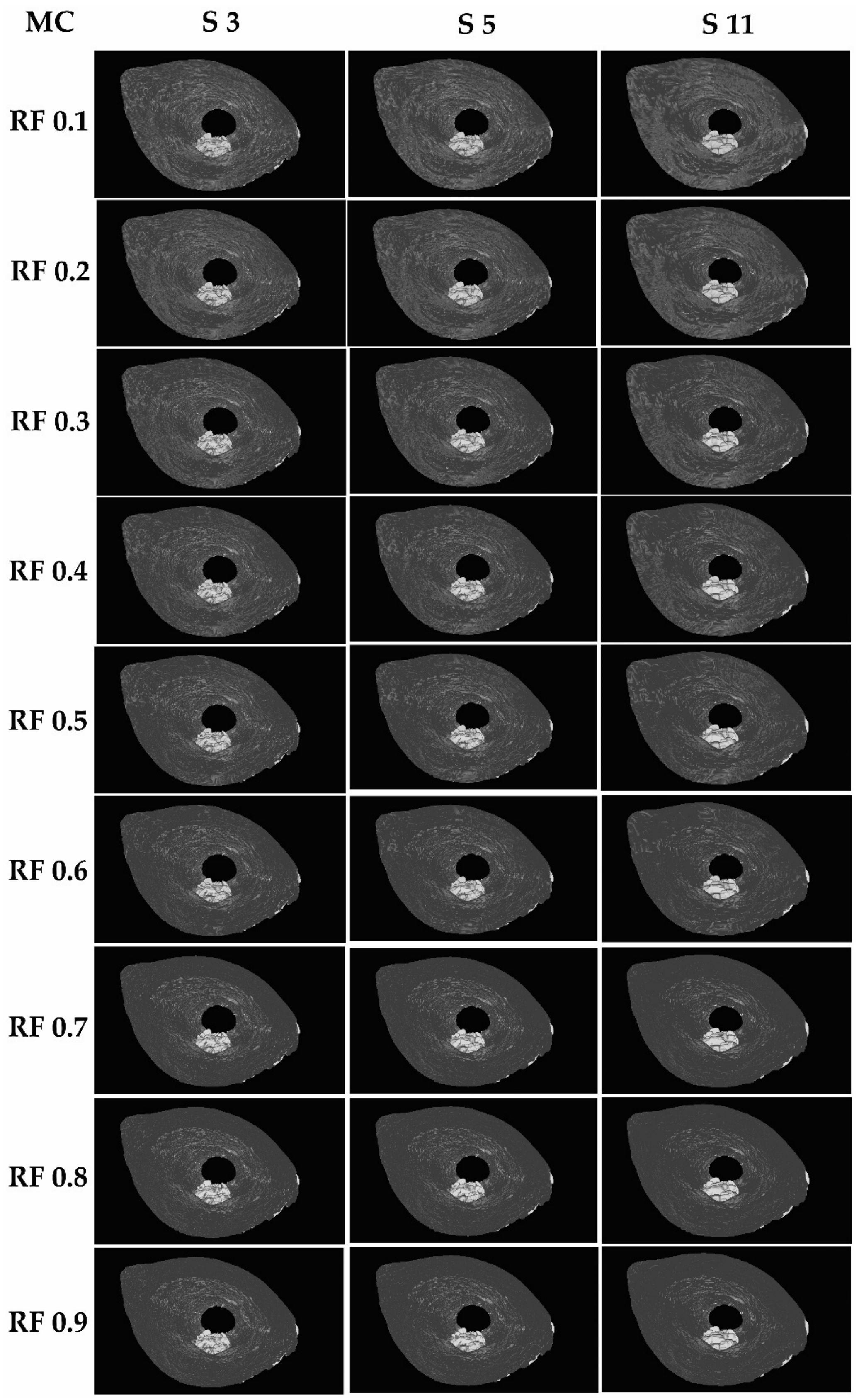

Figure 2. 3D models reconstructed with proposed improvement of different parameter value combinations.

Amongst all the parameter value combinations, the one with the fastest total execution time (reconstruction time added with rendering time) is Marching Cubes with a convolution kernel box size 3 and a reduction factor of 0.6, with the reconstruction time being 99.41 s and the rendering time being 11.13 s, totaling 110.54 s. As for the parameter value combination with the lowest number of vertices and faces, Marching Cubes with a convolution kernel box size 11 and a reduction factor of 0.1, and Marching Cubes with a convolution kernel Gaussian size 11 and a reduction factor of 0.1 are tied for having 315,490 vertices and 629,003 faces. Between these two, Marching Cubes with a convolution kernel Gaussian size 11 and a reduction factor of 0.1 has a lower total execution time.

To determine which parameter value combination is the most optimal to achieve high reconstruction accuracy, fast reconstruction and rendering time, and a low number of vertices and faces, firstly, the percentages of increase for the SSIM and MS-SSIM values, and percentages of decrease for the total time taken (reconstruction time added with rendering time), and the number of vertices and faces, are calculated for every parameter value combination. After calculating the percentages of increase and decrease, any negative values will be replaced with the value -1, and positive values are divided by 100 to become decimal form. These values are then multiplied with a weightage score. The weightage scores for SSIM and MS-SSIM are 0.3 and 0.3, respectively, totaling 0.6. The weightage score for the total time taken is 0.3, and the weightage scores for the number of vertices and faces are 0.05, respectively, totaling 0.1. Thus, the total weightage score is 1.0. After multiplying with the respective weightage score, the values are added to get the final score for every parameter value combination. The scores are tabulated in Table A2.

Based on the scores in Table A2, Marching Cubes with a convolution kernel Gaussian size 5 and a reduction factor of 0.1 has the highest score, obtaining 0.2962 out of 1.0. This means that this combination is the most optimal in achieving a reconstructed 3D model with high accuracy, low reconstruction and rendering time, and a low number of vertices and faces. As well as this, the proposed improvement (Marching Cubes with a convolution kernel Gaussian size 5 and a reduction factor of 0.1) managed to increase the SSIM and MS-SSIM values by 0.011% and 0.036%, respectively, decrease the total execution time by 69.81%, and decrease the number of vertices and faces by 86.57% and 86.61%, respectively. This means that the number of vertices and faces can be reduced by 86.57% and 86.61%, respectively, and still have relatively high reconstruction accuracy, which means a lot of the vertices and faces in 3D models reconstructed with the original Marching Cubes are unnecessary. They can be safely removed without affecting the shape, boundary, and accuracy of the 3D models. It is noticed that the scores for the convolution kernel box size 3 and Gaussian size 3 are all negative values across all the reduction factors. This happens when either one of the percentages of increase or decrease is a negative value, resulting in its value being replaced with -1 during the calculation of the combination score. This is to penalize the decrease in the reconstruction accuracy. Based on Table A1, it seems that either the SSIM or MS-SSIM or both the metrics' values are lower than the original Marching Cubes algorithm, therefore, resulting in negative values when calculating the percentage of increase in the SSIM and MS-SSIM values.

Additionally, it is noticed that the scores decrease when the reduction factor increases. This is mainly due to the increase in the total execution time and the number of vertices and faces as the reduction factor increases. This is normal as the larger the reduction factors, the higher the number of vertices and faces retained during the mesh simplification step. It is also noticed that the scores for the convolution kernel size 5 are higher than the scores for the convolution kernel size 11. This is largely due to the drastic increase in total execution time as shown in Table A1.

3.2.1. Effect of Convolution Kernel Size and Reduction Factor on Reconstruction Accuracy

In Figures 3 and 4, the convolution kernel size and reduction factor for both the convolution kernel box and Gaussian are merged into one as both have the same SSIM values. Based on Figure 3, the SSIM values across all reduction factors increased from the convolution kernel size 3 to convolution kernel size 5 before increasing to 87.77% for the convolution kernel size 11, except the reduction factors of 0.2, 0.3, and 0.4 whereby their SSIM values for the convolution kernel size 3 are slightly lower than the SSIM values for the reduction factors of 0.1, 0.5, 0.6, 0.7, 0.8, and 0.9. In Figure 4, the SSIM values across all three convolution kernel sizes remained near-constant as the reduction factor increased, except for the convolution kernel size 3, which showed a slight variant in the SSIM values from the reduction factor of 0.1 to 0.5, before remaining constant from 0.6 to 0.9.

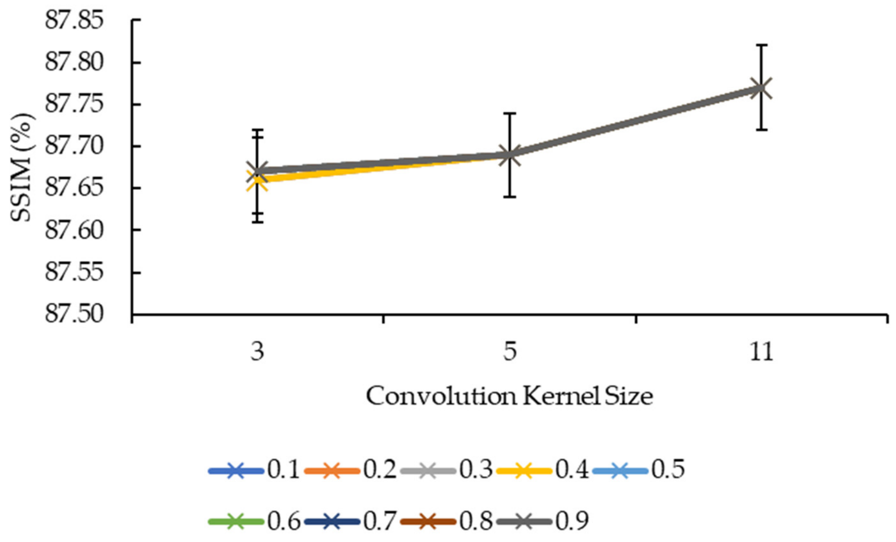

Figure 3. Graph of SSIM against convolution kernel size.

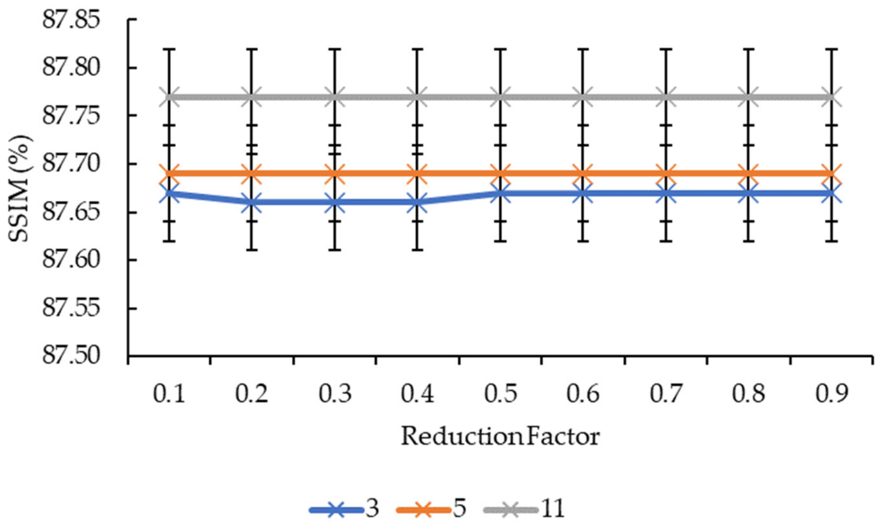

Figure 4. Graph of SSIM against reduction factor.

In Figures 5 and 6, the convolution kernel size and reduction factor for both the convolution kernel box and Gaussian are combined as both have the same MS-SSIM values. Based on Figure 5, the MS-SSIM values for reduction factors of 0.1, 0.2, 0.3, and 0.4 showed a slight increase from the convolution kernel size 3 to the convolution kernel size 5, before showing another slight increase from the convolution kernel size 5 to the convolution kernel size 11. This is different for the remaining reduction factors whereby the MS-SSIM values showed a big jump from convolution kernel size 3 to the convolution kernel size 5, before showing the same behavior, which is a slight increase from the convolution kernel size 5 to the convolution kernel size 11. In Figure 6, the MS-SSIM values showed the same behavior as the SSIM values for the convolution kernel sizes 5 and 11, whereby the values remained near-constant as the reduction factor increased, and that only the convolution kernel size 3 showed an extensive variant in the MS-SSIM values when the reduction factor increased from 0.4 to 0.5.

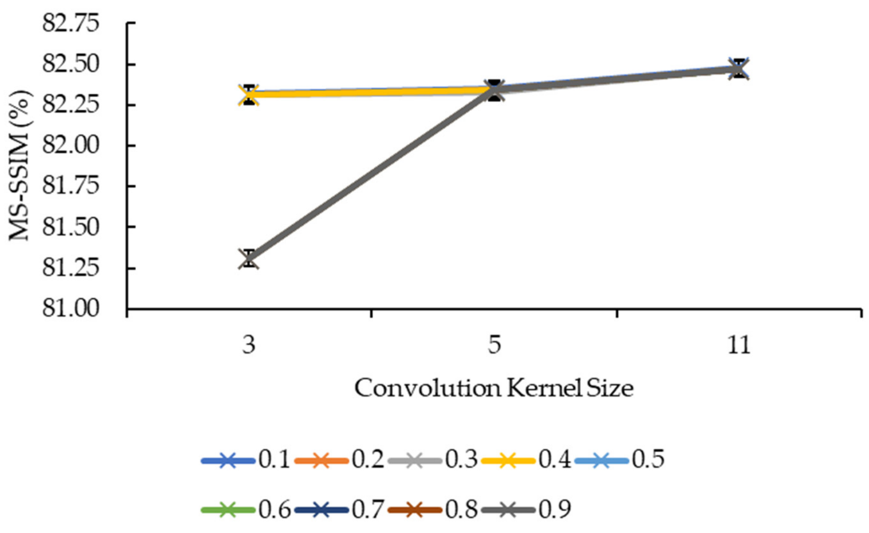

Figure 5. Graph of MS-SSIM against convolution kernel size.

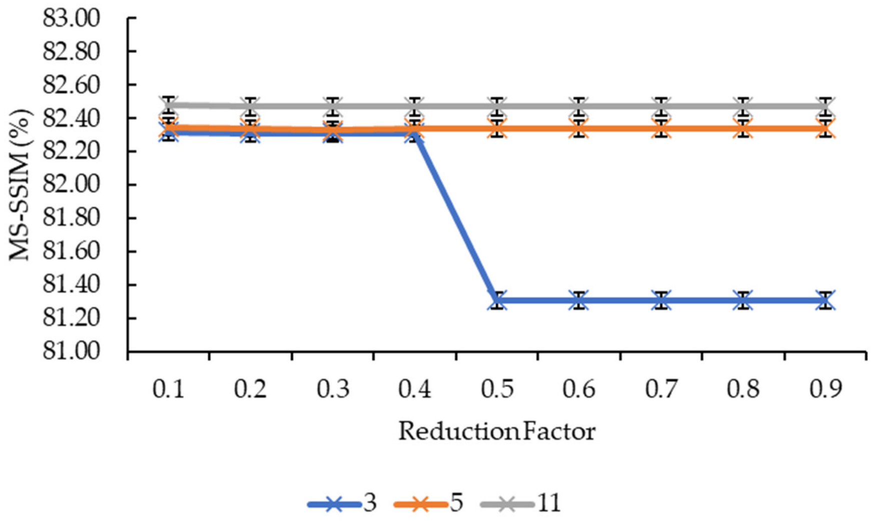

Figure 6. Graph of MS-SSIM against reduction factor.

3.2.2. Effect of Convolution Kernel Size and Reduction Factor on Reconstruction Time

Another four graphs are plotted based on the results obtained in Table A1. The first graph is plotted with the reconstruction time against the convolution kernel size for the convolution kernel box. The second graph is plotted with the reconstruction time against the reduction factor also for the convolution kernel box. The third and fourth graphs are essentially the same thing as the first and second graphs but for the convolution kernel Gaussian.

Referring to Figure 7, it seems that, overall, the reconstruction time increased when the convolution kernel box size increased for all reduction factors. All the reduction factors showed a significant increase in the reconstruction time when the convolution kernel box size increased from 5 to 11 compared to when convolution kernel box size increased from 3 to 5, except for the reduction factor of 0.7, whereby it showed a minor increase in the reconstruction time when the convolution kernel size increased from 5 to 11 compared to when the convolution kernel size increased from 3 to 5. Based on Figure 8, for the convolution kernel box size 3, other than the reconstruction time showing an increase as the reduction factor increased from 0.6 to 0.7, the rest showed a decrease in the reconstruction time as the reduction factor increased. For the convolution kernel box size 5, the line graph is much more complicated, with the reconstruction time showing an increase when the reduction factor increased from 0.2 to 0.3, 0.4 to 0.5, 0.6 to 0.7, and 0.8 to 0.9, while the rest showed a decrease in the reconstruction time. For the convolution kernel box size 11, the line graph showed a consistent increase in the reconstruction time when the reduction factor increased from 0.2 to 0.4 and from 0.7 to 0.9, while the rest showed a consistent decrease in the reconstruction time. The position of the line graph for the convolution kernel box size 11 supports the description for Figure 7, whereby it is clearly above the rest of the line graphs.

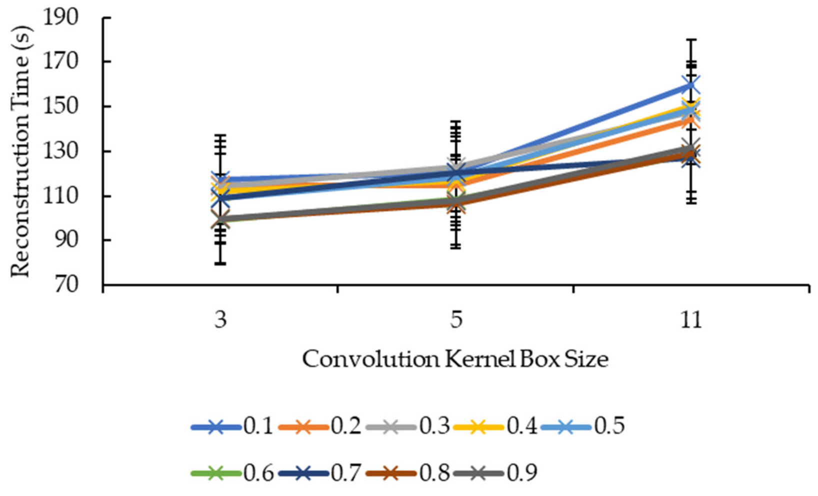

Figure 7. Graph of reconstruction time against convolution kernel size for convolution kernel box.

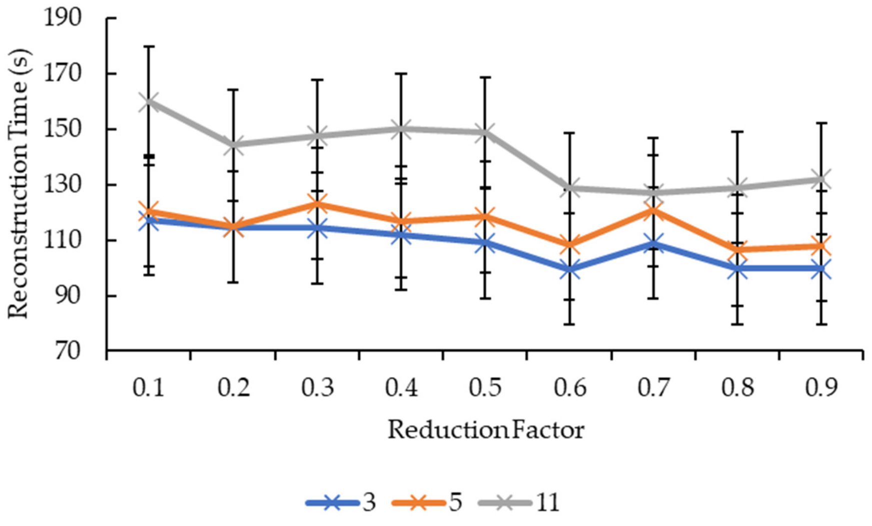

Figure 8. Graph of reconstruction time against reduction factor for convolution kernel box.

Based on Figure 9, all the reduction factors showed similar behavior as the convolution kernel box whereby a significant increase in the reconstruction time is seen when the convolution kernel size increased from 5 to 11 compared to when the convolution kernel size increased from 3 to 5, except for two reduction factors. The reduction factor of 0.4 showed a decrease in the reconstruction time when the convolution kernel Gaussian size increased from 3 to 5 before showing an increase in the reconstruction time when the convolution kernel size increased from 5 to 11. The reduction factor of 0.8, on the other hand, showed a near-linear increase in the reconstruction time as the convolution kernel Gaussian size increased. The decrease in the reconstruction time for the reduction factor of

0.4 when the convolution kernel Gaussian size increased from 3 to 5 is seen in Figure 10 whereby the line graph for the convolution kernel size 3 is above the line graph for the convolution kernel size 5 at the reduction factor of 0.4. Referring to Figure 10, besides the same placement of line graph for the convolution kernel Gaussian size 11 as the convolution kernel box size 11 in Figure 9, the line graphs in Figure 10 are also complex. For the convolution kernel Gaussian size 3, the reconstruction time increased when the reduction factor increased from 0.1 to 0.3, from 0.6 to 0.7, and from 0.8 to 0.9, while the rest showed a decrease in the reconstruction time. For the convolution kernel Gaussian size 5, the reconstruction time increased when the reduction factor increased from 0.1 to 0.3, from 0.4 to 0.5, and from 0.7 to 0.8, while the rest showed a decrease in the reconstruction time. The convolution kernel Gaussian size 11, on the other hand, showed an increase in the reconstruction time when the reduction factor increased from 0.1 to 0.2, from 0.3 to 0.5, and from 0.6 to 0.8, while the rest showed a decrease in the reconstruction time.

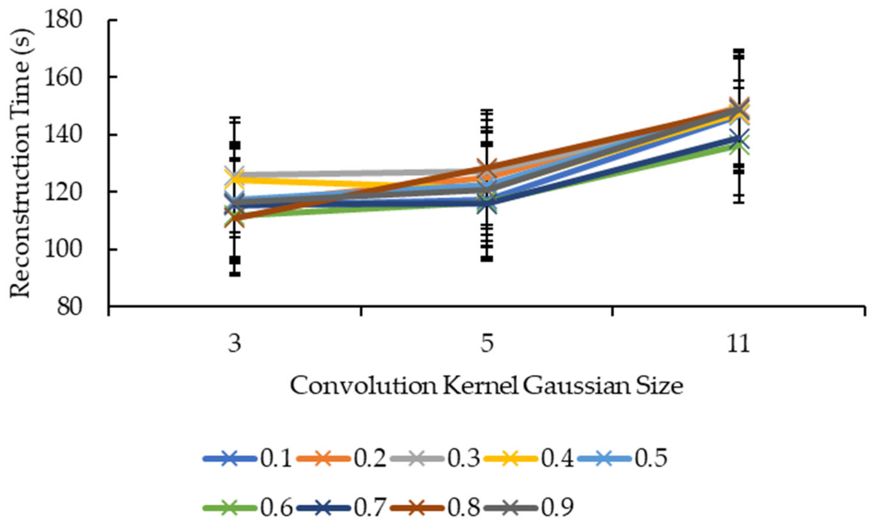

Figure 9. Graph of reconstruction time against convolution kernel size for convolution kernel Gaussian.

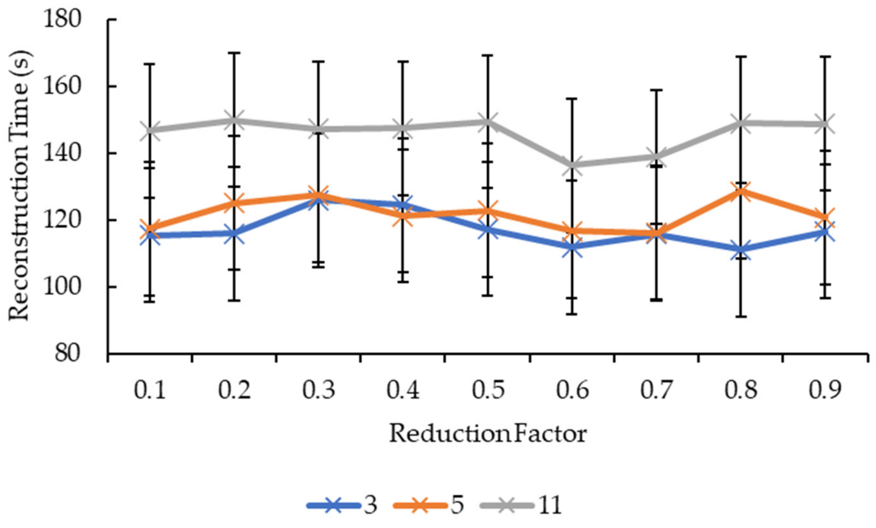

Figure 10. Graph of reconstruction time against reduction factor for convolution kernel Gaussian.

3.2.3. Effect of Convolution Kernel Size and Reduction Factor on Rendering Time

Different from Figures 7 and 8, Figures 11 and 12 show more direct lines. In Figure 11, all the reduction factors decreased the rendering time as the convolution kernel box size increased. The placement of the lines in the graph is higher as the reduction factor increases, which can be seen more clearly in Figure 12. In Figure 12, for all the convolution kernel box sizes, the rendering time increased logarithmically as the reduction factor increased. The lines show signs of plateauing at the reduction factor of 0.6 before starting to plateau at the reduction factor of 0.7 onwards, except for the convolution kernel box size 3 where it showed a slight variant in the rendering time from the reduction factors of 0.7 to 0.9. Moreover, there is a slight drop in the rendering time for the convolution kernel box sizes 5 and 11 when the reduction factor increased from 0.8 to 0.9.

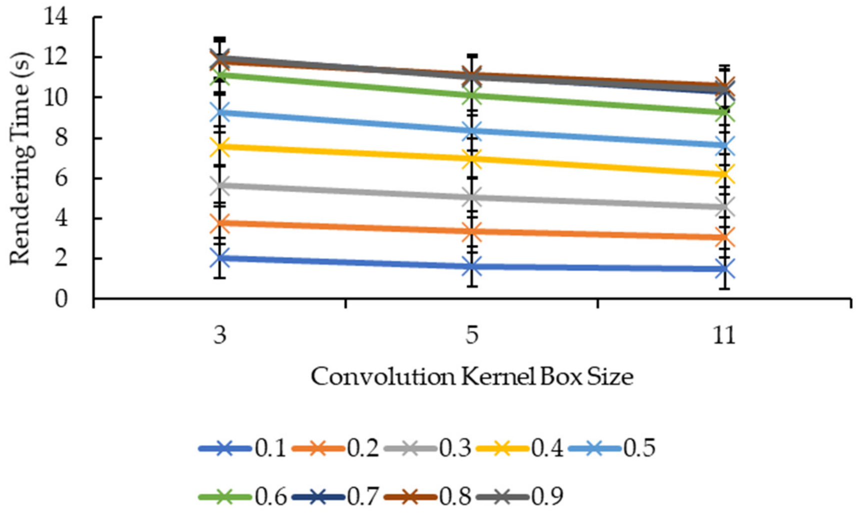

Figure 11. Graph of rendering time against convolution kernel size for convolution kernel box.

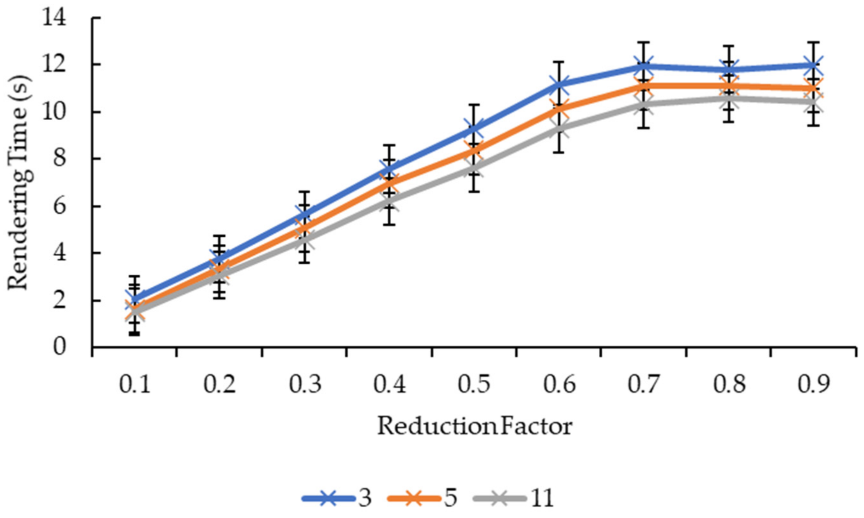

Figure 12. Graph of rendering time against reduction factor for convolution kernel box.

Figures 13 and 14, show nearly the same line shapes like the ones in Figures 11 and 12, respectively. In Figure 13, the rendering time decreases as the convolution kernel Gaussian size increases for all the reduction factors. Similarly, the placement of the lines in the graph shows that the rendering time increases as the reduction factor increases, which can be seen in Figure 14. Figure 14 also shows a logarithmic increase in the rendering time as the reduction factor increases for all the convolution kernel Gaussian sizes. However, for the convolution kernel Gaussian, only kernel size 11 slightly decreased the rendering time when the reduction factor increased from 0.8 to 0.9.

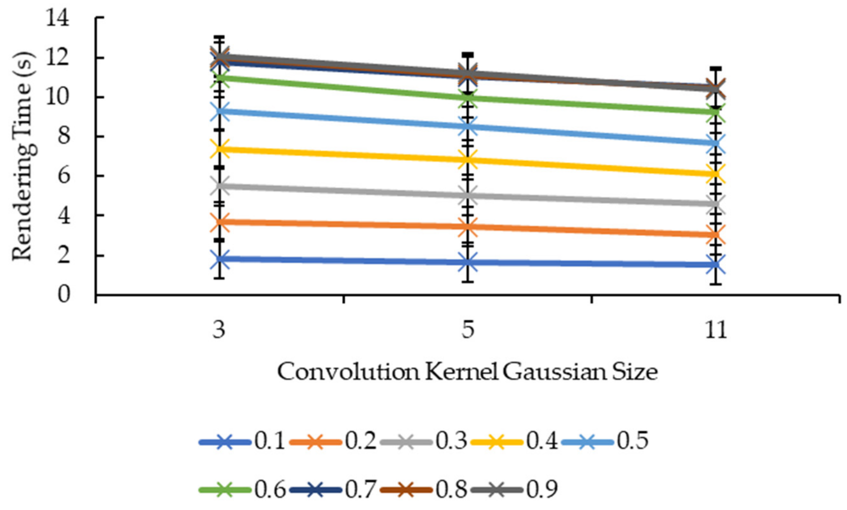

Figure 13. Graph of rendering time against convolution kernel size for convolution kernel Gaussian.

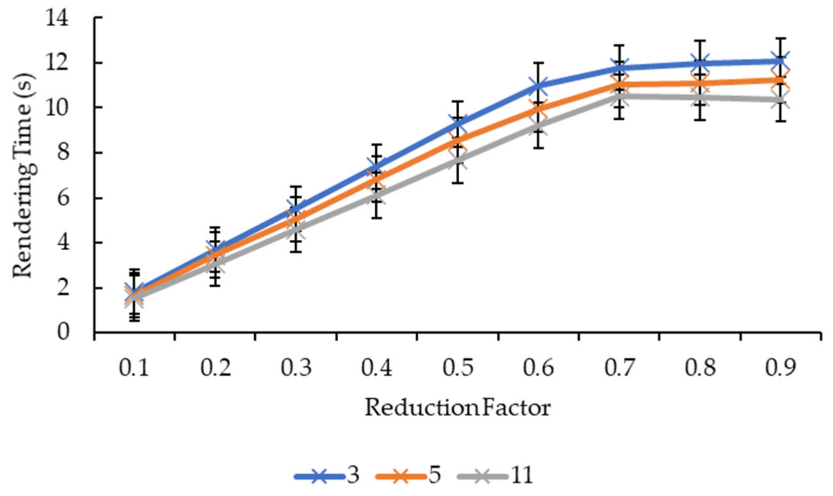

Figure 14. Graph of rendering time against reduction factor for convolution kernel Gaussian.

3.2.4. Effect of Convolution Kernel Size and Reduction Factor on Number of Vertices and Faces

Figures 15 and 16 showed very similar line shapes like the ones in Figures 11 and 13, and Figures 12 and 14, respectively. Based on Figure 15, all the reduction factors showed a decrease in vertices as the convolution kernel size increases for both box and Gaussian since both convolution kernels have the same number of vertices. The placement of the lines also showed that the number of vertices increases as the reduction factor increases, which is expected as more vertices are retained when the reduction factor increases. In Figure 16, the increase in the number of vertices as the reduction factor increases can be seen, in which the increase is logarithmic. The reduction factor of 0.6 is where the lines showed signs of plateauing and completely plateaued from the reduction factor of 0.7 onwards.

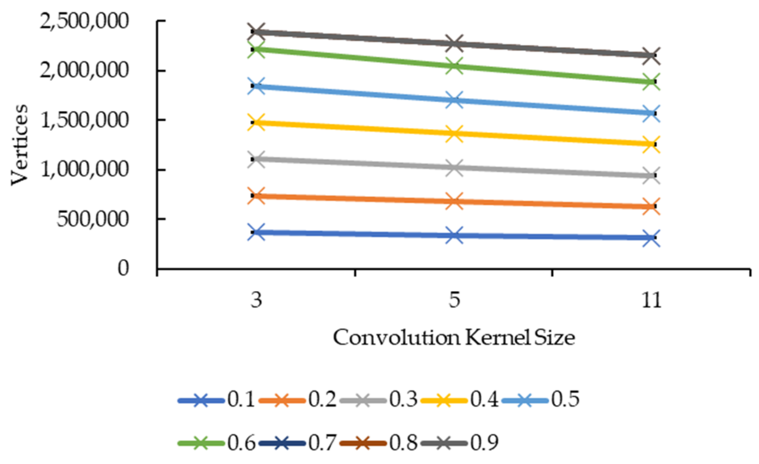

Figure 15. Graph of number of vertices against convolution kernel size.

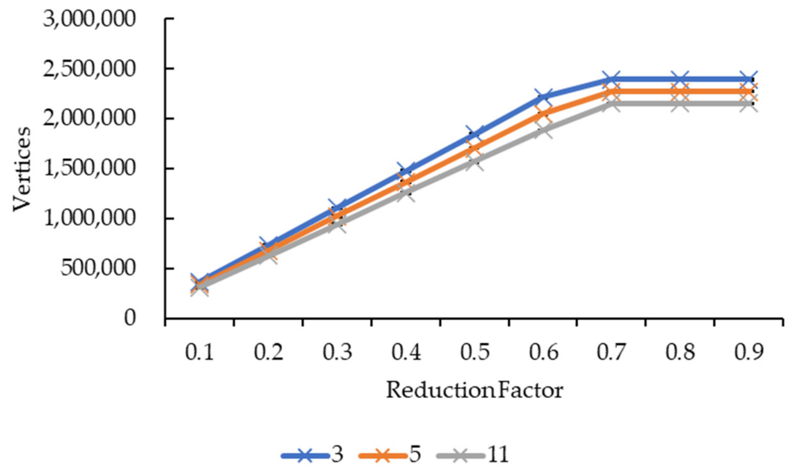

Figure 16. Graph of number of vertices against reduction factor.

Figures 17 and 18 show the same line patterns as in Figures 15 and 16, respectively. This is because faces are dependent on the vertices, hence, exhibiting the same pattern. In Figure 17, the number of faces decreases as the convolution kernel size increases for all reduction factors. Similarly, in Figure 18, the number of faces showed a logarithmic increase as the reduction factor increases for all the convolution kernel sizes in both the convolution kernel box and Gaussian. The line plateaued at a reduction factor of 0.7 onwards.

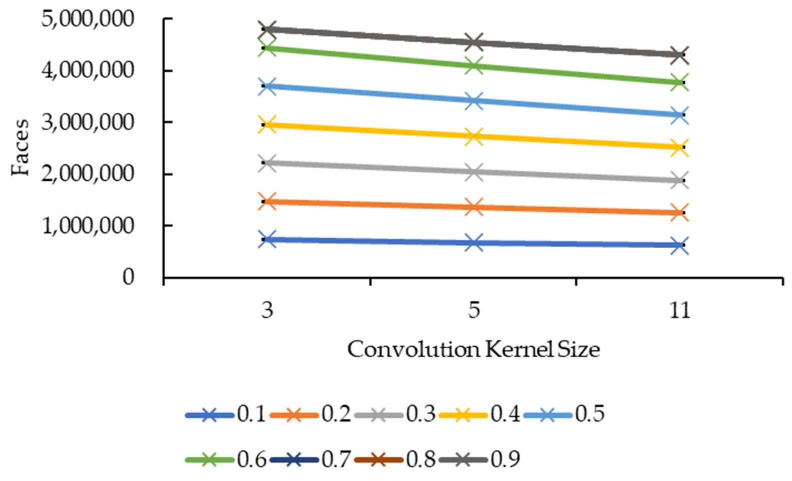

Figure 17. Graph of number of faces against convolution kernel size.

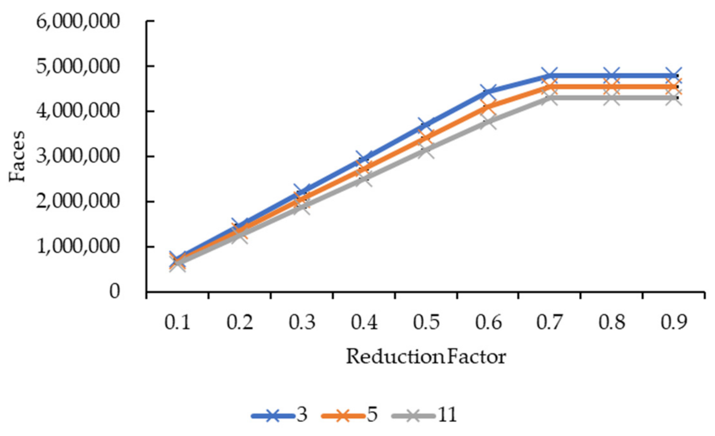

Figure 18. Graph of number of faces against reduction factor.

3.3. Marching Cubes vs. Wang et al. vs. Wi et al. vs. Proposed Enhancement

As well as comparing the original Marching Cubes and the proposed enhancement, works by Wang et al. [13] and Wi et al. [17] are replicated and compared in this study. Their works were selected to replicate as both involve the smoothing step and mesh decimation step in their improvement, and the proposed enhancement also involves smoothing and mesh decimation; hence, a direct comparison can be made. The result of the comparison is illustrated in Figures 19–21. The surfaces of the reconstructed 3D models are illustrated in Figure 22.

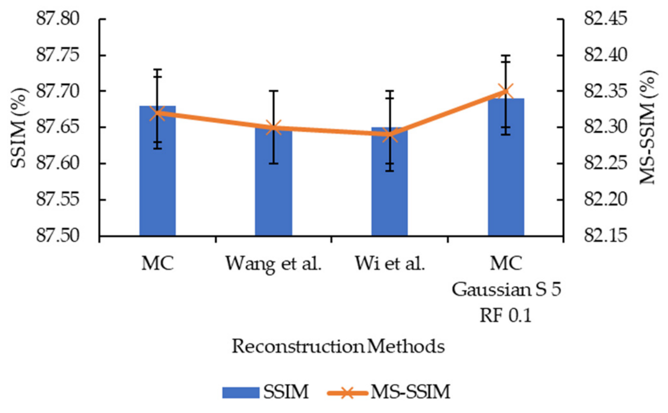

Figure 19. Graph of SSIM and MS-SSIM against different reconstruction methods.

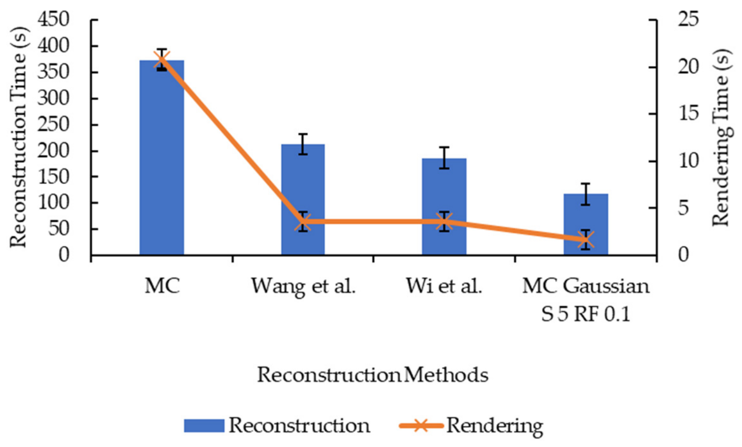

Figure 20. Graph of reconstruction and rendering time against different reconstruction methods.

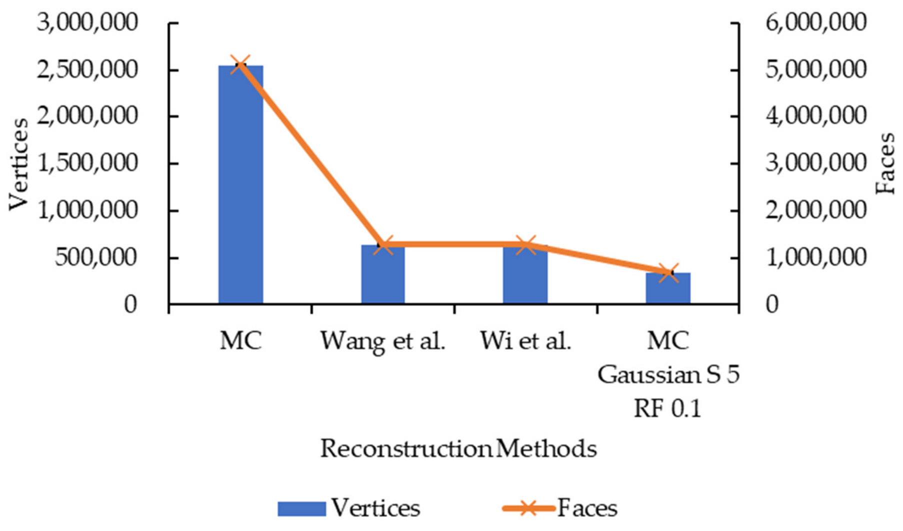

Figure 21. Graph of number of vertices and faces against different reconstruction methods.

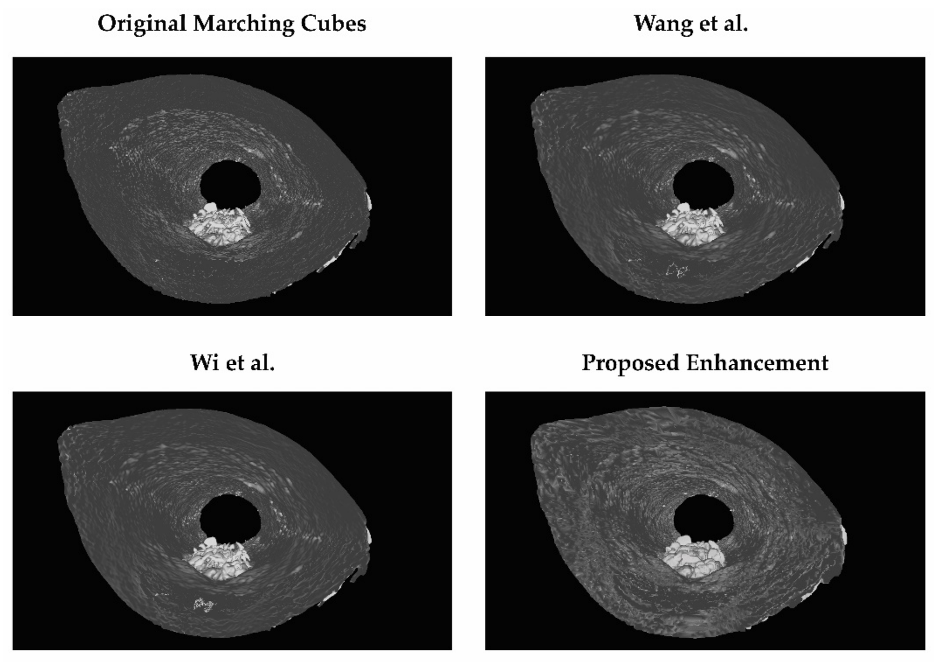

Figure 22. 3D models reconstructed with different reconstruction methods.

Based on Figure 19, the proposed enhancement (Marching Cubes with 3D data smoothing of Gaussian convolution kernel size 5 and mesh simplification with reduction factor 0.1) has higher SSIM and MS-SSIM values than original Marching Cubes, improvement by Wang et al. [13], and improvement by Wi et al. [17], albeit the difference is minimal (less than 1.0%). In Figures 20 and 21, the proposed enhancement has a lower reconstruction time, rendering time, and the number of vertices and faces compared to the original Marching Cubes, improvement by Wang et al. [13], and improvement by Wi et al. [17]. It is to be noted that the replication of methods proposed by Wang et al. and Wi et al. are made in MATLAB, and different evaluation metrics are applied; hence, there may be a difference in terms of implementation, which explains why their implementation has a lower SSIM and MS-SSIM.

4. Discussion

When progressing towards near real-time 3D reconstruction for large, high-resolution medical images, many methods can be employed to speed up the reconstruction process.

Two methods are applied in this paper, one is GPU processing, another is vectorization. GPU processing is a common technique applied to speed up the reconstruction process which heavily relies on the GPU. Parallel processing is one of the many approaches in GPU processing. In MATLAB, parallel looping relies heavily on the number of cores supported by the CPU. The higher the number of CPU cores, the higher the number of workers in the MATLAB parallel pool, hence, the faster the processing is. As for vectorization, this is mainly effective for MATLAB implementations. This results in a great reduction in total execution time. However, the total execution time can be further decreased by exploring other methods, like downsizing or downsampling the medical images before the reconstruction process takes place. CPU with more cores can also be used to increase the number of workers in the MATLAB parallel pool. A stronger GPU with more memory can also be tested on.

Based on Figure 3, two possible explanations can be made on the shape of the lines: One, the increase in the SSIM values is proportional to the increase in the convolution kernel size. This suggests that if the convolution kernel size is increased further, the SSIM value growth will follow an exponential growth; Two, the graph does not exhibit exponential growth, meaning to say that the SSIM values may not increase exponentially as the convolution kernel size increases. This is largely due to the lack of results as only three convolution kernel sizes are tested. Additionally, because there is a huge jump in kernel size from 5 to 11, and that is exactly where the signs of exponential growth started showing up as illustrated in the figure, it is difficult to judge the growth rate of SSIM when the convolution kernel size increases. Nevertheless, it is safe to say that the larger the convolution kernel size, regardless of whether it is a box or Gaussian, the higher the SSIM values, which means the higher the reconstruction accuracy. A possible explanation for this is that as the convolution kernel size increases, more areas are covered by the convolution kernel during the smoothing step, hence, having a higher tendency to remove noises in the 3D volumetric data.

Figure 5, however, is a different story from Figure 3, whereby the lines showed signs of logarithmic growth. Even so, not all reduction factors exhibit logarithmic signs. Again, it is difficult to judge the growth of MS-SSIM with just three convolution kernel sizes. But it is safe to say that the larger the convolution kernel size, the higher the MS-SSIM values are. One possible explanation for this is the same as the explanation for Figure 3.

As for the lines in Figures 4 and 6, it is clear that the increase in the reduction factor does not equate to an increase in the SSIM and MS-SSIM values. This means the reduction factor for the mesh simplification step has a near-to-no influence on the reconstruction accuracy. It is safe to say that this statement stays true for all convolution kernel sizes as only the convolution kernel size 3 showed a slight variation in values whereas the convolution kernel sizes larger than that remained constant across all nine reduction factors. A possible explanation for this is that mesh simplification methods work in such a way that they preserve the shape and boundary of the 3D models as much as possible whilst reducing the vertices and faces, hence, resulting in nearly the same 3D model shape. Moreover, as the comparison is made between two 2D images, it does not compare well for simplified mesh models.

For Figures 7 and 9, the same logic and explanation can be applied here from Figure 3: One, it is difficult to tell whether the growth is exponential or otherwise as only three convolution kernel box sizes for both box and Gaussian are tested; Two, it is safe to say that the larger the convolution kernel size for both box and Gaussian, the longer the reconstruction time. One possible explanation for this is that more calculations are involved when performing convolution over the 3D volumetric data using a larger convolution kernel size, therefore, more time is needed to complete the convolution process.

In Figure 8, because the lines do not exhibit a specific pattern in shape, it is difficult to judge the effect of the reduction factor on the reconstruction time. However, the reconstruction time for reduction factors above 0.5 are relatively lower than the ones below 0.5, hence one possible conclusion can be drafted here whereby the time needed to complete the reconstruction process favors higher reduction factors. One possible explanation for this is that as the reduction factor increases, the faster the mesh simplification process ends as it will be able to reach the target number of vertices and faces earlier. However, this explanation only holds true for the convolution kernel box. Referring to Figure 10, the reconstruction times for reduction factors above and below 0.5 showed no signs of being relatively higher or lower than the other half. There is nearly no similarity between all three lines, except for reduction factors 0.5 to 0.6 whereby all three lines dropped, albeit at a different rate. Hence, one possible conclusion that can be drawn from this is that the "right" reduction factor for faster reconstruction time when convolution kernel Gaussian is used is 0.6.

Both graphs in Figures 11 and 13 showed similarity in terms of line shape and pattern whereby the larger the convolution kernel size for both the convolution kernel box and Gaussian, the lower the rendering time for all reduction factors. As the rendering time is closely related to the size of the 3D model, which is equivalent to the number of vertices and faces, one possible explanation on this will need to be explained together with graphs in Figures 15 and 17. In Figures 15 and 17, the same explanation can be applied whereby the larger the convolution kernel size for both box and Gaussian, the lower the number of vertices and faces. The lower the number of vertices and faces, the smaller the 3D model sizes are, which means the faster the rendering process of the 3D models. The reason behind this is that as the convolution kernel size increases, more areas are covered by the convolution kernel, therefore, resulting in more noise being removed during the smoothing step. When more noises are removed from the 3D volumetric data, after the reconstruction process, lesser vertices and faces are needed to represent the 3D models as the removed noises are not reconstructed.

Similarly, both graphs in Figures 12 and 14 showed similarity in terms of line shape and pattern, which is also the same case for graphs in Figures 16 and 18. All four graphs showed that the rendering time and the number of vertices and faces increase as the reduction factor increases for both convolution kernels. This is because as the reduction factor increases, the number of vertices and faces retained increases too, which leads to an increase in the rendering time as the 3D models' sizes increase as well. However, this is only up until the reduction factor of 0.7 whereby the increase plateaued as it is a logarithmic growth. One possible explanation for this is that when the reduction factor decreases from 1.0, in which the number of vertices and faces are not reduced at all, to 0.9, 0.8, and then to 0.7, the further decrease in the number of vertices and faces do not conform to the criterion or the error metric that preserves the shape and boundary of the 3D model; thus, the number of vertices and faces remained the same from the reduction factors of 0.7 to 0.9. When the number of vertices and faces remained constant, the size of the 3D models remained constant. When the size remained constant, technically the rendering time for those 3D models should remain constant as well. In this case, there are slight variations in the rendering time across the reduction factors of 0.7 to 0.9. This is because the recorded rendering time changes per run, even if the tabulated rendering time is an average over five runs, if one of the runs is recorded to have a slightly higher time than the usual recorded time range, it will easily affect the averaged time.

In previous discussions, it is mentioned that the SSIM and MS-SSIM values, and the number of vertices and faces, are the same even though different convolution kernels are used across the same convolution kernel size. This remains true in both quantitative analysis and qualitative analysis. There is no difference between either 3D models when observed with human eyes. When the list of vertices and faces are observed for both 3D models, they are of the same values. This means that in the context of 3D data smoothing, specifically in MATLAB's implementation, a different convolution kernel does not have any effect on the reconstruction accuracy, but it affects the reconstruction time.

For future studies, several things can be considered:

1. Image downsampling or downsizing before reconstruction;
2. Stronger GPU and CPU with more cores;

3. Experimenting with more convolution kernel sizes;
4. Different mesh simplification approaches;
5. Testing with more medical image datasets that can be processed at the same time;
6. Implementing with languages like C++;
7. Different metrics to evaluate the reconstruction accuracy;
8. Different code optimization approaches;
9. Support different GPU products like GPU from AMD.

5. Conclusions

In conclusion, Marching Cubes have higher reconstruction accuracy than Marching Tetrahedra for large bone defect medical images. The proposed improvement, which involves 3D data smoothing with convolution kernel Gaussian size 5 and mesh simplification with a reduction factor of 0.1, is the most optimal parameter value combination in achieving a balance between high reconstruction accuracy, low reconstruction and rendering time, and a low number of vertices and faces. With help from GPU, parallel processing, and vectorization, the proposed enhancement has the potential in the near real-time reconstruction of large bone defect medical images.

Based on the obtained results, it can also be concluded that:

- The larger the convolution kernel size, the higher the reconstruction accuracy;
- The reduction factor does not affect the reconstruction accuracy;
- The larger the convolution kernel size, the higher the reconstruction time;
- The reduction factor has an effect on the reconstruction time but no specific growth pattern can be deduced from the graphs; thus, the effect is random;
- The larger the convolution kernel size, the lower the rendering time;
- The higher the reduction factor, the higher the rendering time, up until the reduction factor of 0.7 where it stopped increasing;
- The larger the convolution kernel size, the lower the number of vertices and faces;
- The higher the reduction factor, the higher the number of vertices and faces, up until the reduction factor of 0.7 where it stopped increasing;
- Different convolution kernels do not affect the result of the reconstruction qualitatively and quantitatively, except for the reconstruction time.

Author Contributions: Writing—original draft preparation, D.J.Y.C.; writing—review and editing, A.S.A.M., K.A.S., M.N.A.W. and K.I.; visualization, D.J.Y.C.; supervision, A.S.A.M. and K.A.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Higher Education Malaysia for Fundamental Research Grant Scheme with Project Code: FRGS/1/2020/STG07/USM/02/12.

Institutional Review Board Statement: The study was conducted according to the guidelines of the Declaration of Helsinki and approved by the Ethics Committee of ANIMAL ETHICS COMMITTEE USM, USM/Animal Ethics Approval/2016/(104)(814), 22 December 2016.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author. The data are not publicly available due to research confidentiality.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. SSIM, MS-SSIM, reconstruction time, rendering time, and number of vertices and faces for Marching Cubes and proposed improved Marching Cubes with different parameter combinations.

Reconstruction Method	SSIM (%)	MS-SSIM (%)	Reconstruction Time (s)	Rendering Time (s)	Vertices	Faces
MC	87.68	82.32	373.62	20.87	2,550,055	5,108,402
MC Box S 3 RF 0.1	87.67	82.32	117.14	2.05	369,786	740,229
MC Box S 5 RF 0.1	87.69	82.35	120.43	1.64	342,443	683,804
MC Box S 11 RF 0.1	87.77	82.48	159.81	1.52	315,490	629,003
MC Gaussian S 3 RF 0.1	87.67	82.32	115.37	1.82	369,786	740,229
MC Gaussian S 5 RF 0.1	87.69	82.35	117.45	1.66	342,443	683,804
MC Gaussian S 11 RF 0.1	87.77	82.48	146.73	1.53	315,490	629,003
MC Box S 3 RF 0.2	87.66	82.31	114.69	3.76	738,421	1,480,457
MC Box S 5 RF 0.2	87.69	82.34	114.72	3.34	683,832	1,367,611
MC Box S 11 RF 0.2	87.77	82.47	144.22	3.07	630,077	1,258,006
MC Gaussian S 3 RF 0.2	87.66	82.31	115.91	3.68	738,421	1,480,457
MC Gaussian S 5 RF 0.2	87.69	82.34	125.07	3.45	683,832	1,367,611
MC Gaussian S 11 RF 0.2	87.77	82.47	149.84	3.06	630,077	1,258,006
MC Box S 3 RF 0.3	87.66	82.31	114.37	5.63	1,108,621	2,220,687
MC Gaussian S 5 RF 0.3	87.69	82.33	127.39	5.05	1,025,215	2,051,416
MC Gaussian S 11 RF 0.3	87.77	82.47	147.25	4.58	944,575	1,887,008
MC Box S 3 RF 0.4	87.66	82.31	111.94	7.56	1,478,169	2,960,916
MC Box S 5 RF 0.4	87.69	82.34	116.59	6.96	1,366,705	2,735,223
MC Box S 11 RF 0.4	87.77	82.47	150.07	6.2	1,259,023	2,516,012
MC Gaussian S 3 RF 0.4	87.66	82.31	124.41	7.39	1,478,169	2,960,916
MC Gaussian S 5 RF 0.4	87.69	82.34	121.19	6.83	1,366,705	2,735,223
MC Gaussian S 11 RF 0.4	87.77	82.47	147.41	6.11	1,259,023	2,516,012
MC Box S 3 RF 0.5	87.67	81.31	109.07	9.29	1,847,830	3,701,144
MC Box S 5 RF 0.5	87.69	82.34	118.32	8.36	1,708,398	3,419,029
MC Box S 11 RF 0.5	87.77	82.47	148.62	7.63	1,573,546	3,145,015
MC Gaussian S 3 RF 0.5	87.67	81.31	117.18	9.28	1,847,830	3,701,144
MC Gaussian S 5 RF 0.5	87.69	82.34	122.75	8.54	1,708,398	3,419,029
MC Gaussian S 11 RF 0.5	87.77	82.47	149.38	7.67	1,573,546	3,145,015
MC Box S 3 RF 0.6	87.67	81.31	99.41	11.13	2,217,200	4,441,374
MC Box S 5 RF 0.6	87.69	82.34	108.28	10.14	2,049,797	4,102,833
MC Box S 11 RF 0.6	87.77	82.47	128.75	9.29	1,887,987	3,774,019
MC Gaussian S 3 RF 0.6	87.67	81.31	111.83	10.98	2,217,200	4,441,374
MC Gaussian S 5 RF 0.6	87.69	82.34	116.61	9.95	2,049,797	4,102,833
MC Gaussian S 11 RF 0.6	87.77	82.47	136.32	9.21	1,887,987	3,774,019
MC Box S 3 RF 0.7	87.67	81.31	108.81	11.94	2,394,652	4,798,599
MC Box S 5 RF 0.7	87.69	82.34	120.53	11.08	2,275,927	4,556,820
MC Box S 11 RF 0.7	87.77	82.47	126.8	10.32	2,154,966	4,308,705
MC Gaussian S 3 RF 0.7	87.67	81.31	115.68	11.78	2,394,652	4,798,599
MC Gaussian S 5 RF 0.7	87.69	82.34	116.06	11.04	2,275,927	4,556,820
MC Gaussian S 11 RF 0.7	87.77	82.47	138.83	10.5	2,154,966	4,308,705
MC Box S 3 RF 0.8	87.67	81.31	99.71	11.8	2,394,652	4,798,599
MC Box S 5 RF 0.8	87.69	82.34	106.33	11.11	2,275,927	4,556,820
MC Box S 11 RF 0.8	87.77	82.47	128.9	10.57	2,154,966	4,308,705
MC Gaussian S 3 RF 0.8	87.67	81.31	111.12	11.99	2,394,652	4,798,599
MC Gaussian S 5 RF 0.8	87.69	82.34	128.52	11.11	2,275,927	4,556,820
MC Gaussian S 11 RF 0.8	87.77	82.47	148.97	10.47	2,154,966	4,308,705
MC Box S 3 RF 0.9	87.67	81.31	99.59	11.97	2,394,652	4,798,599
MC Box S 5 RF 0.9	87.69	82.34	107.82	11.01	2,275,927	4,556,820
MC Box S 11 RF 0.9	87.77	82.47	131.88	10.41	2,154,966	4,308,705
MC Gaussian S 3 RF 0.9	87.67	81.31	116.47	12.08	2,394,652	4,798,599
MC Gaussian S 5 RF 0.9	87.69	82.34	120.75	11.25	2,275,927	4,556,820
MC Gaussian S 11 RF 0.9	87.77	82.47	148.69	10.38	2,154,966	4,308,705

Appendix B

Table A2. Parameter value combination scores.

Reduction Factor	Box Size 3	Box Size 5	Box Size 11	Gaussian Size 3	Gaussian Size 5	Gaussian Size 11
0.1	−0.0051	0.2939	0.2659	−0.0036	0.2962	0.2758
0.2	−0.3190	0.2835	0.2642	−0.3199	0.2756	0.2599
0.3	−0.3347	0.2624	0.2482	−0.3434	0.2592	0.2484
0.4	−0.3488	0.2526	0.2327	−0.3582	0.2492	0.2348
0.5	−0.3625	0.2368	0.2204	−0.3686	0.2333	0.2198
0.6	−0.3710	0.2297	0.2219	−0.3803	0.2235	0.2162
0.7	−0.3857	0.2108	0.2122	−0.3909	0.2142	0.2029
0.8	−0.3787	0.2216	0.2104	−0.3875	0.2047	0.1952
0.9	−0.3788	0.2205	0.2082	−0.3917	0.2105	0.1955

References

1. Alasal, S.A.; Alsmirat, M.; Al-Mnayyis, A.; Baker, Q.B.; Al-Ayyoub, M. Improving radiologists' and orthopedists' QoE in diagnosing lumbar disk herniation using 3D modeling. *Int. J. Electr. Comput. Eng.* **2021**, *11*, 4336–4344. [CrossRef]
2. Al-Mnayyis, A.; Alasal, S.A.; Alsmirat, M.; Baker, Q.B.; AlZu'bi, S. Lumbar disk 3D modeling from limited number of MRI axial slices. *Int. J. Electr. Comput. Eng.* **2020**, *10*, 4101–4108. [CrossRef]
3. Stein, D.; Assaf, Y.; Dar, G.; Cohen, H.; Slon, V.; Kedar, E.; Medlej, B.; Abbas, J.; Hay, O.; Barazany, D.; et al. 3D virtual reconstruction and quantitative assessment of the human intervertebral disc's annulus fibrosus: A DTI tractography study. *Sci. Rep.* **2021**, *11*, 6815. [CrossRef] [PubMed]
4. Bao, L.; Rong, S.; Shi, Z.; Wang, J.; Zhang, Y. Measurement of femoral posterior condylar offset and posterior tibial slope in normal knees based on 3D reconstruction. *BMC Musculoskelet. Disord.* **2021**, *22*, 486. [CrossRef] [PubMed]
5. Tuecking, L.-R.; Ettinger, M.; Nebel, D.; Welke, B.; Schwarze, M.; Windhagen, H.; Savov, P. 3D-surface scan based validated new measurement technique of femoral joint line reconstruction in total knee arthroplasty. *J. Exp. Orthop.* **2021**, *8*, 16. [CrossRef]
6. Wu, W.; Wu, Y.; Shen, G.; Zhang, G. Preoperative virtual simulation for synchronous multiple primary lung cancers using three-dimensional computed tomography lung reconstruction: A case report. *J. Cardiothorac. Surg.* **2021**, *16*, 10. [CrossRef]
7. Bosc, R.; Tortolano, L.; Hersant, B.; Oudjhani, M.; Leplay, C.; Woerther, P.L.; Aguilar, P.; Leguen, R.; Meningaud, J.-P. Bacteriological and mechanical impact of the Sterrad sterilization method on personalized 3D printed guides for mandibular reconstruction. *Sci. Rep.* **2021**, *11*, 581. [CrossRef]
8. Wang, S.; Leng, H.; Tian, Y.; Xu, N.; Liu, Z. A novel 3D-printed locking cage for anterior atlantoaxial fixation and fusion: Case report and in vitro biomechanical evaluation. *BMC Musculoskelet. Disord.* **2021**, *22*, 121. [CrossRef] [PubMed]
9. van Ooijen, P.M.A.; van Geuns, R.J.M.; Rensing, B.J.W.M.; Bongaerts, A.H.H.; de Feyter, P.J.; Oudkerk, M. Noninvasive coronary imaging using electron beam CT: Surface rendering versus volume rendering. *AJR Am. J. Roentgenol.* **2003**, *180*, 223–226. [CrossRef]
10. Udupa, J.K.; Hung, H.-M.; Chuang, K.-S. Surface and volume rendering in three-dimensional imaging: A comparison. *J. Digit. Imaging* **1991**, *4*, 159. [CrossRef]
11. Lorensen, W.E.; Cline, H.E. Marching cubes: A high resolution 3D surface construction algorithm. *Comput. Graph.* **1987**, *21*, 163–169. [CrossRef]
12. Kim, S.; Sohn, D.; Im, S. Construction of polyhedral finite element meshes based upon marching cube algorithm. *Adv. Eng. Softw.* **2019**, *128*, 98–112. [CrossRef]
13. Wang, J.; Huang, Z.; Yang, X.; Jia, W.; Zhou, T. Three-dimensional Reconstruction of Jaw and Dentition CBCT Images Based on Improved Marching Cubes Algorithm. *Procedia CIRP* **2020**, *89*, 239–244. [CrossRef]
14. Garland, M.; Heckbert, P.S. Surface simplification using quadric error metrics. In Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'97), Anaheim, LA, USA, 3–8 August 1997; ACM Press/Addison-Wesley Publishing Co.: New York, NY, USA, 1997; pp. 209–216.
15. Chernyaev, E.V. Marching Cubes 33: Construction of Topologically Correct Isosurfaces. In Proceedings of the GRAPHICON'95, St. Petersburg, Russia, 3–7 July 1995.
16. Custodio, L.; Pesco, S.; Silva, C. An extended triangulation to the Marching Cubes 33 algorithm. *J. Braz. Comput. Soc.* **2019**, *25*, 6. [CrossRef]
17. Wi, W.; Park, S.M.; Shin, B.S. Computed Tomography-Based Preoperative Simulation System for Pedicle Screw Fixation in Spinal Surgery. *J. Korean Med. Sci.* **2020**, *35*, e125. [CrossRef] [PubMed]
18. Masala, G.L.; Golosio, B.; Oliva, P. An improved Marching Cube algorithm for 3D data segmentation. *Comput. Phys. Commun.* **2013**, *184*, 777–782. [CrossRef]
19. Wang, M.; Luo, H.; Cui, Q. Three-Dimensional Reconstruction Based On Improved Marching Cubes Algorithm. *J. Mech. Med. Biol.* **2020**, *20*, 2040002. [CrossRef]

20. Doi, A.; Koide, A. An Efficient Method of Triangulating Equi-Valued Surfaces by Using Tetrahedral Cells. *IEICE Trans. Inf. Syst.* **1991**, *E74-D*, 214–224.
21. Lu, T.; Chen, F. Quantitative analysis of molecular surface based on improved Marching Tetrahedra algorithm. *J. Mol. Graph.* **2012**, *38*, 314–323. [CrossRef] [PubMed]
22. Bagley, B.; Sastry, S.P.; Whitaker, R.T. A Marching-tetrahedra Algorithm for Feature-preserving Meshing of Piecewise-smooth Implicit Surfaces. *Procedia Eng.* **2016**, *163*, 162–174. [CrossRef]
23. Guo, D.; Li, C.; Wu, L.; Yang, J. Improved marching tetrahedra algorithm based on hierarchical signed distance field and multi-scale depth map fusion for 3D reconstruction. *J. Vis. Commun. Image Represent.* **2017**, *48*, 491–501. [CrossRef]
24. Ren, P.; Wang, H.; Zhou, G.; Li, J.; Cai, Q.; Yu, J.; Yuan, Y. Solid rocket motor propellant grain burnback simulation based on fast minimum distance function calculation and improved marching tetrahedron method. *Chin. J. Aeronaut.* **2021**, *34*, 208–224. [CrossRef]
25. Marching Cubes–File Exchange–MATLAB Central. Available online: https://www.mathworks.com/matlabcentral/fileexchange/32506-marching-cubes (accessed on 27 September 2021).
26. stlwrite–Write ASCII or Binary STL Files–File Exchange–MATLAB Central. Available online: https://www.mathworks.com/matlabcentral/fileexchange/20922-stlwrite-write-ascii-or-binary-stl-files (accessed on 30 September 2021).
27. Polygonising a Scalar Field (Marching Cubes). Available online: http://paulbourke.net/geometry/polygonise/ (accessed on 29 September 2021).
28. Wang, Z.; Simoncelli, E.P.; Bovik, A.C. Multiscale structural similarity for image quality assessment. In Proceedings of the Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, Pacific Grove, CA, USA, 9–12 November 2003; IEEE: Piscataway, NJ, USA; pp. 1398–1402.
29. Smooth 3-D Data—MATLAB Smooth3. Available online: https://www.mathworks.com/help/matlab/ref/smooth3.html (accessed on 1 October 2021).
30. Reduce Number of Patch Faces—MATLAB Reducepatch. Available online: https://www.mathworks.com/help/matlab/ref/reducepatch.html (accessed on 1 October 2021).
31. Array Stored on GPU—MATLAB. Available online: https://www.mathworks.com/help/parallel-computing/gpuarray.html (accessed on 4 October 2021).
32. Execute For-Loop Iterations in Parallel on Workers—MATLAB Parfor. Available online: https://www.mathworks.com/help/parallel-computing/parfor.html (accessed on 5 October 2021).
33. Vectorization–MATLAB & Simulink. Available online: https://www.mathworks.com/help/matlab/matlab_prog/vectorization.html (accessed on 10 October 2021).