 doi:10.20944/preprints202110.0362.v1

Article

Effects of Different Parameter Settings for 3D Data Smoothing and Mesh Simplification on Towards 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, Penang, 11800, Malaysia
[2] School of Materials and Mineral Resources Engineering, Universiti Sains Malaysia Engineering Campus, Penang, 14300, Malaysia
[3] Adjunct Professor, 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 812-8582, Fukuoka, Japan
* Correspondence: sufril@usm.my[1], biokhairul@usm.my[2]

Abstract: Three-dimensional reconstruction plays an important role in assisting doctors and surgeons in diagnosing bone defects' healing progress. Common three-dimensional reconstruction methods include surface and volume rendering. As the focus is on the shape of the bone, volume rendering is omitted. Many improvements have been made on 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, in this study, an attempt towards near real-time surface rendering for large medical images is made. 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 0.5 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 the 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 in treating bone voids caused by various reasons. Traditionally, bone defects implanted with a bone void filler are examined by scanning it through computed tomography (CT) scanner, which results in a stack of two dimensional (2D) CT images. However, it is difficult to judge the bone defect' healing progress as there are noises and artefacts present in the 2D CT bone defect images, which obscures the details of the bone defect. Also, it is difficult to visualize the bone defect as a whole with the 2D CT image stack. This can be improved by visualizing the 2D CT image stack as a three-dimensional (3D) view, which according to [1,2], 3D models can help doctors increase the quality of experience and diagnosis accuracy. The process of visualizing the 2D CT image stack as a 3D view is called 3D reconstruction.

 doi:10.20944/preprints202110.0362.v1

With the advancement of 3D technologies, 3D reconstruction is seen applied in many areas that require handling tasks using computer vision. As this study emphasizes 3D reconstruction using micro-CT images which are medical images, 3D reconstruction in the medical imaging area is studied. In [3], the annulus fibrosus (AF) in intervertebral disc (IVD) is reconstructed, visualized, and tracked from diffusion tensor imaging (DTI) images. 3D models generated through 3D reconstruction are also used for measurement purposes in which the measurements are useful in surgical planning [4,5]. Measurements obtained from the reconstructed 3D models are also beneficial in virtual simulation for preoperative plans and personalized surgery [6–8]. The goal is to compare 3D reconstruction algorithms which are generally grouped into two categories: surface rendering and volume rendering [9,10].

Surface rendering and volume rendering are common methods applied in visualizing medical images in their equivalent 3D form. 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. Vice versa, if the main focus of the visualization is on the internal structures, then volume rendering is more suitable than surface rendering. Both areas are widely researched in recent years, but many of the improvements focused on increasing the reconstruction accuracy and very few on real-time or near real-time surface rendering of large 3D bone defect models. Many of the improved methods which result in reduced reconstruction time 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 improvement methods towards the reconstruction accuracy, reconstruction time, and optimization of vertices and faces. As the main focus is on visualizing the surface of the bone defect, the three main objectives in this study are one, to compare different surface rendering techniques in terms of reconstruction accuracy; two, to optimize the better surface rendering technique towards near real-time rendering and higher reconstruction accuracy by experimenting on different parameter value combinations; three, to study the effects of the improvements on the reconstruction accuracy, reconstruction time, rendering time, and the number of vertices and faces.

Surface rendering is a 3D reconstruction method that focuses on the deposition of isosurface comprising of triangular patches on the segmented region of interest (ROI) per medical image slice. This results in a 3D model made up of isosurface stacks. Common surface rendering techniques discussed in recent years are the Marching Cubes algorithm and the Marching Tetrahedra algorithm. The Marching Cubes algorithm is a generic surface rendering method applied in the medical field, mainly because of its simplicity in terms of implementation and has relatively fast reconstruction speed. The reconstruction speed is subjective to the size of the medical images being reconstructed. Marching Cubes uses a cube as a unit when forming the isosurface, and the cube configurations are based on the 15 unique patterns as identified in the original Marching Cubes algorithm [11]. This causes ambiguity issues during the triangulation step, which leads to the formation of "holes" on isosurfaces.

Despite that, the original Marching Cubes algorithm is still used 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, [13] proposed the application of Laplacian smoothing, edge collapse method using Quadric Error Metric [14], and polygon merging method through polygon normal regression, in the Marching Cubes algorithm. A variation of the Marching Cubes comprising of 33 unique cube configurations, called the Marching Cubes 33, is introduced by [15], which is further extended by [16] by grouping the cube configurations into either simple leaves triangulation, tunnel triangulation, or interior point leaves triangulation. [17] proposed another improved Marching Cubes algorithm that applies Laplacian smoothing, edge collapse method using Quadric Error Metric [14], and Taubin smoothing. [18] proposed an improved Marching Cubes algorithm by extending the original 15 cube

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

configurations to 21 cube configurations. [19] also proposed an improved Marching Cubes algorithm by extending the 15 cube configurations to 24 cube configurations.

Marching Tetrahedra, on the other hand, is a variant of Marching Cubes which uses tetrahedron as a unit instead of a cube for isosurface [20]. One major advantage of Marching Tetrahedra over Marching Cubes is that there is no ambiguity issue. However, a huge 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, [21] used bisection iterations to shorten the time needed to calculate the intersection points and proposed a simpler redundant vertices elimination strategy as a post-reconstruction step. [22] proposed an improved Marching Tetrahedra algorithm by applying a simple bisection technique to discover the edge-cut locations, which are then warped and retetrahedralized. [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. [24] proposed a more complex improved Marching Tetrahedra algorithm that uses the second-order tetrahedral element called C3D10 as a unit during surface reconstruction.

2. Materials and Methods

A high-resolution micro-CT image dataset is collected from Dr. Khairul Anuar Shariff from 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 is sitting at 1400 dots per inch (DPI) for both 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 because it can be reconstructed as a whole without running out of memory.

The overall flowchart is as illustrated in Figure 1. All the implementations are made in matrix laboratory (MATLAB). Before the image stack can be reconstructed into a 3D model, the images need to be pre-processed so that they can be supported by the reconstruction algorithms. Firstly, the ROI is labelled using the Canny edge detector with a threshold value of 0.5. This will result in some of the artifacts being labelled as well. This is followed by thresholding with a threshold value of 0.3, which removes some of the artifacts labelled in the previous step. Lastly, area filtering is applied with only one connected component is retrieved so that the number of artifacts reconstructed can be further reduced. 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.

After the pre-processing step, 3D volumetric data is formed. The 3D volumetric data is then used to reconstruct 3D models for the comparison study. In this case, Marching Cubes and Marching Tetrahedra are chosen in the comparison study to see which method has higher reconstruction accuracy. The Marching Cubes code used in this paper is the implementation by [25]. A 3D model is generated after the reconstruction process, which is then exported out as a Standard Tessellation Language (STL) file. The Marching Tetrahedra code implemented in this paper uses Peter Hammer's Marching Cubes implementation as a codebase, and vertices and faces orientation, and lookup tables changes are made on top of that so that it runs as Marching Tetrahedra instead of as Marching Cubes. The code changes referred to the source code written in C by [26]. As each tetrahedron configuration, after the reconstruction process, is exported as one individual STL file, and that the Marching Tetrahedra replicated in this study is the six tetrahedron variant, hence a total of six STL files are exported. They are then imported into Blender to combine them into one complete model before exporting it out as another single STL file. This results in two 3D models, one from Marching Cubes and another one from Marching Tetrahedra. They are then imported into Meshlab and the 2D images of the 3D models are captured as black and white images. The captured 2D images are then used to calculate the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

reconstruction accuracy with the 2D image of the bone defect provided in the dataset as ground truth.

In this study, the structural similarity index (SSIM) and multiscale structural similarity index (MS-SSIM) are used to evaluate the reconstruction accuracy. Both metrics measure the structural similarity over the 2D images, with the main difference in MS-SSIM measures it over different image scaling, which is performed through different image subsampling processes with low-pass filters [27]. Values of both metrics in MATLAB implementation are already normalized, hence no further action is needed. The higher the SSIM and MS-SSIM values, the higher the reconstruction accuracy. The algorithm with the higher SSIM and MS-SSIM values will be used as the base for improvement, which will be referred to as algorithm *A* for the rest of the content in this chapter.

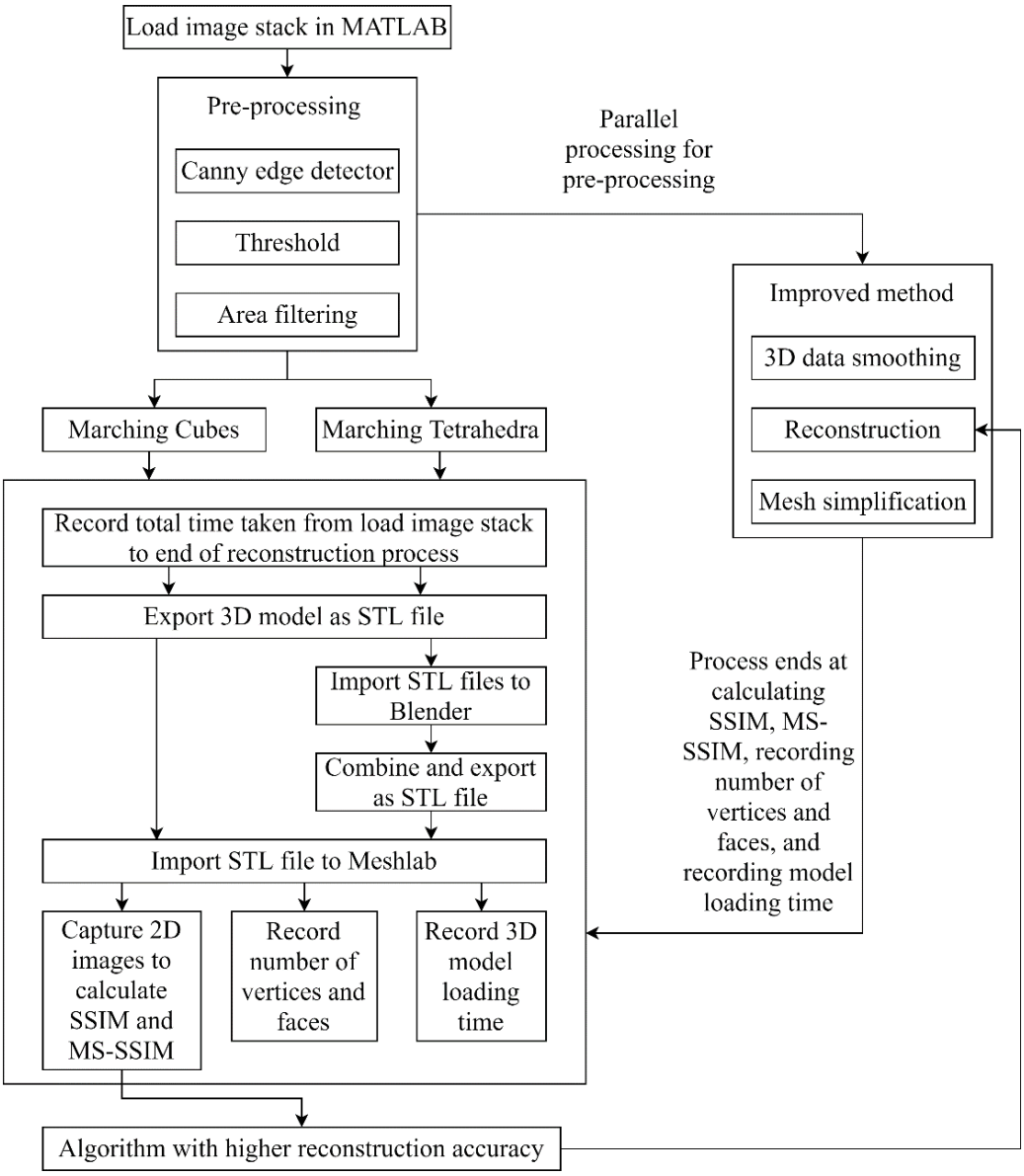

Figure 1. Overall flowchart of methodology.

The improvement comprises applying two additional steps on top of algorithm *A*, which are the 3D data smoothing step and mesh simplification step. 3D data smoothing step is applied to the 3D volumetric data before the reconstruction step by algorithm *A* takes place. What this step does is remove noises in the 3D data which may improve the 3D reconstruction accuracy because noises often lead to wrong triangulations. The 3D data smoothing step applied in this paper is a MATLAB function called *smooth3* [28],

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

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 out to investigate the effect of different convolution kernels and its' sizes on the reconstruction accuracy. Convolution kernel sizes 3 and 5 are common kernel sizes. Another commonly used kernel size is 10, but because *smooth3* only accepts odd-numbered convolution kernel size, hence 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 to be 3D printed, which greatly reduces the slicing time. The mesh simplification step applied in this paper is also a MATLAB function called *reducepatch* [29]. 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, which are 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 out as STL files following the MATLAB implementation by [30] and imported into Meshlab to capture the 2D black and white images. These images will be used in calculating the reconstruction accuracy as mentioned previously. 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 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, which are 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 [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 in parallel over a parallel pool of six workers defined by MATLAB during code execution. Lastly, although there are very minimal code changes made, the reconstruction function used is the vectorized version of the original algorithm *A*. As MATLAB is optimized to perform operations over array-based objects like matrices and vectors [33], any loops that can be vectorized will perform way 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 made as stated above can be evaluated by recording the reconstruction time and rendering time for both original and improved algorithm *A*.

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 taken to export the 3D models as STL files and is recorded in seconds. It is tabulated as average over 5 runs. Rendering time refers to the total time taken to load the 3D model in a 3D software, which in this case refers to the time taken to load the 3D model as STL files in Meshlab. It is also recorded in seconds and tabulated as average over 5 runs.

Besides 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. Graphs are plotted to study the effect of 3D data smoothing and mesh simplification on

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

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

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

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 very small 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 reconstruction accuracy is given the highest emphasis in this comparison study. Marching Cubes will be used as a codebase for improvement.

3.2. Marching Cubes vs Proposed Improved Marching Cubes

Table 2. *Cont.*

Reconstruction Method	SSIM (%)	MS-SSIM (%)	Reconstruction Time (s)	Rendering Time (s)	Vertices	Faces
MC	87.68	82.32	373.62	20.87	2550055	5108402
MC Box S 3 RF 0.1	87.67	82.32	117.14	2.05	369786	740229
MC Box S 5 RF 0.1	87.69	82.35	120.43	1.64	342443	683804
MC Box S 11 RF 0.1	87.77	82.48	159.81	1.52	315490	629003
MC Gaussian S 3 RF 0.1	87.67	82.32	115.37	1.82	369786	740229
MC Gaussian S 5 RF 0.1	87.69	82.35	117.45	1.66	342443	683804
MC Gaussian S 11 RF 0.1	87.77	82.48	146.73	1.53	315490	629003
MC Box S 3 RF 0.2	87.66	82.31	114.69	3.76	738421	1480457
MC Box S 5 RF 0.2	87.69	82.34	114.72	3.34	683832	1367611
MC Box S 11 RF 0.2	87.77	82.47	144.22	3.07	630077	1258006

Table 2. *Cont.*

Reconstruction Method	SSIM (%)	MS-SSIM (%)	Reconstruction Time (s)	Rendering Time (s)	Vertices	Faces

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

MC Gaussian S 3 RF 0.2	87.66	82.31	115.91	3.68	738421	1480457
MC Gaussian S 5 RF 0.2	87.69	82.34	125.07	3.45	683832	1367611
MC Gaussian S 11 RF 0.2	87.77	82.47	149.84	3.06	630077	1258006
MC Box S 3 RF 0.3	87.66	82.31	114.37	5.63	1108621	2220687
MC Gaussian S 5 RF 0.3	87.69	82.33	127.39	5.05	1025215	2051416
MC Gaussian S 11 RF 0.3	87.77	82.47	147.25	4.58	944575	1887008
MC Box S 3 RF 0.4	87.66	82.31	111.94	7.56	1478169	2960916
MC Box S 5 RF 0.4	87.69	82.34	116.59	6.96	1366705	2735223
MC Box S 11 RF 0.4	87.77	82.47	150.07	6.2	1259023	2516012
MC Gaussian S 3 RF 0.4	87.66	82.31	124.41	7.39	1478169	2960916
MC Gaussian S 5 RF 0.4	87.69	82.34	121.19	6.83	1366705	2735223
MC Gaussian S 11 RF 0.4	87.77	82.47	147.41	6.11	1259023	2516012
MC Box S 3 RF 0.5	87.67	81.31	109.07	9.29	1847830	3701144
MC Box S 5 RF 0.5	87.69	82.34	118.32	8.36	1708398	3419029
MC Box S 11 RF 0.5	87.77	82.47	148.62	7.63	1573546	3145015
MC Gaussian S 3 RF 0.5	87.67	81.31	117.18	9.28	1847830	3701144
MC Gaussian S 5 RF 0.5	87.69	82.34	122.75	8.54	1708398	3419029
MC Gaussian S 11 RF 0.5	87.77	82.47	149.38	7.67	1573546	3145015
MC Box S 3 RF 0.6	87.67	81.31	99.41	11.13	2217200	4441374
MC Box S 5 RF 0.6	87.69	82.34	108.28	10.14	2049797	4102833
MC Box S 11 RF 0.6	87.77	82.47	128.75	9.29	1887987	3774019
MC Gaussian S 3 RF 0.6	87.67	81.31	111.83	10.98	2217200	4441374
MC Gaussian S 5 RF 0.6	87.69	82.34	116.61	9.95	2049797	4102833
MC Gaussian S 11 RF 0.6	87.77	82.47	136.32	9.21	1887987	3774019

Table 2. 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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

Reconstruction Method	SSIM (%)	MS-SSIM (%)	Reconstruction Time (s)	Rendering Time (s)	Vertices	Faces
MC Box S 3 RF 0.7	87.67	81.31	108.81	11.94	2394652	4798599
MC Box S 5 RF 0.7	87.69	82.34	120.53	11.08	2275927	4556820
MC Box S 11 RF 0.7	87.77	82.47	126.8	10.32	2154966	4308705
MC Gaussian S 3 RF 0.7	87.67	81.31	115.68	11.78	2394652	4798599
MC Gaussian S 5 RF 0.7	87.69	82.34	116.06	11.04	2275927	4556820
MC Gaussian S 11 RF 0.7	87.77	82.47	138.83	10.5	2154966	4308705
MC Box S 3 RF 0.8	87.67	81.31	99.71	11.8	2394652	4798599
MC Box S 5 RF 0.8	87.69	82.34	106.33	11.11	2275927	4556820
MC Box S 11 RF 0.8	87.77	82.47	128.9	10.57	2154966	4308705
MC Gaussian S 3 RF 0.8	87.67	81.31	111.12	11.99	2394652	4798599
MC Gaussian S 5 RF 0.8	87.69	82.34	128.52	11.11	2275927	4556820
MC Gaussian S 11 RF 0.8	87.77	82.47	148.97	10.47	2154966	4308705
MC Box S 3 RF 0.9	87.67	81.31	99.59	11.97	2394652	4798599
MC Box S 5 RF 0.9	87.69	82.34	107.82	11.01	2275927	4556820
MC Box S 11 RF 0.9	87.77	82.47	131.88	10.41	2154966	4308705
MC Gaussian S 3 RF 0.9	87.67	81.31	116.47	12.08	2394652	4798599
MC Gaussian S 5 RF 0.9	87.69	82.34	120.75	11.25	2275927	4556820
MC Gaussian S 11 RF 0.9	87.77	82.47	148.69	10.38	2154966	4308705

Based on the results tabulated in Table 2, two parameter value combinations are tied for the highest SSIM and MS-SSIM, which are Marching Cubes with convolution kernel box size 11 and reduction factor of 0.1, and Marching Cubes with convolution kernel Gaussian size 11 and reduction factor 0.1, with both having 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 convolution kernel Gaussian size 11 and reduction factor 0.1. It is noticed that 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.

Amongst all the parameter value combinations, the one with the fastest total execution time (reconstruction time added with rendering time) is Marching Cubes with convolution kernel box size 3 and reduction factor 0.6, with the reconstruction time being 99.41 seconds and rendering time being 11.13 seconds, totaling to 110.54 seconds. As for

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

the parameter value combination with the lowest number of vertices and faces, Marching Cubes with convolution kernel box size 11 and reduction factor of 0.1, and Marching Cubes with convolution kernel Gaussian size 11 and reduction factor 0.1 are tied for having 315490 vertices and 629003 faces. Between these two, Marching Cubes with convolution kernel Gaussian size 11 and reduction factor 0.1 has 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 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. The weightage score for the total time taken is 0.6, and the weightage scores for the number of vertices and faces are 0.05 respectively. The total weightage score is 1.0. After multiplying with the respective weightage score, the values are added together to get the final score for every parameter value combination. The scores are tabulated in Table 3.

Table 3. Parameter values 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

Based on the scores in Table 3, Marching Cubes with convolution kernel Gaussian size 5 and reduction factor 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. Besides that, the proposed improvement (Marching Cubes with convolution kernel Gaussian size 5 and reduction factor 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 and can be safely removed without affecting the shape, boundary, and accuracy of the 3D models.

It is noticed that the scores for 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, hence 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 2, it seems that either SSIM or MS-SSIM or both the metrics' values are lower than the original Marching Cubes algorithm, hence resulting in negative values when calculating the percentage of increase in SSIM and MS-SSIM values.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

Besides that, it is noticed that the scores decrease when the reduction factor increases. This is largely due to the increase in total execution time, and the number of vertices and faces as the reduction factor increase. 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 convolution kernel size 5 are higher than the scores for convolution kernel size 11. This is largely due to the drastic increase in total execution time as shown in Table 2.

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

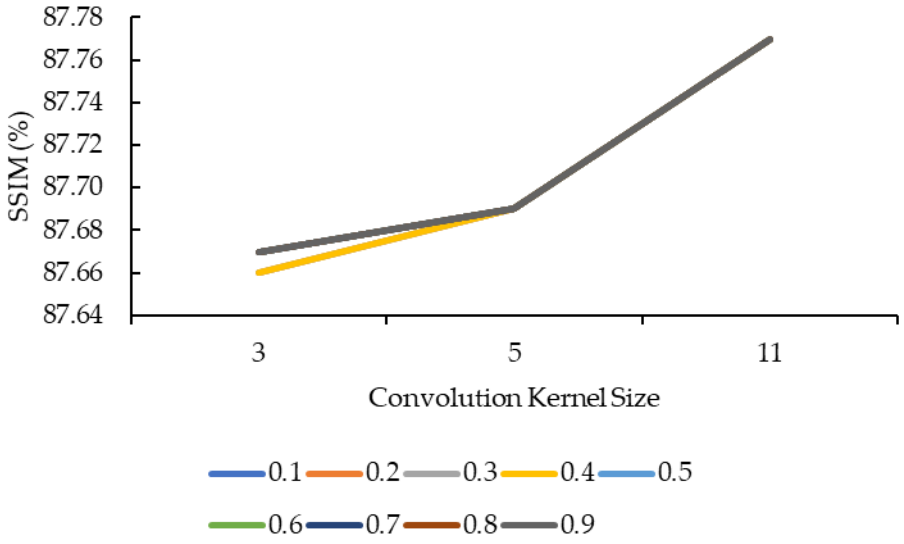

Figure 2. Graph of SSIM against convolution kernel size.

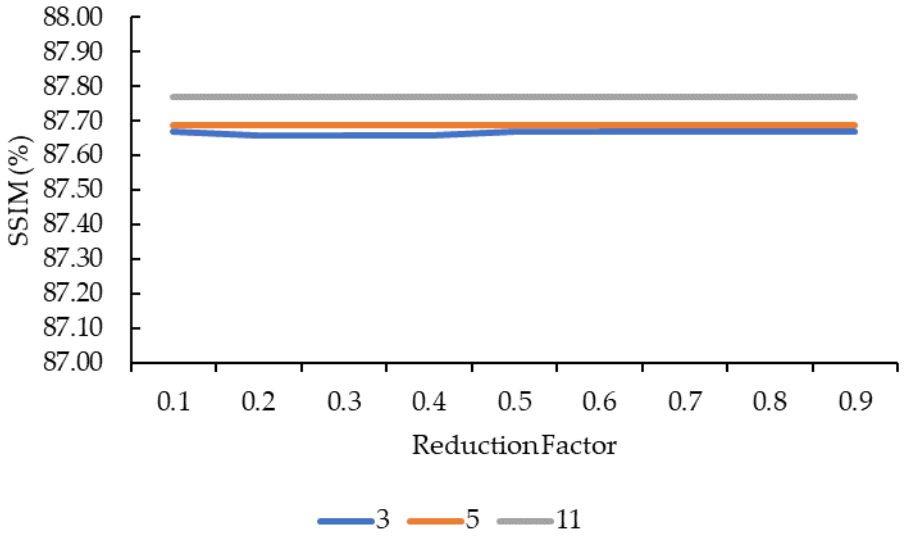

Figure 3. Graph of SSIM against reduction factor.

In Figure 2 and Figure 3, the convolution kernel size and reduction factor for both convolution kernel box and Gaussian are merged into one as both have the same SSIM values. Based on Figure 2, the SSIM values across all reduction factors increased from convolution kernel size 3 to convolution kernel size 5 before shooting up to 87.77% for convolution kernel size 11, except reduction factors 0.2, 0.3, and 0.4 whereby their SSIM values for convolution kernel size 3 are slightly lower than the SSIM values for reduction factors 0.1, 0.5, 0.6, .7, 0.8, and 0.9. In Figure 3, the SSIM values across all three different

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

convolution kernel sizes remained near-constant as the reduction factor increased, except for convolution kernel size 3, which showed a slight variant in SSIM values from the reduction factor 0.1 to 0.5., before remaining constant from 0.6 to 0.9.

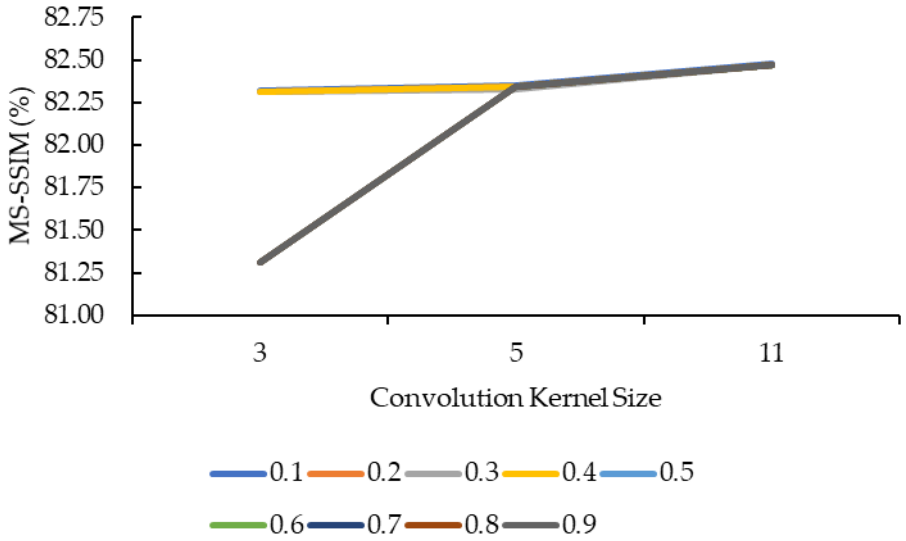

Figure 4. Graph of MS-SSIM against convolution kernel size.

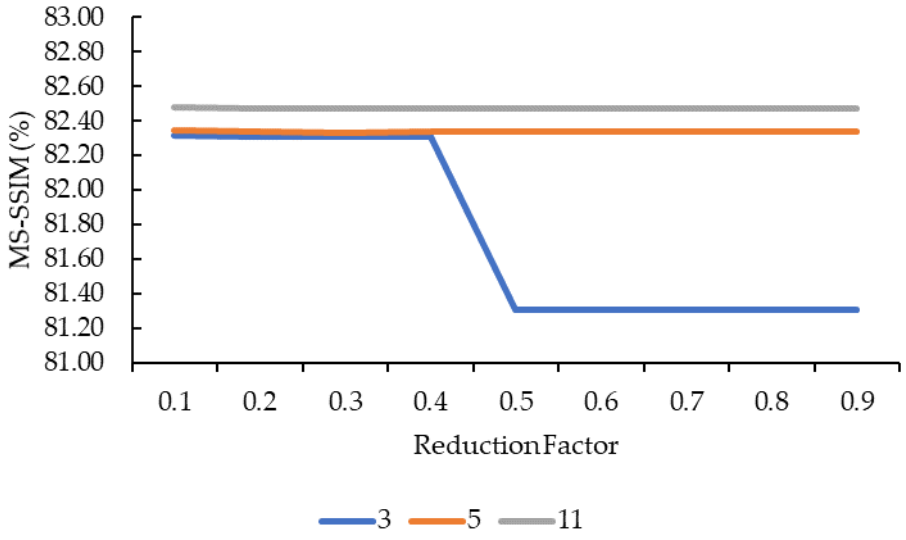

Figure 5. Graph of MS-SSIM against reduction factor.

In Figure 4 and Figure 5, the convolution kernel size and reduction factor for both convolution kernel box and Gaussian are combined into one as both have the same MS-SSIM values. Based on Figure 4, the MS-SSIM values for reduction factors 0.1, 0.2, 0.3, and 0.4 showed a slight increase from convolution kernel size 3 to convolution kernel size 5, before showing another slight increase from convolution kernel size 5 to 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 convolution kernel size 5, before showing the same behavior which is a slight increase from convolution kernel size 5 to convolution kernel size 11. In Figure 5, the MS-SSIM values showed the same behavior as SSIM values for convolution kernel sizes 5 and 11 whereby the values remained near-constant as the reduction factor increased, and that only convolution kernel size 3 showed a big variant in MS-SSIM values when the reduction factor increased from 0.4 to 0.5.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

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

Another four graphs are plotted based on the results obtained in Table 2. The first graph is plotted with reconstruction time against convolution kernel size for the convolution kernel box. The second graph is plotted with reconstruction time against reduction factor also for convolution kernel box. The third and fourth graphs are essentially the same thing as the first and second graphs, but for convolution kernel Gaussian.

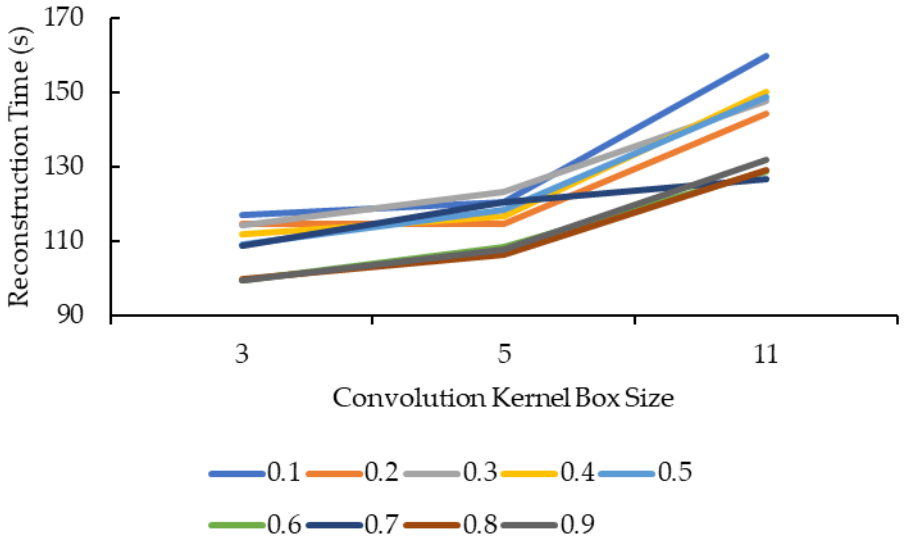

Figure 6. Graph of reconstruction time against convolution kernel size for convolution kernel box.

Referring to Figure 6, it seems that overall, the reconstruction time increases when the convolution kernel box size increases for all reduction factors. All the reduction factors showed a bigger increase in reconstruction when convolution kernel box size increases from 5 to 11 than when convolution kernel box size increases from 3 to 5, except for reduction factor 0.7 whereby it showed a smaller increase in reconstruction time when convolution kernel size increases from 5 to 11 than when convolution kernel size increases from 3 to 5. Based on Figure 7, for convolution kernel box size 3, other than the reconstruction time showing an increase as the reduction factor increases from 0.6 to 0.7, the rest showed a decrease in reconstruction time as the reduction factor increases. For convolution kernel box size 5, the line graph is much more complicated, with the reconstruction time showing an increase when the reduction factor increases 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 reconstruction time. For convolution kernel box size 11, the line graph is showed a consistent increase in reconstruction time when the reduction factor increases from 0.2 to 0.4, and from 0.7 to 0.9, while the rest showed a consistent decrease in reconstruction time. The position of the line graph for convolution kernel box size 11 supports the description for Figure 6 whereby it is clearly above the rest of the line graphs.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

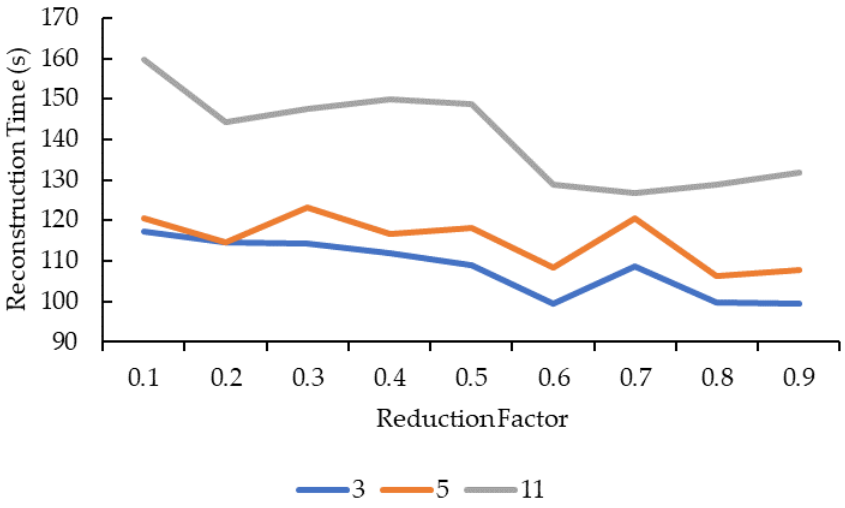

Figure 7. Graph of reconstruction time against reduction factor for convolution kernel box.

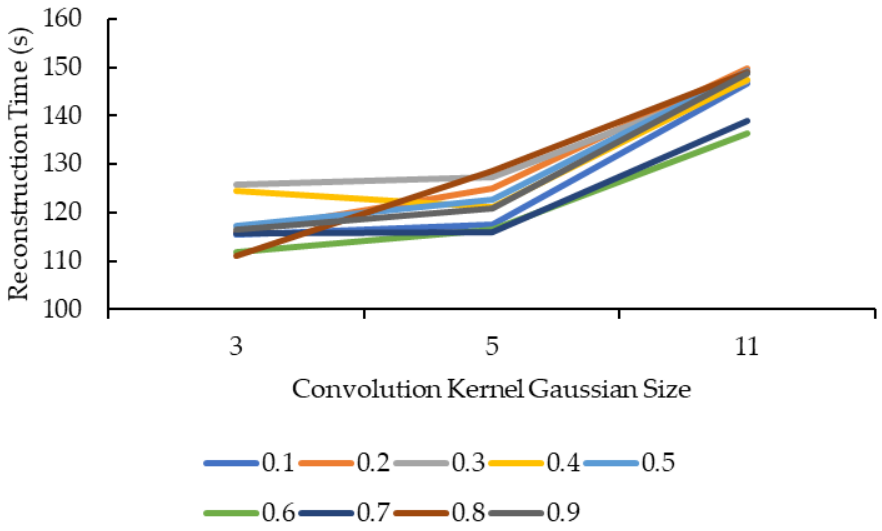

Figure 8. Graph of reconstruction time against convolution kernel size for convolution kernel Gaussian.

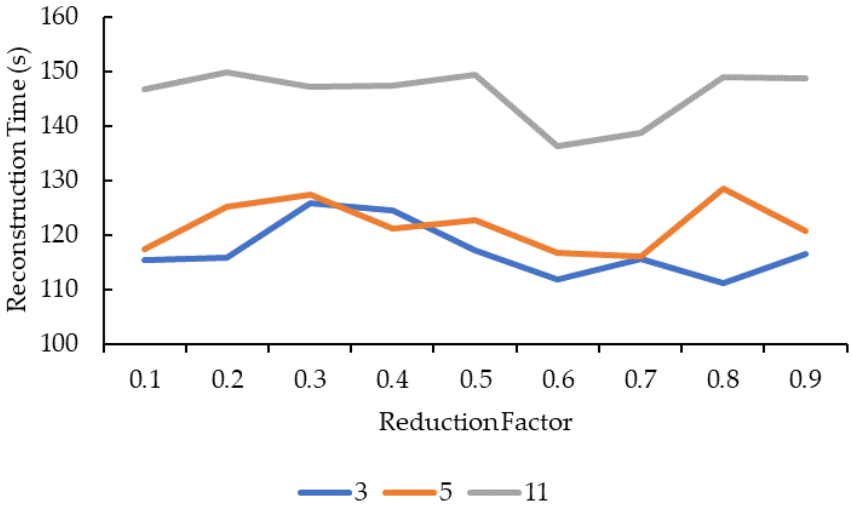

Figure 9. Graph of reconstruction time against reduction factor for convolution kernel Gaussian.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

Based on Figure 8, all the reduction factors showed similar behavior as the convolution kernel box whereby a bigger increase in reconstruction time is seen when convolution kernel size increases from 5 to 11 than when convolution kernel size increases from 3 to 5, except for two reduction factors. Reduction factor 0.4 showed a decrease in reconstruction time when convolution kernel Gaussian size increases from 3 to 5 before showing an increase in reconstruction time when convolution kernel size increases from 5 to 11. Reduction factor 0.8, on the other hand, showed a near-linear increase in reconstruction time as convolution kernel Gaussian size increases. The decrease in reconstruction time for reduction factor 0.4 when convolution kernel Gaussian size increases from 3 to 5 is seen clearly in Figure 9 whereby the line graph for convolution kernel size 3 is above the line graph for convolution kernel size 5 at reduction factor 0.4. Referring to Figure 9, besides the same placement of line graph for convolution kernel Gaussian size 11 as convolution kernel box size 11 in Figure 8, the line graphs in Figure 9 are also complex. For convolution kernel Gaussian size 3, the reconstruction time increases 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 reconstruction time. For convolution kernel Gaussian size 5, the reconstruction time increases 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 reconstruction time. For convolution kernel Gaussian size 11, on the other hand, showed an increase in reconstruction time when the reduction factor increases 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 reconstruction time.

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

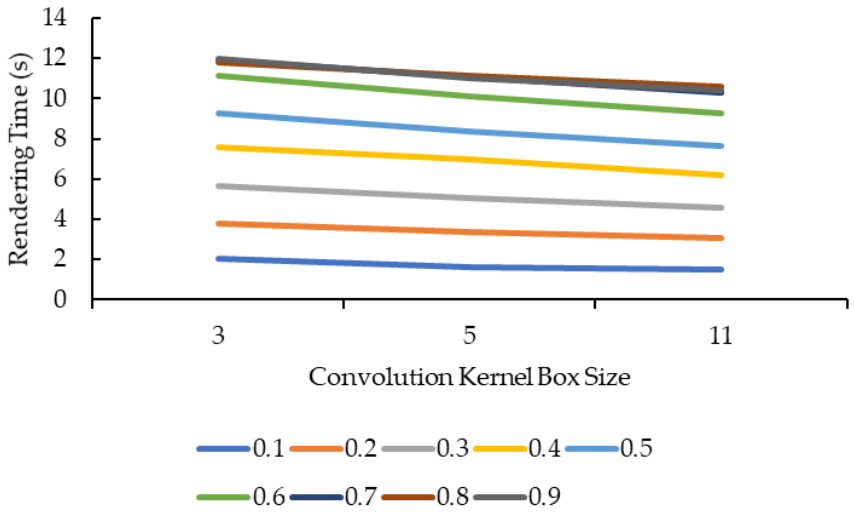

Figure 10. Graph of rendering time against convolution kernel size for convolution kernel box.

Different from Figure 6 and Figure 7, Figure 10 and Figure 11 showed more straightforward lines. In Figure 10, all the reduction factors showed a decrease in rendering time as convolution kernel box size increases. It is noticed that the lines' placement in the graph is higher as the reduction factor increases, which can be seen more clearly in Figure 11. In Figure 11, for all convolution kernel box sizes, the rendering time increases logarithmically as the reduction factor increases. The lines showed signs of plateauing at reduction factor 0.6 before starting to plateau at reduction factor 0.7 onwards, except for convolution kernel box size 3 where it showed a slight variant in rendering time from reduction factors 0.7 to 0.9. Also, there is a slight drop in rendering time for convolution kernel box sizes 5 and 11 when the reduction factor increases from 0.8 to 0.9.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

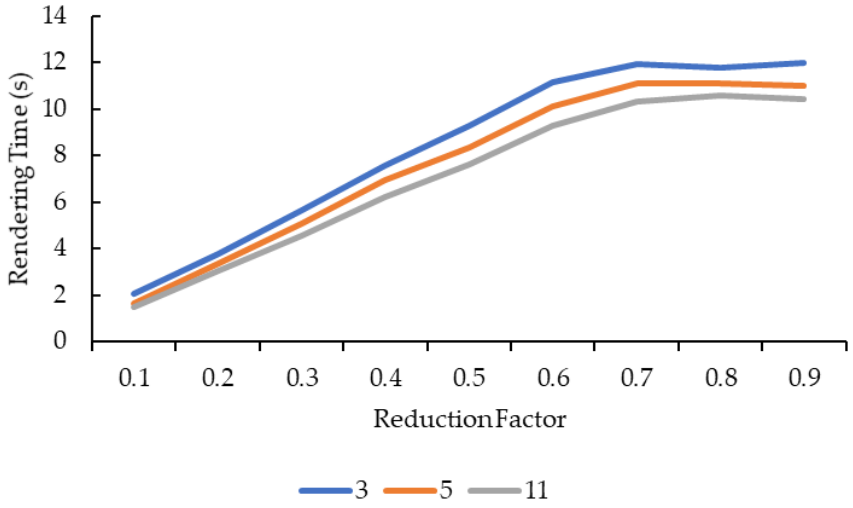

Figure 11. Graph of rendering time against reduction factor for convolution kernel box.

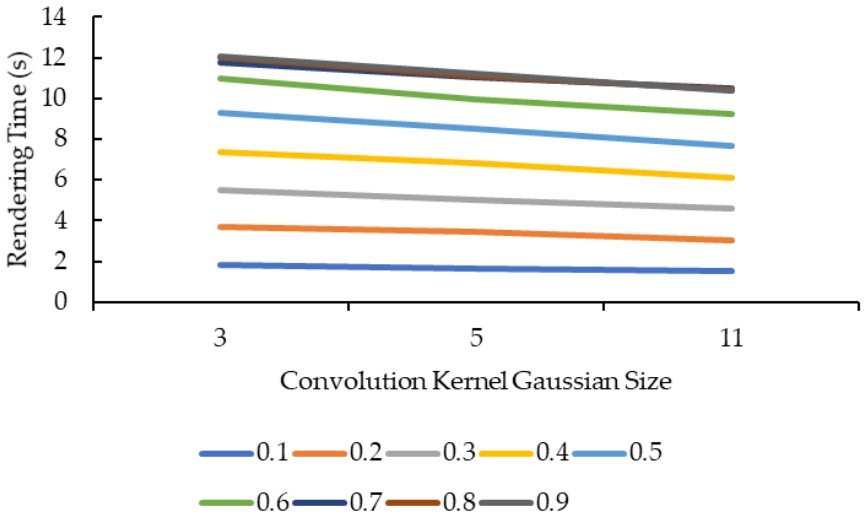

Figure 12. Graph of rendering time against convolution kernel size for convolution kernel Gaussian.

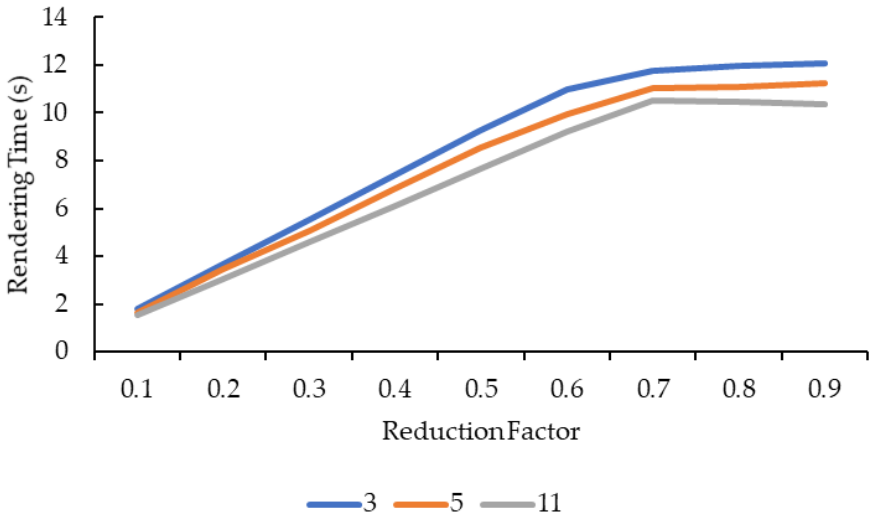

Figure 13. Graph of rendering time against reduction factor for convolution kernel Gaussian.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

In Figure 12 and Figure 13, they are showing nearly the same line shapes like the ones in Figure 10 and Figure 11 respectively. In Figure 12, for all the reduction factors, the rendering time decreases as convolution kernel Gaussian size increases. Similarly, the placement of the lines in the graph showed that rendering time increases as the reduction factor increases, which can be clearly seen in Figure 13. Figure 13 also showed a logarithmic increase in rendering time as the reduction factor increases for all convolution kernel Gaussian sizes. However, for convolution kernel Gaussian, only kernel size 11 showed a slight decrease in rendering time when the reduction factor increases from 0.8 to 0.9.

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

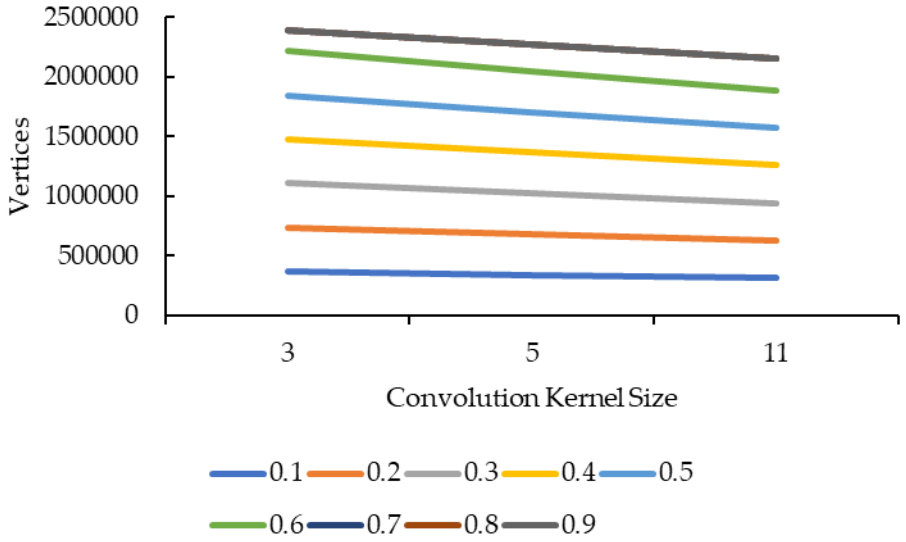

Figure 14. Graph of number of vertices against convolution kernel size.

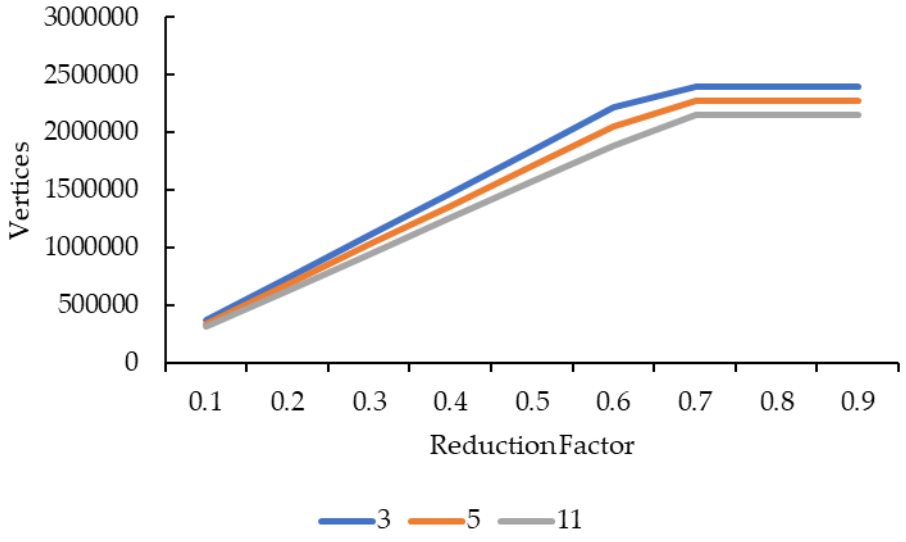

Figure 15. Graph of number of vertices against reduction factor.

Figure 14 and Figure 15 showed very similar line shapes like the ones in Figure 10 and Figure 12, and in Figure 11 and Figure 13 respectively. Based on Figure 14, all the reduction factors showed a decrease in the number of vertices as 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 normal as more vertices and retained

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

when the reduction factor increases. In Figure 15, the increase in the number of vertices as the reduction factor increases can be seen clearly, in which the increase is logarithmic. Reduction factor 0.6 is where the lines showed signs of plateauing and completely plateaued from reduction factor 0.7 onwards.

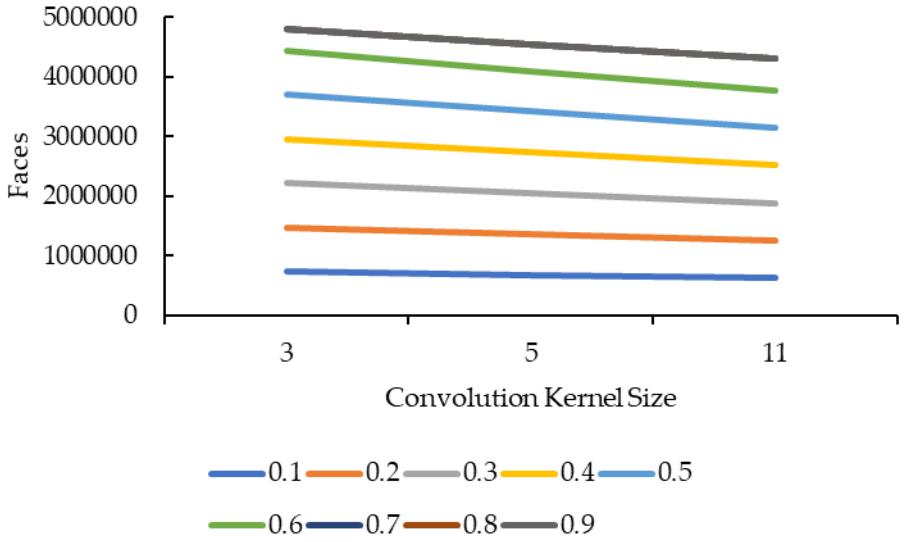

Figure 16. Graph of number of faces against convolution kernel size.

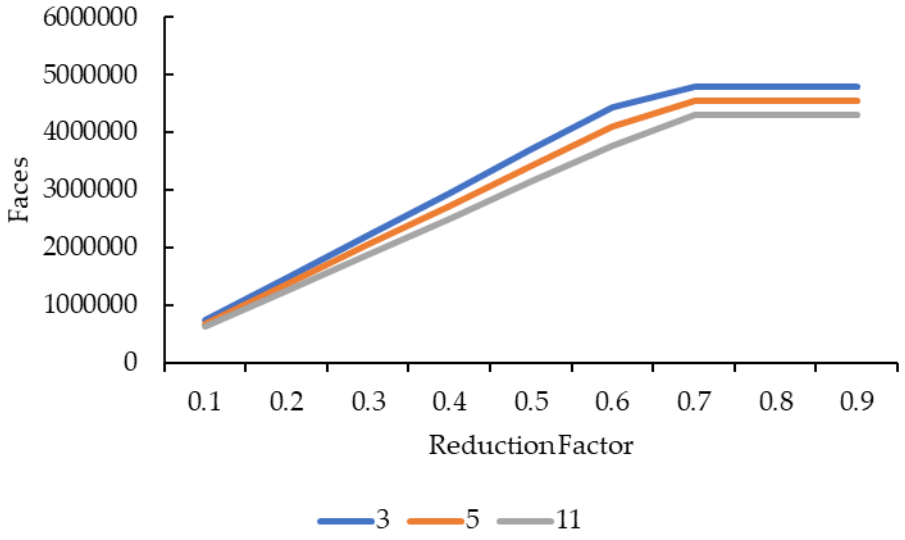

Figure 17. Graph of number of faces against reduction factor.

Figure 16 and Figure 17 showed the same line patterns as in Figure 14 and Figure 15 respectively. This is because faces are dependent on the vertices, hence exhibiting the same pattern. In Figure 16, the number of faces decreases as convolution kernel size increases for all reduction factors. Similarly, in Figure 17, the number of faces showed a logarithmic increase as the reduction factor increases for all convolution kernel sizes in both convolution kernel box and Gaussian. The line plateaued at a reduction factor of 0.7 onwards.

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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

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 2, two possible explanations can be made on the shape of the lines: one, the graph exhibits exponential growth, meaning to say that the increase in SSIM values is proportional to the increase in 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 convolution kernel size increases. This is largely due to the lack of results as only three convolution kernel sizes are tested. Also, because there is a huge jump of 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 convolution kernel size increases. Nevertheless, it is safe to say 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 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 4, however, is a different story from Figure 2, 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 large the convolution kernel size, the higher the MS-SSIM values are. One possible explanation for this is the same as the explanation for Figure 2.

As for the lines in Figure 3 and Figure 5, it is clear that the increase in reduction factor does not equate to an increase in 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 convolution kernel size 3 showed slight variation in values whereas 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. Also, as the comparison is made between two 2D images, it does not compare well for mesh simplified models.

For Figure 6 and Figure 8, the same logic and explanation can be applied here from Figure 2: 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, hence more time is needed to complete the convolution process.

In Figure 7, because the lines do not exhibit a specific pattern in shape, hence 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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

However, this explanation only holds true for the convolution kernel box. Referring to Figure 9, 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 Figure 10 and Figure 12 showed similarity in terms of line shape and pattern whereby the larger the convolution kernel size for both convolution kernel box and Gaussian, the lower the rendering time for all reduction factors. As 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 Figure 14 and Figure 16. In Figure 14 and Figure 16, 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, hence 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 Figure 11 and Figure 13 showed similarity in terms of line shape and pattern, which is also the same case for graphs in Figure 15 and Figure 17. 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 reduction factor 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, hence the number of vertices and faces remained the same from reduction factor 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 reduction factors 0.7 to 0.9. This is because the recorded rendering time changes per run, hence 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 both 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, different convolution kernel does not have any effect on the reconstruction accuracy, but it affects the reconstruction time. Screenshots of the reconstructed 3D models are as illustrated in Figure 18.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 25 October 2021 doi:10.20944/preprints202110.0362.v1

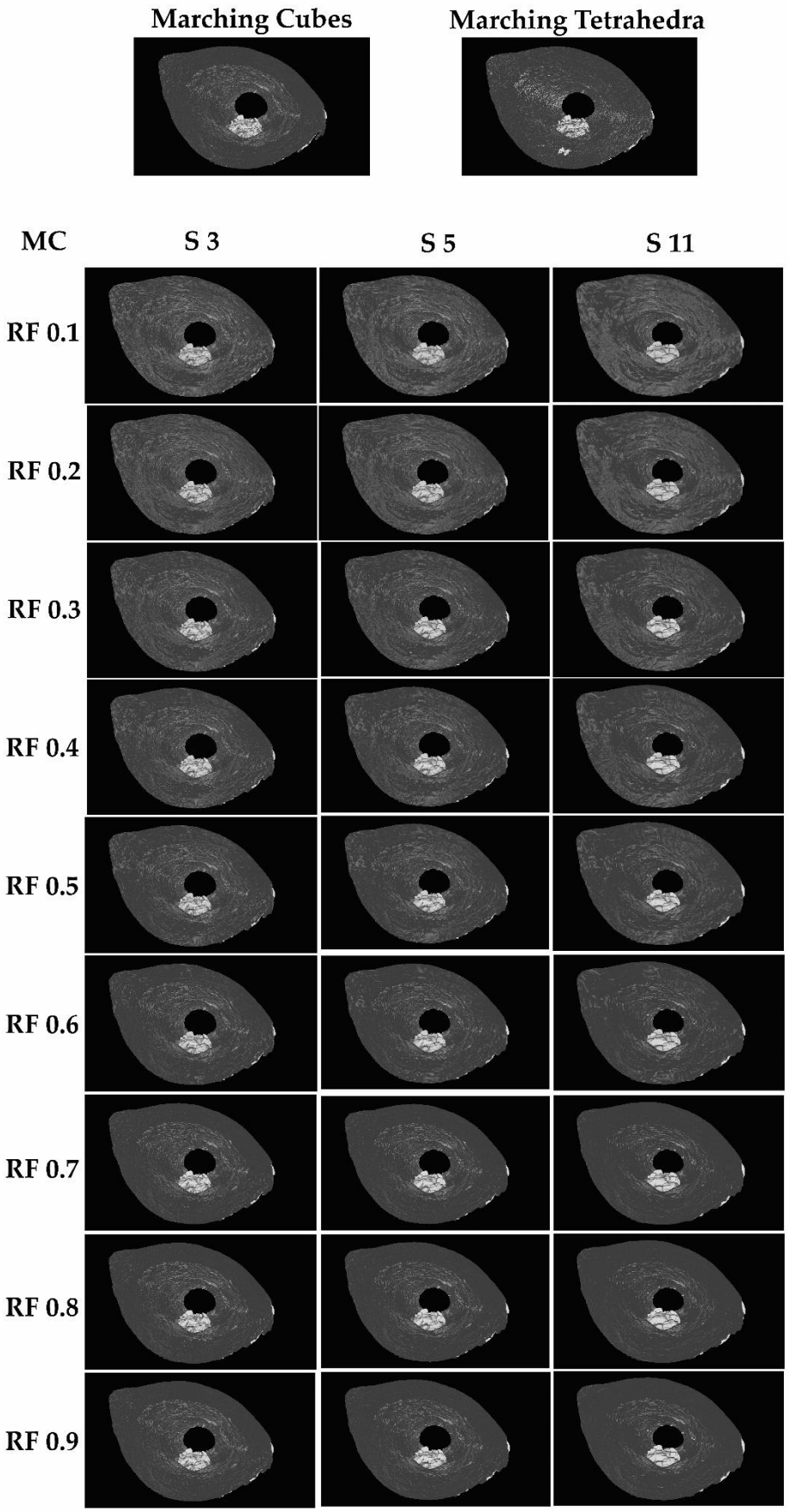

Figure 18. 3D models reconstructed with original Marching Cubes, Marching Tetrahedra and proposed improvement.

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 all in one go;
6. Implementing with languages like C++;
7. Different metrics to evaluate the reconstruction accuracy;
8. Different code optimization approaches.

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 0.5 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, surface rendering techniques do have 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;
- Reduction factor does not affect the reconstruction accuracy;
- The larger the convolution kernel size, the higher the reconstruction time;
- Reduction factor has an effect on the reconstruction time but no specific growth pattern can be deduced from the graphs, hence 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 reduction factor 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 reduction factor 0.7 where it stopped increasing;
- Different convolution kernels do not affect the result of reconstruction both qualitatively and quantitatively, except for 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 ASEAN University Network/Sourtheast Asia Engineering Education Development Network (AUN/SEED-Net) under the Special Program for Research Against COVID-19 (SPRAC) grant, grant number [304/PBAHAN/6050449/A119].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement Not applicable.

Data Availability Statement: The data presented in this study are available on 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.

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.
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.

3. Stein, D.; Assaf, Y.; Dar, G.; Cohen, H.; Slon, V.; Kedar, E.; Medlej, B.; Abbas, J.; Hay, O.; Barazany, D.; Hershkovitz, I. 3D virtual reconstruction and quantitative assessment of the human intervertebral disc's annulus fibrosus: a DTI tractography study. *Sci. Rep.* **2021**, *11*, 6815.
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.
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.
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.
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.
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.
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.
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.
11. Lorensen, W.E.; Cline, H.E. Marching cubes: A high resolution 3D surface construction algorithm. *Comput. Graph.* **1987**, *21*, 163–169.
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.
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.
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), LA, USA, 3–8 August 1997; ACM Press/Addison-Wesley Publishing Co.: NY, USA, 1997; 209–216.
15. Chernyaev, E.V.; Marching Cubes 33: Construction of Topologically Correct Isosurfaces. In Proceedings of 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. Brazilian Comput. Soc.* **2019**, *25*, 6.
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.
18. Masala, G.L.; Golosio, B.; Oliva, P. An improved Marching Cube algorithm for 3D data segmentation. *Comput. Phys. Commun.* **2013**, *184*, 777–782.
19. Wang, M.; Luo, H.; Cui, Q. Three-Dimensional Reconstruction Based On Improved Marching Cubes Algorithm. *J. Mech. Med. Biol.* **2020**, *20*, 2040002.
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.
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.
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.
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. *Chinese J. Aeronaut.* **2021**, *34*, 208–224.
25. Marching Cubes – File Exchange – MATLAB Central. Available online: https://www.mathworks.com/matlabcentral/fileexchange/32506-marching-cubes (accessed on 27 September 2021).
26. Polygonising a scalar field (Marching Cubes). Available online: http://paulbourke.net/geometry/polygonise/ (accessed on 29 September 2021).
27. 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, CA, USA, 9–12 November 2003; IEEE; 1398–1402.
28. Smooth 3-D data – MATLAB smooth3. Available online: https://www.mathworks.com/help/matlab/ref/smooth3.html (accessed on 1 October 2021).
29. Reduce number of patch faces – MATLAB reducepatch. Available online: https://www.mathworks.com/help/matlab/ref/reducepatch.html (accessed on 1 October 2021).
30. 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).
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).