Surface Reconstruction: An Improved Marching Triangle Algorithm for Scalar and Vector Implicit Field Representations

Marc Fournier
Quebec University in Montreal
CP 8888, Succ. Downtown, Montreal, Canada H3C 3P8
fournier.marc@uqam.ca

Abstract

In this paper we propose a new polygonization method based on the classic Marching Triangle algorithm. It is an improved and efficient version of the basic algorithm which produces a complete mesh without any cracks. Our method is useful in the surface reconstruction process of digitized objects. It works over the discrete distance transform of the object to produce the resulting triangle mesh. The new algorithm is also adapted to a recently introduced vector field distance transform model which is more accurate than the classic scalar field discrete distance transform of meshes. Our polygonization method is simplified and it produces better results compared to Marching Triangle basic algorithm while working on the vector field distance transform model. We use relevant error metric tools to compare results and show our new method is more accurate than Marching Cube which is the most widely used triangulation algorithm in the surface reconstruction process of digitized objects.

Keywords: *Marching Triangle, Discrete distance transform, Polygonization algorithm, Surface reconstruction, Digitized objects, Triangle mesh surface.*

1. INTRODUCTION

In the past decade a lot of improvements have been made in the field of 3D scanners to acquire and to digitize real world objects. The advancement in computer technologies have made possible the design of 3D scanners to respond to the increasing needs in many fields such as digitizing precious cultural heritage [1]. The most common output data structure produced by 3D scanners is range images. From these raw data, many operations are needed in order to produce the final mesh which represents the real object geometry. All these operations can be achieved in the scalar field distance transform (SFDT) domain which is often used because it produces good results for each step of the reconstruction procedure.

To perform surface reconstruction in the SFDT domain, one needs to convert the explicit range images, or other initial triangle meshes dataset, by computing their SFDT discrete implicit field which is defined over a regular 3D grid created inside a mesh bounding box. The cubic grid cells are called voxels and for each voxel, the closest point on the mesh surface is found and the shortest distance to the mesh is saved in the voxel. For a given surface $S \subset \Re^3$ this volume representation consist of a scalar value function $f: \Re^3 \rightarrow \Re$ such as the zero-set $f(x,y,z)=0$ defines the surface and in that case $[x,y,z] \in S$. To obtain a unique volumetric description for a given surface, this distance field is also signed according to surface normal vectors.

The surface reconstruction process begins with a mesh registration procedure [2] which is performed to express all range images in the same coordinates system. The first operation in the SFDT domain is mesh fusion [3] to integrate all initial range images into a unique representation, followed by mesh repair [4] to fill holes in the model, and then mesh smoothing [5] to remove acquisition noise introduced by the scanner and finally mesh simplification [6] to produce a more compact model without loss of details. Since the SFDT is an implicit representation, at the end of the reconstruction process a polygonization algorithm such as Marching Cube [7] or Marching Triangle [8] is needed to produce the final explicit mesh which describes the digitized object surface geometry.

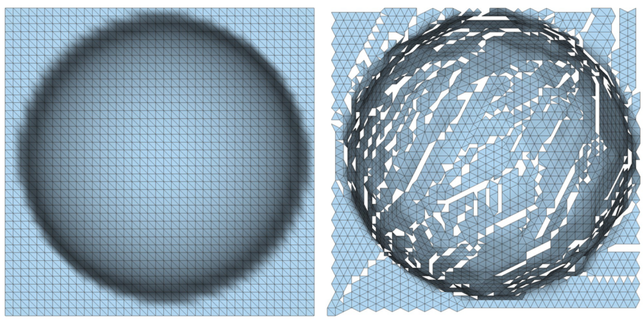
In this paper we introduce a new polygonization method to triangulate the resulting mesh of the surface reconstruction process in the SFDT domain. We propose a set of improvements based on the Marching Triangle algorithm to obtain an efficient method which overcomes some crack problems in the basic version to produce a higher quality resulting mesh. We also adapt our method to the vector field distance transform (VFDT) implicit representation which allows a simplification and a more accurate result using the Marching Triangle algorithm. The remaining parts of the paper are organized as follows: Section 2 overviews related work. Section 3 presents our new improved triangulation algorithm. Section 4 describes the Marching Triangle adaptation to the VFDT representation. And in Section 5, before concluding, we show and compare our triangulation method results with previously introduced algorithms.

2. RELATED WORK

The most widely used algorithm to triangulate a SFDT is Marching Cube [7]. It is a volume-based approach which is very suitable to triangulate discrete implicit fields such as SFDT in the surface reconstruction process of digitized objects. The original Marching Cube method has some ambiguities and many other algorithms, based on the original one, have been proposed to improve the resulting mesh quality. For example the original algorithm has 14 cube triangulation configurations which lead to face ambiguities resolved in [9], and [10] in which the 14 basic configurations are expended in 32 different cases. A remaining cube ambiguity was then solved in [11] to guaranty the resulting mesh topology.

Methods derived from the basic Marching Cube algorithm have been proposed to also resolve the ambiguities. Cubical Marching Squares [12] which opens the cubes into six squares and Marching Tetrahedron [13] which divides the cubes into tetrahedrons are two examples of cube configurations which resolve original ambiguities. Other algorithms have been introduced to improve the resulting mesh quality. The Extended Marching Cube [14] provides a sensitivity feature to better recover sharp edges and Dual Contouring [15] focus on preserving the resulting mesh topology. More recently, the VFDT model [16] which is a vector extension of the SFDT has been proposed to improve the implicit field accuracy and the Marching Cube algorithm has been adapted to this new representation in order to produce a higher quality resulting mesh.

Another well known algorithm to triangulate a SFDT is Marching Triangle [8]. It is a surface-based approach built on Delaunay triangulation definition. Starting from a seed triangle, a region-growing process enables the creation of triangles following the SFDT isosurface. This surface tracking process has been designed to triangulate discrete SFDT of digitized objects and it has also been applied to continuous implicit surfaces describing virtual objects with a set of equations such as parametric ones. The basic Marching Triangle algorithm leaves some part of the model un-triangulated creating cracks in the resulting mesh as shown in Figure 1.



a) Original model b) Marching Triangle result

Figure 1: Basic Marching Triangle algorithm result on the Half-Sphere model

In Figure 1a) a Half-Sphere model lying on a plane is used and its SFDT is computed. Figure 1b) shows the basic Marching Triangle algorithm result which contains cracks in the triangulated resulting mesh. Theses cracks occur because of different reasons which will be detailed in next section.

Beside the classic Marching Triangle, other methods based on a region-growing and surface tracking process have been proposed such as Gopi algorithm [17] which works over a set of unorganized points and Hartmann algorithm [18] which works on continuous implicit surfaces. In this paper we focus on improving the steps of the original Marching Triangle to directly triangulate discrete SFDT of digitized objects. The methods designed to work over point clouds are also suitable for discrete SFDT as described in the Ball-Pivoting method [19] in which a first pass generate points from the SFDT before applying the triangulation on the set of generated points. This Ball-Pivoting method is similar to Marching Triangle, it also uses the Delaunay sphere test but the constraint is applied on the sphere radius instead of the sphere center as for the original Marching Triangle algorithm. In this paper we also modify the Delaunay sphere for the triangulating test which contributes to reduce the resulting mesh cracks.

Previous methods [20, 21, 22] applied to continuous implicit surfaces have been introduced to overcome the basic Marching Triangle algorithm cracks problem. These method triangulations are adaptive to local implicit curvature to obtain variable size triangles. In this paper we also modify the basic algorithm with a variable projection distance to obtain a better adaptive result over discrete SFDT. Some of these methods [20, 21] operate in two passes. They first triangulate a resulting mesh according to the basic algorithm which contains cracks and then they introduce a crack filling algorithm to complete the resulting mesh. The other method [22] introduces a different region-growing algorithm compared to the original Marching Triangle. It is based on a hexagonal triangulation expansion pattern which is able to resolve cracks and to produce a complete resulting mesh in a single pass. In this paper we use the original Marching Triangle algorithm procedure and we propose improvements to several steps of the method. Our new triangulation works over a discrete SFDT in a single pass, as for the original algorithm, and produce a complete mesh without cracks. We do not need a second specific post-processing crack filling pass but instead we introduce an iterative process on our single pass to obtain a complete mesh.

Other algorithms have been proposed to improve the basic Marching Triangle triangulation over discrete implicit surfaces and to achieve specific goals. A topology preserving method [23] introduces a normal consistency constraint which guaranties the resulting mesh topology. An edge constrained method [24] detects discontinuities in the implicit surface and constrains triangle edges to match and better preserve these sharp features in the resulting mesh. In this paper we have a more global approach to produce a higher quality mesh by improving the entire original Marching Triangle algorithm. We also adapt our new triangulation method to the VFDT model [16] which is more accurate than the SFDT. Therefore we obtain a more globally accurate resulting mesh including a better sharp features preserving compared to the original Marching Triangle algorithm.

3. MARCHING TRIANGLE IMPROVED ALGORITHM

In this section we describe our new Marching Triangle improved algorithm. We refer to the original algorithm [8] procedure numbering to identify the steps to improve. In Subsection 3.1 we improve steps 1 and 2 of the original algorithm in finding a new vertex. In Subsection 3.2 we improve step 4 in testing a new triangle. In Subsection 3.3 we improve step 6 in considering new triangles. Finally in Subsection 3.4 we introduce an edge processing sequence to improve the overall algorithm.

3.1 Variable Projection Distance and Vertex Interpolation

The first and second steps of Marching Triangle algorithm are illustrated in Figure 2. The first step is the estimation of a new vertex position P’ with a projection perpendicular to the mid-point P of the boundary edge C in the plane of the model boundary triangle ABC by a constant distance PP’. The second step is the evaluation of the nearest point to P’ which is on the implicit surface. That new potential vertex is V2 in Figure 2 example.

The constant distance projection contributes to produce cracks in the mesh. In high curvature regions the projection can be further away from the isosurface and the new vertex estimation may cause either a test failure resulting in a crack beginning or a bad approximation of the surface local geometry. To correct this problem we propose a variable projection distance which is equal to √3/2 times the length of the boundary edge. This projection distance corresponds to the height of the equilateral triangle composed of the boundary edge. This improvement contributes to obtain more equilateral triangles which are less deformed and which sizes adapt gradually to local geometry curvature.

From the projection point P' on Figure 2 example, we suppose the nearest voxel which is considered on the isosurface is V2. While working with a discrete SFDT a threshold distance comparison is needed to find this nearest voxel. To consider that a voxel is on the isosurface it must contain a distance smaller than half the grid resolution. This approximation introduces a significant error in the vertex position according to the underlying implicit surface and this error can also contributes to a crack in the mesh if the new triangle test fails. To correct this problem we propose a linear vertex position interpolation between the nearest voxel found and its closest neighbor with opposite distance sign. The interpolation finds the new vertex position which corresponds to a distance equal to zero between the two distances of opposite sign. This interpolation uses the implicit surface definition to obtain a more accurate approximation of the isosurface.

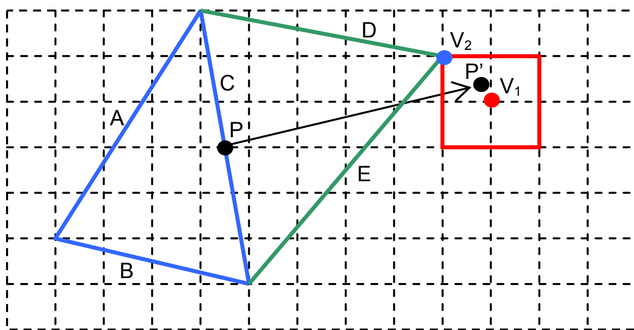
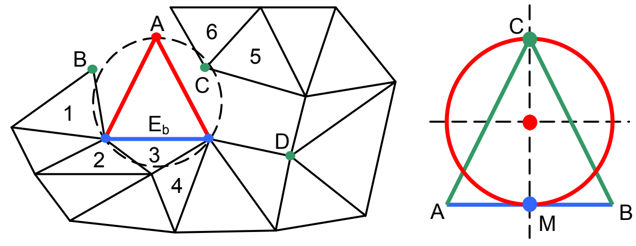


Figure 2: Marching Triangle projection and nearest vertex on the isosurface

3.2 Modified Delaunay Sphere Test

Step 4 of the original algorithm suggests testing the new potential triangle according to the Delaunay sphere which is circumscribed to the new triangle as shown in Figure 3a) in which the current boundary edge E_b forms a new triangle with the new estimated vertex A. In Figure 3a) example the test would fail because there are parts of the six numbered triangles inside the sphere. According to the original algorithm, further tests would be made with edge E_b and vertices B, C and D to consider these three new triangles. These tests would also fail because the Delaunay sphere would still contain parts of neighbor triangles. This test limits the original algorithm performances and contributes to produce cracks in the mesh. The Delaunay test sphere was designed to triangulate a set of unorganized points which is not exactly the same situation in the case of a region-growing algorithm over a SFDT.

We propose the modified sphere shown in Figure 3b) for new triangles testing. The modified sphere passes through the mid-point M of the current boundary edge and the new estimated vertex C. Its diameter is equal to the distance between these two points and its center is the mid-point between M and C. The modified sphere is smaller than the original one and it allows obtaining more successful tests which improve the algorithm and reduce the cracks in the resulting mesh. Since some parts of the new triangle are outside the sphere, we also need to test if there is no intersection between the new triangle and other triangles of the mesh. We test all triangles which have parts inside the original Delaunay sphere with the new triangle according to Moller intersection test [25] which is fast and efficient.



a) Interfering triangles with the original sphere b) Modified test sphere

Figure 3: Interfering triangles with the original Delaunay sphere and modified test sphere

3.3 New Triangles to Consider

Step 6 of the original algorithm suggests considering two new potential triangles illustrated in Figure 3a) if the first new triangle test fails at step 4. These new triangles are composed of the current boundary edge E_b and vertices B or D which are the previous and the next vertices along the boundary from the current boundary edge E_b. If the sphere test fails for these two new triangles, other new triangles should be tested to improve the algorithm. The original algorithm was upgraded in [26] with a seventh step which suggests to test another new triangle composed of the current boundary edge E_b and vertex C in reference of Figure 3a). Vertex C is the nearest boundary vertex of overlapping triangles number 5 and 6 in the sphere test. This new potential triangle contributes to reduce the cracks in the mesh compared to the original algorithm but if the test fails with this triangle other triangles should be tested to better improve the algorithm.

We propose to test not only the nearest but all boundary vertices of overlapping triangles if they exist. In some particular cases as the one shown in Figure 4 the sphere test would fail with the nearest boundary vertex leaving a part of the mesh un-triangulated and a new triangle could be added to the mesh if another boundary vertex of the overlapping triangle was considered. In Figure 4 E_b is the current boundary edge and V_1 is the triangle nearest vertex from E_b. The new potential triangle composed of E_b and V_1 would fail the test but another triangle composed of E_b and V_2 would be a better candidate even if the distance between E_b and V_2 is greater than the distance between E_b and V_1.

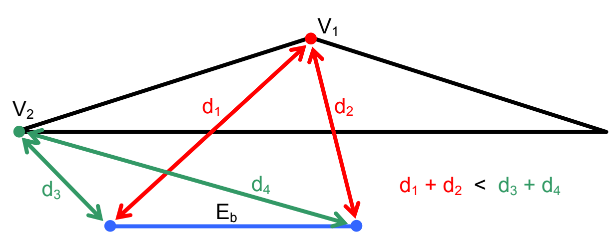
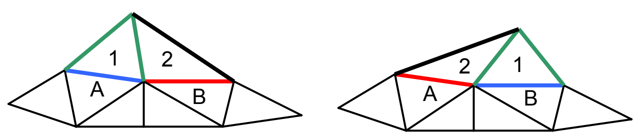


Figure 4: New triangles to consider in the triangulation process

3.4 Edge Processing Sequence and Iterative Process

The original Marching Triangle algorithm does not specify any boundary edges processing sequence. It is only defined as a single pass into the edge list to process all boundary edges with the procedure steps including the estimation of a new potential triangle and its sphere test to determine if it will be added or not to the mesh. According to the implemented data structure to triangulate the SFDT and the method to add new edges in the edge list, the edge processing sequence can be different from one implementation to another. The resulting mesh from the Marching Triangle depends on the edge processing sequence and it can be different if the sequence is changed as shown in Figure 5.



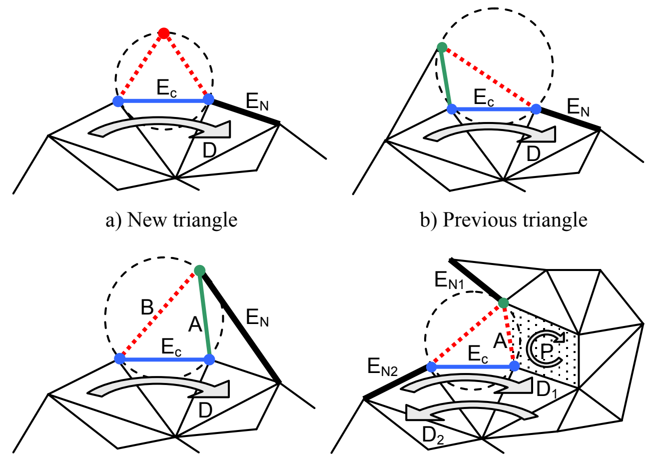
a) First possible mesh b) Second possible mesh

Figure 5: Modified mesh according to the edge processing sequence

In Figure 5 example we assume the current mesh is composed of the six bottom triangles, therefore edges A and B are boundary edges. In Figure 5a) edge A is considered first and triangle 1 is added then edge B is processed and triangle 2 is added next. In Figure 5b) edge B is considered first and triangle 1 is added first then edge A is tested and triangle 2 is also added. In Figure 5 simple example both results are different just because of an edge permutation. The results would have been more different if for example after adding triangle 1 in Figure 5a) the new boundary edges of that triangle were processed before edge B and if that same processing sequence was applied in Figure 5b) to the new boundary edges of triangle 1.

We tested different edge sequences and combinations on the original algorithm and we selected the one which optimizes the result in terms of minimizing the cracks in the mesh. We propose the following procedure to improve the resulting mesh quality. Starting from a boundary edge, an arbitrary direction is selected and the next edge to consider is the neighbor boundary edge always in that same direction around the contour of the current mesh. Newly added boundary edges are not considered immediately, they will only be considered on the next turn around. This procedure is illustrated in Figure 6 with E_c corresponding to the current boundary edge to be processed and E_N the next boundary edge to consider according to the chosen arrow direction D.

In Figure 6a) New triangle and Figure 6b) Previous triangle cases the next boundary edge to consider is straight forward. In Figure 6c) Next triangle case if the test is successful and the new triangle is added then the next boundary edge to consider jumps over edge A which is lo longer a boundary edge. The new added boundary edge B will be considered on the next turn around only. In Figure 6d) Overlapping triangle case a bridge is created in the mesh if the new triangle is added. In that case the next edge to consider is either E_{N1} or E_{N2} depending on the chosen direction D_1 or D_2 respectively. In that special case the inside contour in the region P is triangulated in priority immediately after the new triangle is added to the mesh and before pursuing with the outside contour. Starting from the new inside boundary edge A an arbitrary direction is selected and the previously described procedure is applied to the inside contour until no more triangle can be added.



a) New triangle b) Previous triangle

c) Next triangle d) Overlapping triangle

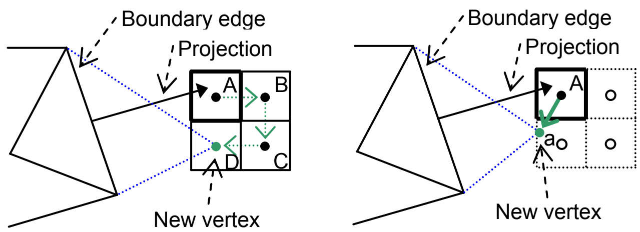
Figure 6: Next edge to consider according to the edge processing sequence

The original Marching Triangle is defined as a single pass into the edge list. Our proposed procedure is iterative and the boundary edges are tested more than once, until a complete loop is made around the mesh contour without adding any triangle. A boundary edge can be tested without adding any triangle to it at the first pass and at the second pass the test could be successful depending on the local mesh neighbourhood configuration which can be different from one pass to another. Our iterative procedure continues as long as new triangles are added. Compared to the original algorithm our procedure contributes to add more triangles and to reduce the cracks in the final resulting mesh.

4. VECTOR FIELD AND ERROR METRIC ADAPTATION

We also adapt our improved Marching Triangle algorithm as defined in Section 3 to the VFDT [16]. The discrete vector field distance transform is an extension of the SFDT. Instead of saving in each voxel only the scalar shortest distance to the surface as in the SFDT, in the VFDT a vector is saved in each voxel. This vector gives the shortest distance to the surface and in addition it also gives the orientation of the closest point on the surface. This new representation is more accurate than the classic SFDT and it is used in the surface reconstruction process of digitized objects. Marching Cube algorithm was previously adapted to the VFDT and this adaptation produces better triangulation results than other Marching Cube versions over the SFDT. Our Marching Triangle adaptation to the VFDT is simple and it has two advantages. First it saves computation time and second it also produces more accurate results compared to the scalar version over the SFDT. Figure 7 illustrates our Marching Triangle adaptation to the VFDT.

Figure 7 shows steps 1 and 2 of the Marching Triangle algorithm with a) the SFDT and b) the VFDT. Step 1 is the same for both representations; a projection is made from the boundary edge mid-point to estimate a new vertex position. Step 2 is different and a search is needed with the SFDT to find the nearest voxel which is considered on the isosurface. In Figure 7a) example the search stopped on voxel D which is the nearest voxel from the projection point to contain a distance smaller than half the grid resolution. In the original Marching Triangle algorithm, voxel D coordinates are used as the new vertex which forms the new potential triangle to be tested. This is a coarse approximation of the isosurface. In Section 3.1 of this paper we propose a linear interpolation to obtain a better estimation of the isosurface. This is an improvement from the original algorithm but the result still is an approximation. In Figure 7b) there is no need for a search and interpolation with the VFDT. From the projection point which is in voxel A, the vector saved in that voxel is used and a simple redirection gives immediately the coordinates of the new vertex. According to the VFDT definition, this new vertex is the nearest point from the projection point which is exactly on the isosurface. With the VFDT, step 2 of the Marching Triangle is simplified and upgraded to a better result. The algorithm other steps are exactly the same as for the SFDT.



a) SFDT search b) VFDT redirection

Figure 7: Comparison of the new vertex estimation with the SFDT and the VFDT

In order to evaluate and compare our improved algorithm on the SFDT, we use the vertex to surface error metric defined in [5] to quantify the relative quality of the triangulation algorithms tested. This error metric is based on the minimal Hausdorff distance between two meshes. We start from a reference mesh and compute its SFDT. Then we triangulate the SFDT with different algorithms. The resulting meshes are compared to the reference mesh using this vertex to surface error metric which gives a scalar value of the average distance or error between two meshes. The smallest distance from each vertex of the triangulated result to the surface of the reference mesh is evaluated and a weighted average of these distances gives the associated error of the triangulation result compared to the reference mesh. Then the triangulation result errors of different algorithms from the same reference mesh can be compared together to evaluate the relative quality of each result.

The vertex to surface error metric is not suitable to evaluate and compare our adapted algorithm to the VFDT because the resulting triangulation vertices are exactly on the reference mesh surface. Using the vertex to surface error metric with our adaptation would produce an error equal to zero even if our result and the reference mesh are different. In order to evaluate and compare our VFDT adaptation, we define a triangle to surface error metric which is based on the previous vertex to surface error metric. The difference is instead of computing the distance from the resulting mesh vertices to the reference mesh surface, we compute the distance from each triangle centroid of the resulting mesh to the reference mesh surface. Our triangle to surface error metric is defined by:

$$\varepsilon_t = \sqrt{\frac{1}{A(M')} \sum_{i=1}^{N} A(t'_i)\, dist(ct'_i, M)^2} \qquad (1)$$

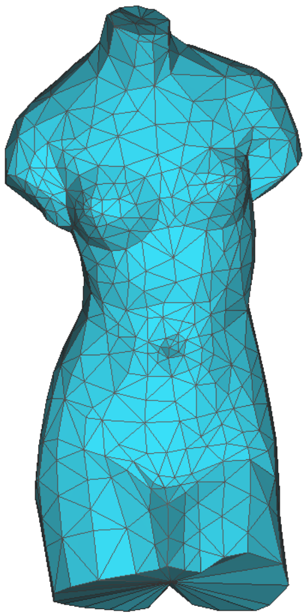
M is the initial reference mesh and M' is the triangulated result to evaluate. A(M') is the total area of M'. $A(t'_i)$ is the area of each triangle t'_i of the resulting mesh. ct'_i are the resulting mesh triangles centroid. $dist(ct'_i, M)$ is the minimal distance between the centroid ct'_i and the initial reference mesh M. This triangle to surface error metric is used to compare all results when our Marching Triangle VFDT adaptation is evaluated.

5. RESULTS

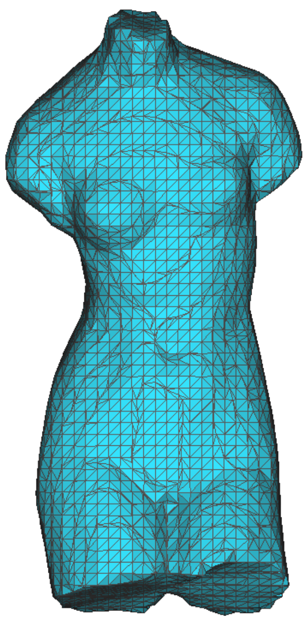
In this section we evaluate our improved algorithm on both SFDT and VFDT representations and we compare our results with Marching Cube triangulation using the vertex to surface and the triangle to surface error metrics with the procedure described in the previous section. Regarding the Marching Cube results, we implemented the Cubical Marching Squares [12] and the Extended Marching Cube [14] algorithms which are two recent and efficient versions and we kept the one showing the best result for each model tested.

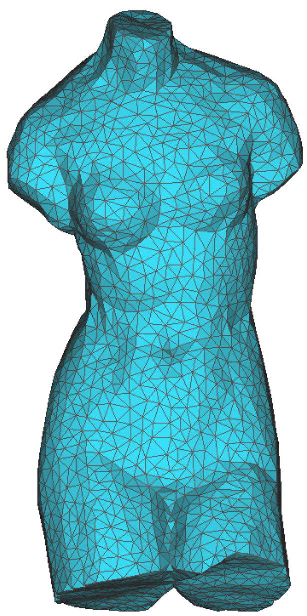
First we used the Venus model reference mesh shown in Figure 8a) to compute its SFDT with an appropriate grid resolution according to the model level of details. Then we triangulated the SFDT with Marching Cube and our improved Marching Triangle algorithms and these results are shown in Figure 8. We compared these two results to the reference mesh using the vertex to surface error metric and Table 1 shows these error values along with the number of triangles and the triangulation computing times for both results. The timing measures were made using a Pentium 4 CPU computer with a 3.03GHz clock.

In Figure 8 both results are of good quality but we see that the triangles of Marching Cube result are more dependent on the voxels size. Figure 8b) result also shows small degenerated triangles forming elevation lines depending on the grid resolution which is the classic signature of Marching Cube algorithm. Marching Triangle result in Figure 8c) shows a more homogeneous triangulation and sharp edges such as the one at the bottom are better preserved compared to Marching Cube result. Table 1 also shows that Marching Triangle result is of better quality according to the error metric. Our result also contains fewer triangles, thus optimizing the model quality, storage and memory space with almost one third less triangles than Marching Cube result. The drawback of our algorithm is the computation time which is almost the double compared to Marching Cube. The triangulation time is still reasonable for this model but it could make a difference for example in real time applications on large models.



a) Reference mesh





b) Marching Cube c) Marching Triangle

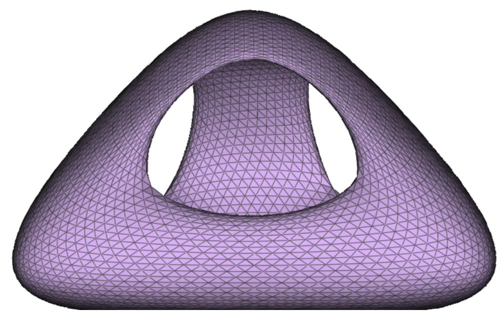
Figure 8: Triangulation results on the Venus model

Table 1: Triangulation parameters for the Venus model

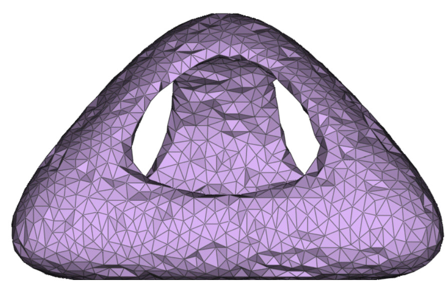
Parameters	Marching Cube	Marching Triangle	Difference
Nb Triangle	7 465	5 152	31.0%
Error	7.58×10^{-3}	6.88×10^{-3}	9.23%
Time (ms)	26.7	48.5	81.6%

We used a Genus3 virtual model to compare our algorithm adaptation to the VFDT and results are shown in Figure 9. For this evaluation a coarse grid resolution was used to highlight the advantage of the vector field adaptation. Starting with the initial model in Figure 9a) we computed both SFDT and VFDT of the model at same resolution. Then we triangulated both implicit representations with our Marching Triangle improved algorithm. Results in Figure 9b) and 9c) were compared to the initial model using our triangle to surface error metric definition and Table 2 shows the results for this model.

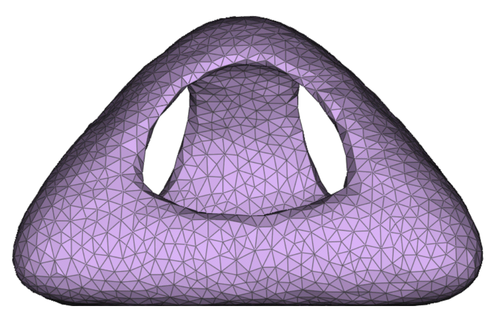
In Figure 9 we see that our VFDT adaptation is of better quality compared to the algorithm applied to the SFDT. Using the exact isosurface point at step 2 in the VFDT adaptation produces a more accurate result than the approximation of the isosurface with the SFDT. Table 2 error metric values show that our adaptation takes advantage of the VFDT improved representation to obtain a better result. The timings in Table 2 show that the simplification of step 2 in the algorithm adaptation to the VFDT is faster than with the SFDT but the difference is not significant compared to the overall algorithm timing. The resulting meshes numbers of triangles are very similar with both representations; our VFDT adaptation produced a few more triangles compared to the SFDT version.



a) Reference mesh



b) SFDT triangulation



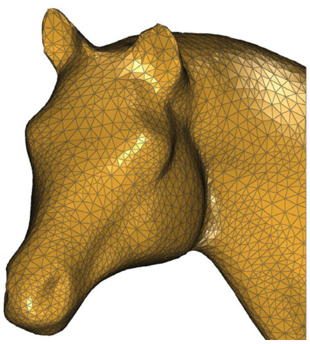
c) VFDT triangulation

Figure 9: Triangulation results on the Genus3 model

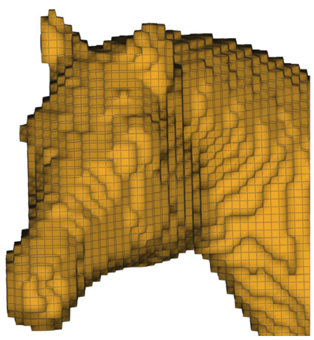
Table 2: Triangulation parameters for the Genus3 model

Parameters	SFDT Imp. Rep.	VFDT Imp. Rep.	Difference
Nb Triangle	3 481	3 615	3.71%
Error	6.14×10^{-2}	5.63×10^{-2}	8.31%
Time (ms)	33.6	32.7	2,68%

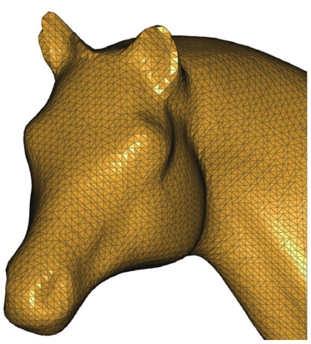
The Horse model shown in Figure 10 was used to compare previously introduced Marching Cube adaptation to the VFDT and our Marching Triangle adaptation to the VFDT. The voxel grid size shown in Figure 10b) was used to compute the VFDT of the initial mesh in Figure 10a). Figure 10c) and 10e) show both triangulation results over the VFDT.



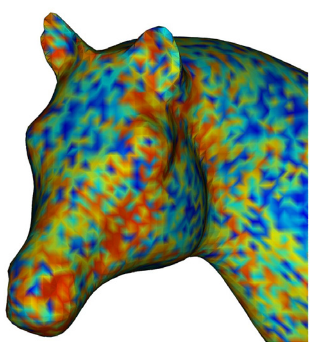
a) Initial reference mesh



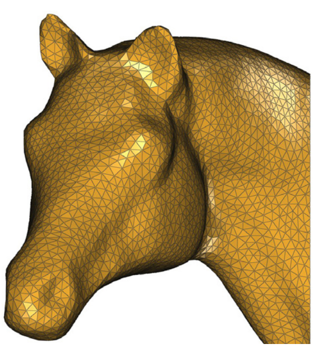
b) Voxel grid



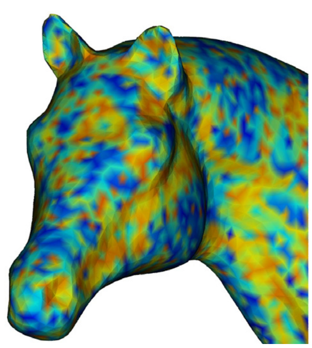
c) Marching Cube result



d) Marching Cube error



e) Marching Triangle result



f) Marching Triangle error

Figure 10: Triangulation results on the Horse model

The triangle to surface error metric was used to compare Figure 10 triangulations to the initial mesh and results are listed in Table 3 along with the other relevant parameters for this model. The local triangle to surface error metric computed at each triangle centroid has been converted into an error colormap in which red corresponds to the greatest and blue to the lowest error. These two colormaps are shown in Figure 10d) and 10f) for both triangulations.

Figure 10c) shows that Marching Cube adaptation to the VFDT is of better quality than the algorithm applied to the SFDT. The result contains no more small degenerated triangle. But the triangle sizes are still very dependent on the grid resolution and Marching Cube result still contains significantly more triangles compared to our Marching Triangle adaptation. Marching Cube adaptation to the VFDT provides a better approximation than the SFDT version but vertices positions are not exactly on the isosurface, they still are approximations. Therefore our Marching Triangle adaptation which takes fully advantage of the VFDT representation with exact vertices positions on the isosurface produces a more accurate result according to the error metric in Table 3. Visually in Figure 10 we also see that Marching Triangle colormap contains less red color errors compared to Marching Cube colormap. Even if our algorithm processing time is a bit faster over the VFDT compared to the SFDT, it is still much slower than Marching Cube adaptation to the VFDT.

Table 3: Triangulation parameters for the Horse model

Parameters	Marching Cube	Marching Triangle	Difference
Nb Triangle	57 251	46 716	18.4%
Error	3.40×10^{-3}	3.19×10^{-3}	6.18%
Time (ms)	245.4	429.7	75.1%

6. CONCLUSION AND FUTURE WORK

In this paper we designed a new Marching Triangle algorithm to improve the triangulation result of such method over the SFDT of digitized objects in their surface reconstruction process. Our contribution focused on improving several steps of the original algorithm to overcome the crack forming problem. We proposed a variable projection distance and a vertex position interpolation to provide a better isosurface approximation. We introduced a modified sphere for testing potential triangles geometry consistency before adding them to the resulting mesh. We also proposed testing new potential triangles which can lead to more complete results in particular cases. We structured an edge processing sequence which is more efficient during the triangulation process. Our new algorithm was simplified and adapted to the VFDT to provide more accurate results based on this improved representation. We compared our algorithm results to Marching Cube triangulation and demonstrated its relevancy based on error metric measurements. In that comparison procedure we also designed a new error metric which is more suitable for our algorithm adaptation to the VFDT.

Future work will include optimizing the processing time of every step of our new algorithm since it is a drawback compared to Marching Cube performances. We will mainly focus on the triangle geometry consistency testing step since it is the most time consuming one of the overall algorithm. We will also work on adapting our new algorithm to other representations such as continuous implicit surfaces, point clouds and 3D volumetric datasets for medical and related applications. Adapting our algorithm to these representations will provide a useful tool to a wider range of applications in computer graphics. Surface-based triangulation algorithms such as Marching Triangle are more complex to design to obtain efficient results but in general their resulting meshes are of higher quality compared to volume-based methods which are usually simpler to implement.

7. REFERENCES

[1] Rocchini C., Cignoni P., Montani C., Pingi P., Scopigno R.: A Low Cost 3D Scanner Based on Structured Light. EUROGRAPHICS 2001, Manchester, UK, Computer Graphics Forum, Vol. 20, No. 3, 2001, 299-308.

[2] Besl P.J., McKay N.D.: A Method of Registration of 3D Shapes. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 2, 1992, 239-256.

[3] Hilton A., Stoddart A.J., Illingworth J., Windeatt T.: Implicit Surface-Based Geometric Fusion. Computer Vision and Image Understanding, Vol. 69, No. 3, 1998, 273-291.

[4] Davis J., Marschner S., Garr M., Levoy M.: Filling Holes in Complex Surfaces Using Volumetric Diffusion. First International Symposium on 3D Data Processing, Visualization, and Transmission, Padua, Italy, 2002, 428-438.

[5] Fournier M., Dischler J-M., Bechmann D.: 3D Distance Transform Adaptive Filtering for Smoothing and Denoising Triangle Meshes. International Conference on Computer Graphics and Interactive Techniques, Kuala Lumpur, Malaysia, 2006, 407-416.

[6] Nooruddin F.S., Turk G.: Simplification and Repair of Polygonal Models Using Volumetric Techniques. IEEE Transactions on Visualization and Computer Graphics, Vol. 9, No. 2, 2003, 191-205.

[7] Lorensen W.E., Cline H.E.: Marching Cubes: A High Resolution 3D Surface Reconstruction Algorithm. ACM SIGGRAPH 1987, Anaheim, USA, 1987, Computer Graphics, 163-169.

[8] Hilton A., Stoddart A.J., Illingworth J., Windeatt T.: Marching Triangles: Range Image Fusion for Complex Object Modelling. International Conference on Image Processing, Lausanne, Switzerland, 1996, Vol. 1.

[9] Nielson G.M., Hamann B.: The Asymptotic Decider: Resolving the Ambiguity in Marching Cubes. IEEE Conference on Visualization 1991, San Diego, USA, 1991, 83-91.

[10] Chernyaev E.V.: Marching Cubes 33: Construction of Topologically Correct Isosurfaces. Tech. Report, No. CN/95-17, CERN, Geneva, Switzerland, 1995.

[11] Lewiner T., Lopes H., Vieira A.W., Tavares G.: Efficient Implementation of Marching Cubes Cases with Topological Guarantees. Journal of Graphics Tools, Vol. 8, No. 2, 2003, 1-15.

[12] Chien-Chang H., Fu-Che W., Bing-Yu C., Yung-Yu C., Ming O.: Cubical Marching Squares: Adaptive Feature Preserving Surface Extraction from Volume Data. EUROGRAPHICS 2005, Dublin, Ireland, 2005, Computer Graphics Forum, Vol. 24, No. 3, 537-545.

[13] Chan S.L., Purisima E.O.: A New Tetrahedral Tesselation Scheme for Isosurface Generation. Computers and Graphics, Vol. 22, No. 1, 1998, 83-90.

[14] Kobbelt L.P., Botsch M., Schwanecke U., Seidel H.P.: Feature Sensitive Surface Extraction from Volume Data. ACM SIGGRAPH 2001, Los Angeles, USA, 2001, Computer Graphics, 57-66.

[15] Zhang N., Hong W., Kaufman A.: Dual Contouring with Topology-Preserving Simplification Using Enhanced Cell Representation. IEEE Visualization, Austin, USA, 2004, 505-512.

[16] Fournier M., Dischler J-M., Bechmann D.: A New Vector Field Distance Transform and its Application to Mesh Processing from 3D Scanned Data. The Visual Computer Journal, Vol. 23, No. 9-11, 2007, 915-924.

[17] Gopi M., Krishnan S., Silva C.T: Surface Reconstruction Based on Lower Dimensional Localized Delaunay Triangulation. Computer Graphics Forum, Vol. 19, No. 3, 2000, 467-478.

[18] Hartmann E.: A Marching Method for the Triangulation of Surfaces. The Visual Computer, Vol. 14, No. 3, 1998, 95-108.

[19] Bernardini F., Mittleman J., Rushmeier H., Silva C., Taubin G.: The Ball-Pivoting Algorithm for Surface Reconstruction. IEEE Transactions on Visualization and Computer Graphics, Vol. 5, No. 4, 1999, 349-359.

[20] Akkouche S., Galin E.: Adaptive Implicit Surface Polygonization Using Marching Triangles. Computer Graphics Forum, Vol. 20, No. 2, 2001, 67-80.

[21] Karkanis T., Stewart A.J.: Curvature-Dependent Triangulation of Implicit Surfaces. IEEE Computer Graphics and Applications, Vol. 21, No. 2, 2001, 60-69.

[22] Araujo B.R., Jorge J.A.P.: Curvature Dependent Polygonization of Implicit Surfaces. Brazilian Symposium on Computer Graphics and Image Processing, Curitiba, Brazil, 2004, 266-273.

[23] Xi Y., Duan Y.: A Region-Growing Based Iso-Surface Extraction Algorithm. IEEE International Conference on Computer-Aided Design and Computer Graphics, Beijing, China, 2007, 120-125.

[24] McCormick N.H., Fisher R.B.: Edge-Constrained Marching Triangles. International Symposium on 3D Data Processing Visualization and Transmission, Padova, Italy, 2002, 348-351.

[25] Moller T.: A Fast Triangle-Triangle Intersection Test. Journal of Graphics Tools, Vol. 2, No. 2, 1997, 25-30.

[26] Hilton A., Illingworth J.: Marching Triangles: Delaunay Implicit Surface Triangulation. Tech. Rep. EPSRC-GR/K04569, Centre for Vision Speech and Signal Processing, University of Surrey, Guildford, UK, 1997, 1-12.