Bio-Medical Materials and Engineering 26 (2015) S1417–S1429
DOI 10.3233/BME-151440
IOS Press

Novel irregular mesh tagging algorithm for wound synthesis on a 3D face

Sangyong Lee[a] and Seongah Chin[b,*]
[a]*Department of Computer, DXP Lab. College of Information and Communication, Korea University, Republic of Korea*
[b]*Division of Multimedia, College of Engineering, Xicom Lab. Sungkyul University, Anyang, Republic of Korea*

Abstract. Recently, advanced visualizing techniques in computer graphics have considerably enhanced the visual appearance of synthetic models. To realize enhanced visual graphics for synthetic medical effects, the first step followed by rendering techniques involves attaching albedo textures to the region where a certain graphic is to be rendered. For instance, in order to render wound textures efficiently, the first step is to recognize the area where the user wants to attach a wound. However, in general, face indices are not stored in sequential order, which makes sub-texturing difficult. In this paper, we present a novel mesh tagging algorithm that utilizes a task for mesh traversals and level extension in the general case of a wound sub-texture mapping and a selected region deformation in a three-dimensional (3D) model. This method works automatically on both regular and irregular mesh surfaces. The approach consists of mesh selection (MS), mesh leveling (ML), and mesh tagging (MT). To validate our approach, we performed experiments for synthesizing wounds on a 3D face model and on a simulated mesh.

Keywords: Wound synthesis, medical image synthesis, irregular mesh, regular mesh, mesh indexing

1. Introduction

In the visualization of medical synthesis, computer graphic rendering techniques for medical synthetic effects play a critical role in generating advanced scene appearances. To achieve enhanced visual graphics, the first procedure followed by rendering techniques involves attaching albedo textures to the region where a certain graphic is to be rendered. Mesh tagging (MT) is crucial in sub-region processing tasks, such as sub-texture mapping, shading, rendering, medical patch synthesis, and geometrical deformation. For instance, sub-texture mapping is a task that attaches additional texture maps in order to increase visual effects, such as wound synthesis, face painting, and virtual medical simulation. Prior to that, for accurate traversal, one must know the sequential order of mesh indices. In addition, it is necessary to recognize in advance the structure of mesh indices when generating a geometrical deformation to a constraint region, while considering the depths of skin layers. However, in real applications, a 3D model consists of a significant number of meshes whose indices are not in sequential order.

* Address for correspondence: Seongah Chin, Division of Multimedia Engineering, College of Engineering, Sungkyul University, 430-742, South Korea. Tel.: +82-31-467-8198; Fax: +82-31-467-8067; E-mail: solideochin@gmail.com.

In addition, most 3D models tend to be composed of irregular meshes, rather than regular meshes. Traversal in MT seems easier on a regular mesh surface than on an irregular surface. When a user wants to extend a level on a mesh, it is easier for a regular mesh than an irregular mesh to select a sub-mesh area, because a regular mesh does not cause distortions or holes, unlike an irregular mesh.

Computer graphics researchers have studied remeshing with respect to the optimization of the number of polygons in order to meet real-time rendering speeds and fidelity of visual appearance. Most remeshing algorithms tend to be based on the global parameterization of the original mesh in which a resampling of the parameter domain can be performed. In addition, techniques to find a global parameterization require considerable additional computation [1, 2]. Another approach considers a series of local modification on the mesh, which also requires a high computation cost for optimizations [3-5].

Mesh painting that is supported by most modern software is used by artists. Further, to enhance visual appearance, people require sub-region deformation and mesh painting. For this, MT is an essential task because the sub-region indices of meshes have to be known. Physiologically, wounds are complex. Sometimes, a wound structure is considerably intricate. For instance, skin depth varies slightly depending on location. For example, the skin on the cheek is deeper than the skin on the forehead. Therefore, a region must be deformed as well as rendered in order to represent particular features, such as color, depth, and components. Hence, MT is essential. However, in real applications, a 3D model consists of a large number of faces whose indices are not in sequential order. In addition, most 3D models tend to be composed of irregular meshes rather than regular meshes. Most traditional mesh generation algorithms have focused on enhancing sampling rate, given a mathematical model. The sampling rate cannot usually be changed within an application. However, in a game, for instance, wounds of various sizes can occur on any part of a character. Hence, an MT task is required when multi-texturing. This method improves our previous MT [6] that is suitable only for surfaces composed of a set of regular mesh. The proposed method is applicable to both a regular and an irregular mesh surface on which the method can tag a face without producing holes. In addition, mesh indices need to be recognized when sub-region deformation is performed. The spherical volume method is intuitive and easy to implement. However, it cannot easily deform around a nose region where the area has a high slope. In addition, our method shows an overall lower complexity $O(n)$ where n is the number of faces.

Presently, very few studies related to wound synthesis have been conducted. A procedural wound geometry generation technique is presented to allow interactive real-time operation [7]. Multi-layer based wound synthesis is generated using skin depth, such as epidermis, dermis, and sub-cutis [6]. For body parts with a rough and round shape, abrasion and burn patch synthesis models are explored [8]. In wound synthesis, wound textures must be represented efficiently. The first step is to recognize the area where the user wants to render a wound. However, in general, mesh indices are not stored in sequential order, which makes MT complicated. Hence, we validate our MT algorithm in terms of wound synthesis.

In this paper, we propose an enhanced irregular MT algorithm. This algorithm allows the user to simply select a mesh at a desired location. Then, the algorithm automatically extends the mesh region with a level index, regardless of the regularity or irregularity of the surface. Our approach follows certain tasks, including mesh selection (MS), mesh leveling (ML), and MT. Finally, we present experimental results that validate the proposed method.

2. Related works

Campen *et al.* [9] explained that, in practice, meshes often have a number of defects and flaws. This fact causes meshes to be incompatible with quality requirements. Campen *et al.* systematically analyzed the various types of defects, as well as typical application contexts. In addition, Campen *et al.* consider existing techniques to process, repair, and improve the structure, geometry, and topology of imperfect meshes. Their study included key algorithms, discussed extensions and improvements, and analyzed the respective advantage and disadvantages of meshes. Attene *et al.* [10] presented a survey paper that reviewed typical defects that make a 3D model unsuitable for key application contexts, and presented existing algorithms that can process, repair, and improve the structure, geometry, and topology of the 3D model. Their article was focused on a structured overview of mesh repairing techniques. Numerous mesh repair methods were analyzed and classified with respect to capabilities, properties, and guarantees. Guidelines indicated how to derive the identification of repairing algorithms best suited for bridging the compatibility gap between the quality provided by upstream processes and the quality required by downstream applications. Alliez *et al.* [11] surveyed recent remeshing techniques focused on graphics applications, including structured, compatible, high quality, feature, and error-driven remeshing. They also summarized open problems for future research. Sheffer *et al.* [12] presented a survey of methods for creating linear mappings between triangulation in 3D and simpler domains. The survey also debated emerging tools, including global parameterization, and inter-surface mapping. The survey is emphasized practical aspects such as time complexity and robustness. These four methods seemed to focus on mesh repairing and remeshing rather than mesh tagging that allows users to indicate specific regions and deform them.

Moon *et al.* [13] presented a task that provides real-time rendering of wound synthesis. In order for such a task to work, anisotropic diffusion was considered, rather than isotropic surface. An easy interface was offered to allow users to simulate the wound synthesis of a face. To validate the method, Moon et al. compared wound synthesis samples between traditional methods and their proposed method. Shen *et al.* [14] presented an approach for realistic irrigation visualization of a surgical wound debridement simulator. Shen *et al.* utilized image-processing techniques with images by modeling the wound cleanliness. The output is highly realistic; in addition, their method has low cost and is simple to implement. Oppenheimer *et al.* [15] presented methods for rapidly generating 3D dermatologic datasets that aim to support educational use, training simulations, and procedure planning. The wound repository was collected from captured photographic images. The 3D anatomy archived was built from MRI scans and from multiple photographic views. Stam *et al.* [16] developed a numerical 3D model to synthesize the spatial kinetics of hemoglobin and bilirubin during the healing of bruises. Similar spatial and temporal dynamics are found in both simulated and natural bruises. The different spatio-temporal dynamics of hemoglobin and bilirubin help us estimate the age of model bruises. Some previous works concerning medical synthesis seemed to represent various applications rather than wound attachments using mesh tagging.

3. Mesh tagging algorithm

The purpose of MT is to allow an arbitrary face on a surface chosen by users to be automatically traversed and extended when enlarging a selected area. First, we present MS by describing the selection of a mate face, anchor vertices, and an anchor vector. Second, we define ML, which marks a set

of neighbor meshes given a level. Last, we define MT, which involves traversing a set of meshes in a counter-clockwise manner.

In general, a 3D mesh models can be classified as either regular or an irregular mesh. As shown in Figure 1, a mesh model can be created using Delaunay triangulation (DT), which maximizes the minimum angle of all the angles of triangles in order to avoid "skinny" triangles. DT can be considered an efficient geometry model for boundary surface representation. A mesh is composed of a set of indexed vertices whose indices are stored sequentially. However, the indices of a set of meshes are located randomly, even if the meshes are neighbors in a regular or irregular mesh.

3.1 Mesh selection

Users can select any suitable face for sub-texturing, rendering, or deforming models. Hence, we design the MS task. When users want to manipulate a sub-mesh, such users need to determine the level that represents the size of an area. A higher level indicates a larger area that consists of a larger number of faces. The notation summary appears in Table 1.

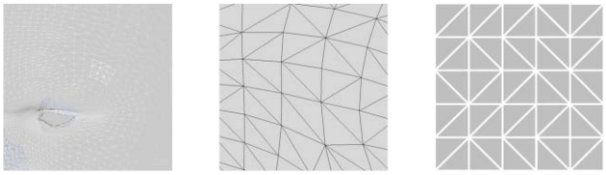

Fig. 1. 3D Mesh configuration around eye on a 3D Face model (left), irregular mesh (middle), and regular mesh (right).

Table 1
Notation Summary

Notation	Name	Section
f_p	Selected face	3.1 Mesh Selection
f_q	Mate face	3.1. Mesh Selection
$p_0^0, p_1^0, p_2^0, p_3^0$	Anchor vertices at level 0	3.1 Mesh Selection
$p'_0{}^1, p'_1{}^1, p'_2{}^1, p'_3{}^1$	Candidate anchor vertices at level 1	3.1 Mesh Selection
l	Longest diagonal edge	3.1 Mesh Selection
v_0, v_1, v_2, v_3	Anchor vectors	3.1 Mesh Selection
f_g^i	Neighbor face at level i with face index g	3.2 Mesh Leveling
B^i	Bounding box at level i	3.2 Mesh Leveling
n_d^i	Anchor line at level i with line index d	3.2 Mesh Leveling
b_d^i	Sub-bounding box	3.2 Mesh Leveling
m_j^i	Decisive variable	3.3 Mesh Tagging

3.1.1 Mate face search

Once users selcet a face, we need to determine a mate face located next to the chosen face. Let f_p be the face selected by users, and let it be composed of three vertices p_0^0, p_1^0, and p_2^0. Let l_0 be an edge made of p_0^0 and p_1^0, let l_1 be an edge that consists of p_1^0 and p_2^0, and let l_2 be an edge created by p_2^0 and p_0^0. Find l called the *longest diagonal edge* such that l is the longest edge among l_0, l_1, and l_2. We search a mate face called f_q such that the face shares l and not f_p.

3.1. 2 Neighbor face marking

Once we know f_p and the mate face f_q, certain adjacent faces located around f_p and f_q, called *neighbor faces*, must be marked as being associated with index level $i = 0$.

A neighbor face is marked as follows. A set of face vertices can be shared with adjacent meshes. We define an initial neighbor face at level i to be an adjacent face that shares a vertex with the faces at level $i - 1$. The level of the initial neighbor face is modified, considering its topological location (as described in the next sub-section).

For efficiency, we search sufficient neighbor faces with level $i + 3$ for level i because an anchor vertex at level i must be one vertex among pre-marked neighbor faces in order to ensure that the anchor vertex is found. This fact implies that we are not required to calculate the distances for all vertices in the 3D model when searching for anchor vertices.

3.1.3 Anchor vertex and anchor vector

Anchor vertices are vertices that can be used to define a mesh level for extension and for mesh traversal (that is explained later in this paper in connection with MT). For instance, at level 0, we define the anchor vertices to be p_0^0, p_1^0, p_2^0, and p_3^0. p_3^0 is the vertex in f_q that is not in f_p. As the level increases, we need to determine the anchor vertices at each given level. We define *anchor vectors* that can be used to determine the anchor vertices for the next level. Given that we know the set of initial anchor vertices at level 0, we can easily calculate four anchor vectors v_0, v_1, v_2, and v_3 using Eq. (1):

$$v_0 = \overrightarrow{p_2^0 p_0^0}, \quad v_1 = \overrightarrow{p_3^0 p_1^0}, \quad v_2 = \overrightarrow{p_0^0 p_2^0}, \quad v_3 = \overrightarrow{p_1^0 p_3^0}, \tag{1}$$

where p_0^0, p_1^0, p_2^0, and p_3^0 are anchor vertices at level 0.

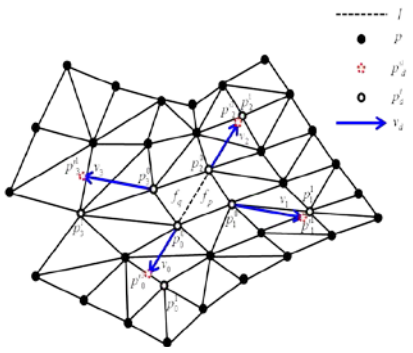

Fig. 2. Selected face f_p and mate face f_q, anchor vertices p_0^0, p_1^0, p_2^0, and p_3^0 at level *0*; extended vectors v_0, v_1, v_2, and v_3; candidate anchor vertices $p'_0{}^1$, $p'_1{}^1$, $p'_2{}^1$, and $p'_3{}^1$; and anchor vertices p_0^1, p_1^1, p_2^1, and p_3^1 at level *1*.

We define the manner in which to determine the candidate anchor vertices at level i as Eq. (2). From Eq. (2), we acquire candidate anchor vertices $p'_0{}^1$, $p'_1{}^1$, $p'_2{}^1$, and $p'_3{}^1$.

$$p_0^{i-1} + v_0 = p_0'^i, \quad p_1^{i-1} + v_1 = p_1'^i, \\ p_2^{i-1} + v_2 = p_2'^i, \quad p_3^{i-1} + v_3 = p_3'^i \tag{2}$$

where i is a level index and v_0, v_1, v_2, and v_3 are fixed anchor vectors. $p_0{}^{i-1}$, $p_1{}^{i-1}$, $p_2{}^{i-1}$, and $p_3{}^{i-1}$ are anchor vertices at level i - 1.

If we require level i, all anchor vertices at level 0 to i can be obtained in a similar manner. Finally, we determine each anchor vertex for the next level such that the nearest vertex among the initially marked neighbor vertices of meshes are introduced in ML. For instance, at level *1,* the candidate anchor vectors are computed as in Eq. (3):

$$p_0^0 + v_0 = p_0'^1, p_1^0 + v_1 = p_1'^1, p_2^0 + v_2 = p_2'^1, \quad p_3^0 + v_3 = p_3'^1 \tag{3}$$

The example shown in Figure 2 provides an idea of the graphical notions, including the notions of mate face, candidate anchor vertex, and anchor vertex.

3.2 Mesh leveling

If users can select a face on a 3D surface, the indices of a set of faces around the chosen face are not located in order. This fact makes it difficult to manage the area where the meshes must be deformed. Thus, in the ML process, we need to mark each neighbor mesh with an appropriate level that indicates its relative distance from the chosen face. An initial neighbor face at level i as described in Section 3.1.2 is denoted by $f_g{}^i$ with level index i and mesh index g. For instance, neighbor meshes at level 1 can be $f_g{}^1$ and $f_g{}^2$ at level 2, as shown in Figure 3.

3.2.1 Level bounding box and sub-bounding box

To extend level i, the area has to be large without containing a hole. However, the area composed of initial neighbor faces cannot guarantee that it does not contain a hole, as shown in Figure 4. Moreover, its shape might be distorted or skewed.

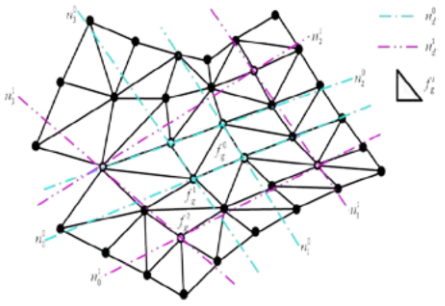

Fig. 3. Initial neighbor face marking $f_g{}^1$ at level *1* and $f_g{}^2$ at level 2 with anchor lines $n_d{}^i$, where i is the level index and $0 \le d \le 3$.

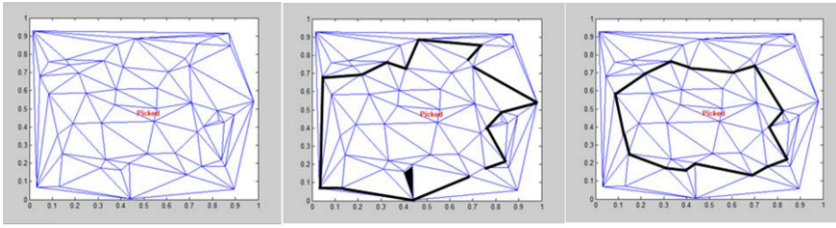

Fig. 4. Initial mesh surface (left), method in [7] (middle), our method (right).

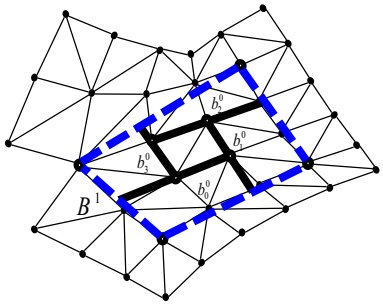

Fig. 5. Level bounding box B^1 at level *1* and sub- bounding box $b_d{}^0$ with $0 \le d \le 3$.

To solve this problem, we perform re-leveling of the initial neighbor faces to the finally used neighbor meshes by defining a *bounding box* denoted by B^i at level *i*. Neighbor mesh leveling is finalized through the bounding box, as explained in Section 3.3.

The bounding box B^i at level *i* is defined by four line equations called *anchor lines* denoted by $n_d{}^i$ at level *i* with line index *d* computed by four anchor vertices at level *i*. Anchor lines can be acquired as $n_0{}^i = p_1{}^i - p_0{}^i$, $n_1{}^i = p_2{}^i - p_1{}^i$, $n_2{}^i = p_3{}^i - p_2{}^i$, $n_3{}^i = p_0{}^i - p_3{}^i$ at level *i*. In addition, we define a *sub-bounding box* denoted by $b_d{}^i$ with level index $i > 0$ and $0 \le d \le 3$ that is a partition of bounding box B^i. The sub-bounding box is used to determine the level and the ordering meshes in detail in Section 3.3. If level index i = 0, we do not apply sub-bounding boxes, but anchor lines.

Namely, $B^{i+1} = \{b_0{}^i, b_1{}^i, b_2{}^i, b_3{}^i\}$ and $b_d{}^i = \{f_g{}^{i+1}\}$ with mesh index *g*. Here, $b_0{}^i$ is defined by $n_0{}^{i+1}$, $n_1{}^i$, $n_0{}^i$, and $n_3{}^{i+1}$. $b_1{}^i$ is defined by $n_0{}^{i+1}$, $n_1{}^{i+1}$, $n_2{}^i$, $n_1{}^i$. $b_2{}^i$ is defined by $n_2{}^i$, $n_1{}^{i+1}$, $n_2{}^{i+1}$, and $n_3{}^i$. $b_3{}^i$ is defined by $n_0{}^i$, $n_3{}^i$, $n_2{}^{i+1}$, and $n_3{}^{i+1}$. For instance, bounding box B^1 and sub-bounding boxes $b_0{}^0$, $b_1{}^0$, $b_2{}^0$, and $b_3{}^0$ at level *1* are shown in Figure 5.

3.3 Mesh Tagging

In this section, we perform re-leveling to verify initially-marked neighbor meshes, and to sort the neighbor meshes for traversal.

3.3.1 Re-leveling of neighbor face

Let us start with initial neighbor faces for level i. As mentioned previously, we calculated faces in advance to a maximum of level $i + 3$. To finalize re-leveling of neighbor faces, we need to know to which $b_d{}^i$ an initial neighbor face $f_g{}^{i+1}$, $f_g{}^{i+2}$, or $f_g{}^{i+3}$ belongs. If a neighbor face belongs to one of $b_d{}^i$, we finalize the neighbor face by $f_g{}^{i+1}$. For instance, $f_g{}^2$, which initially was classified into a face of level 2 (as shown in Figure 3), belongs to $b_0{}^0$, as shown in Figure 5. We re-level $f_g{}^2$ as $f_g{}^1$. Whether a

neighbor face f_g^{i+1} falls into b_d^i is determined as follows. As mentioned, b_d^i is defined by four boundary linear equations. We employ two passes in order to avoid distortion and skew when extending levels: one pass is marked using a center of gravity c_g^{i+1}, and the other pass is marked using the vertex of a face.

For example, b_0^i is defined by n_0^{i+1}, n_1^i, n_0^i, and n_3^{i+1} managed counter-clockwise. Let m_j^i $(0 \le j \le 3)$ be a *decisive variable* that holds an output value that is a positive integer, a negative integer, or zero associated with each linear equation. In addition, m_j^i takes the center of gravity of a neighbor face as input.

At the first pass, we determine whether the variables satisfy all $m_j^i \ge 0$. If so, the neighbor face belongs to b_d^i. We also consider the second pass if the center of gravity does not belong to the mesh, because as the level increases, a skewed area might spread. We take three vertices that consist of a neighbor face f_g^{i+1} as inputs to four linear equations similarly to using the center of gravity.

If one of three vertices satisfies $m_j^i \ge 0$, the face is accepted. In the left image of Figure 6, we see that f_{13}^1 and f_{14}^1 are not contained by b_2^0 and b_3^0, respectively, when the centers of gravity are used. Finally, f_{13}^1 belongs to b_2^0 and f_{14}^1 is in b_3^0 because one of the three vertices is included in the sub-bounding box that can pass through the second pass shown in the right image of Figure 6.

3.3.2 Mesh sorting

Once we complete re-leveling neighbor meshes, we need to sort them because indices have to be in order. That is, we obtain a set of neighbor meshes that belong to an associated sub-bounding box.

To solve this problem, the center of gravity c_g^{i+1} of each neighbor face is used to mark its order. For instance, if a set of meshes is classified into b_0^i, it has to be left-ordered. If the set is in b_1^i, the meshes need to be bottom-ordered. If the set is in b_2^i, the meshes need to be right-ordered. If the set is in b_3^i, the meshes need to be top-ordered. Figure 7 illustrates the ordering procedure. The left image of Figure 7 shows unordered mesh indices, and the right image of Figure 7 shows mesh indices ordered counter-clockwise.

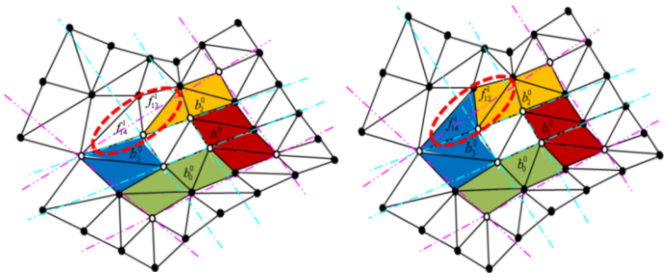

Fig. 6. Markings for two passes. The left uses a center of gravity, and the right uses one of the vertices.

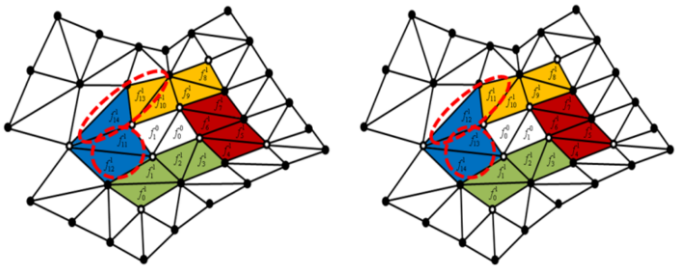

Fig. 7. Mesh sorting.

3.3.3. Validation of our propositions

Our propositions state that the method does not contain any hole after MT. Moreover, our propositions state that the tagged faces must be ordered counter-clockwise. We show proof by mathematical induction as follows:

[Proposition 1] MT does not contain any hole.

Proof) Let i be a level index. An initial neighbor mesh at level i is denoted by f_g^i with level index i and mesh index g. We follow the notation summary listed in Table 1.

1) If $i = 0$, we show that there is no hole when our algorithm is performed.

This is trivial because, if we select face $f_g^0 = f_p$, our algorithm, as defined in Section 2.1.1, can find mate face f_q such that the face shares l, where l *is* the longest diagonal edge among l_0, l_1, and l_2 and not f_p. Selected meshes do not have any hole.

2) If $i = k$, we assume that our proposition is satisfied. In other words, our algorithm can mark all meshes denoted by f_g^k without any holes at level $i = k$.

3) If $i = k + 1$, we need to show that our algorithm can find all meshes that do not contain any holes. According to our assumption 2), we realize that our algorithm can find all meshes that do not include any holes at level k. This fact implies that the anchor vertices p_0^k, p_1^k, p_2^k, and p_3^k at level k are successfully found as defined in Section 3.1.3. In addition, anchor lines n_d^k, where i is the level index and $0 \le d \le 3$, can be found such that the bounding box $B^k = \{b_0^{k-1}, b_1^{k-1}, b_2^{k-1}, b_3^{k-1}\}$ and $b_d^{k-1} = \{f_g^k\}$ with mesh index g at level $k > 0$ are sought as defined in Section 3.2.1. By extending level to $k + 1$, we can finally find p_0^{k+1}, p_1^{k+1}, p_2^{k+1}, and p_3^{k+1} as well as bounding box $B^{k+1} = \{b_0^k, b_1^k, b_2^k, b_3^k\}$ and $b_d^k = \{f_g^{k+1}\}$ with face index g at level $k + 1$. Hence, the algorithm marks all faces at level k + 1 without any holes.

[Proposition 2] Tagged meshes must be ordered counter-clockwise.

Proof) Let i be a level index. An initial neighbor face at level i is denoted by f_g^i with level index i and face index g. We follow the notation summary listed in Table 1.

1) If $i = 0$, we show that tagged meshes must be ordered counter-clockwise.

This is trivial because, if we select face $f_g^0 = f_p$, our algorithm, as defined in Section 3.1.1, can find mate face f_q such that the face shares l, where l *is* the longest diagonal edge among l_0, l_1, and l_2 and not f_p. In addition, the bounding box B^0 at level 0 is only composed of anchor lines $n_d^{i=0}$ at level $i = 0$ with line index $0 \le d \le 3$ computed by four anchor vertices at level *0*. Therefore, the ordering of two tagged faces holds.

2) If $i = k$, we assume that our algorithms can keep all meshes ordered counter-clockwise at level k. Each f_g^k at level $i = k$ is ordered counter-clockwise.

3) If $i = k+1$, we show that tagged meshes are still ordered. By extending the level to k + 1, we follow the algorithm in Sections 3.3.1 and 3.3.2. At level k, tagged meshes are ordered. Hence, we can acquire anchor vertices at level k + 1 by adding anchor vector to anchor vertices at level k as described in Section 3.1.3. Then, we can find the bounding box $B^{k+1} = \{b_0^k, b_1^k, b_2^k, b_3^k\}$ and $b_d^k = \{f_g^{k+1}\}$ with face index g at level $k + 1$. Hence, the algorithm marks all faces at level $k + 1$ as well. Then, we can compute m_j^i $(0 \le j \le 3)$ as a decisive variable that determines the relative locations of the center of gravity of a neighbor face for level $k + 1$ as explained in Section 3.3.1. Finally, mesh shorting is done by Section 3.3.2.

3.3.4 Complexity analysis

We performed complexity analysis including MS, ML, and MT with respect to the number of meshes n. MS needs to refer to all mesh indices when users select a mesh. Thus, MS takes $O(n)$. ML visits certain faces at every level in order to search four anchor vertices. Therefore, an average number of levels can computed as $\frac{1}{2}(\frac{\sqrt{n}}{2} + 1)$ that becomes $O(\sqrt{n})$. When leveling is completed, ML excludes registered meshes such that the number of faces at level l becomes16(l - 1) if level > 0 and 2 if level = 0. Thus, we have $O(l) = O(\sqrt{n})$. Finally, ML becomes $O(n)$. MT performs tagging at every level. Hence, the number of faces at every level needs to be multiplied by the number of levels. Consequently, MT takes $O(n) \approx O(\sqrt{n}) \times O(\sqrt{n})$. Lastly, MS is performed on faces that belong to given a level. Hence, MS also becomes $O(n) \approx O(\sqrt{n}) \times O(\sqrt{n})$. To summarize, our algorithms show approximately the same or lower complexity compared to the Euclidean distance $O(n)$ or the geodesic distance $O(n^2 log n)$, respectively; in addition, our methods provide mesh sorting.

4. Experiments

Experiments were performed with Intel Core™ i5 CPU 750 @ 2.67 GHz, 2.99 RAM, NVIDIA GeForce GTS 250(VGA) PC, and with Windows XP Professional.

4.1 Simulation on an arbitrary surface

To validate the proposed method, we first verified the algorithm on an irregular mesh model that was generated using the Delaunay triangle. A mesh model that contains fifty vertices was created by MATLAB. We randomly selected any face f_p from the mesh model. We performed approximately 100 trials. Through experiments, we chose initial neighbor faces to be level $i + 3$. The experiments show that the algorithm marked the area without showing a hole or skewed distortion. In addition, our algorithms support tagging with regular boundaries. More robust proof has been shown by mathematical induction in Section 3.3.3.

4.2 Wound synthesis on 3D face model

We also used our proposed algorithm to realize wound synthesis on a 3D face model composed of 7,344 vertices and 14,434 faces. In general, designers work manipulating 3D models for texture mapping, geometry modeling, and animation by looking at a display monitor. In our methods, we assume surroundings where users work on screen monitors that return 2D (x, y) screen coordinates when selecting a face, while avoiding 3D distortion caused by 2D projection. In fact, such as scenario is not critical because users are asked to select only one face in our method.

In the experiment, we observed that the proposed method works well for both regular and irregular meshes. For instance, faces seem to be irregular around the nose and chin as shown in Figure 8. Screenshots for wound synthesis with level extension (0-7 levels) are shown in the left pictures in Figure 9. Some large wounds appear at the right pictures of Figure 9. Details of techniques beyond the mash tagging task, including segmentation, augmentation, and multi-layered skin configuration, were used in our previous research [7, 8].

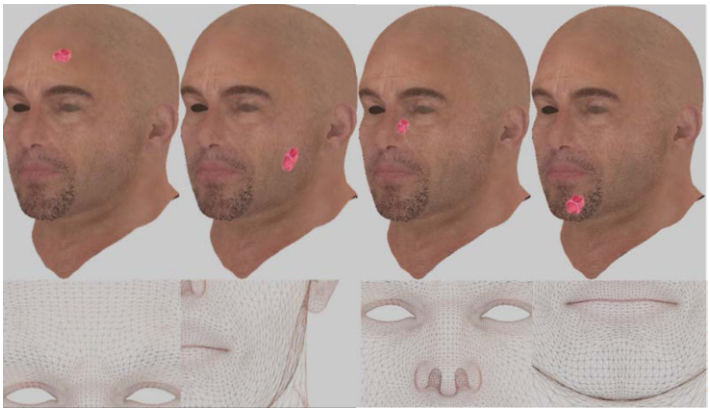

Fig. 8. Synthesized wounds realized on irregular meshes on forehead, cheek, nose, and chin for a 3D face model.

5. Conclusions and discussions

MT is an essential task wherein one must perform sub-region deformation, rendering, texture mapping, and so on. In addition, wound synthesis has not attracted much attention from researchers in computing. Wounds appear sufficiently detailed. For instance, skin depth is truly dependent on the location of the skin. Depth near a cheek is higher than the skin on a forehead. Therefore, MT is necessary in order to facilitate a deformation. However, in real applications, mesh indices are not in sequential order.

In this paper, we presented a novel MT algorithm that utilizes a task for mesh traversals and a level extension in the general case of wound sub-texture mapping or selected region deformation in a 3D model. This method works automatically on both regular and irregular mesh surfaces because it requires initial user input. The approach consists of MS, ML, and MT. To validate our approach, we showed mathematical induction to prove that our propositions are satisfied, and we performed experiments to synthesize wounds on a 3D face model and on a simulated mesh. In addition, we performed a

complexity analysis of our algorithm with respect to MS, ML, and MT. An overall lower complexity $O(n)$ was shown, where n is the number of meshes.

To render wound synthesis, we also performed a simple texture model as shown in the left pictures in Figure 9, and a CG (NVidia) shader model that enhanced the fidelity of synthesis results rendered by the improved Oren-Nayar model [17] as shown in the right pictures in Figure 9. For our future research, we need to consider adding weight to the anchor vector to enhance speed. Then, we must initially register sufficient neighbor meshes that require additional time to retrieve. Furthermore, we cannot assure that geometrical shapes can be sustainable compared to the current method. Moreover, we must investigate how to decide an optimal weight. We do not intend our method to be constrained by the Delaunay method. We have performed our experiments on Delaunay because most 3D authoring tools, such as 3DS MAX and MAYA, provide the Delaunay method. For future work, we will verify that our method is also fitted with other mesh generations.

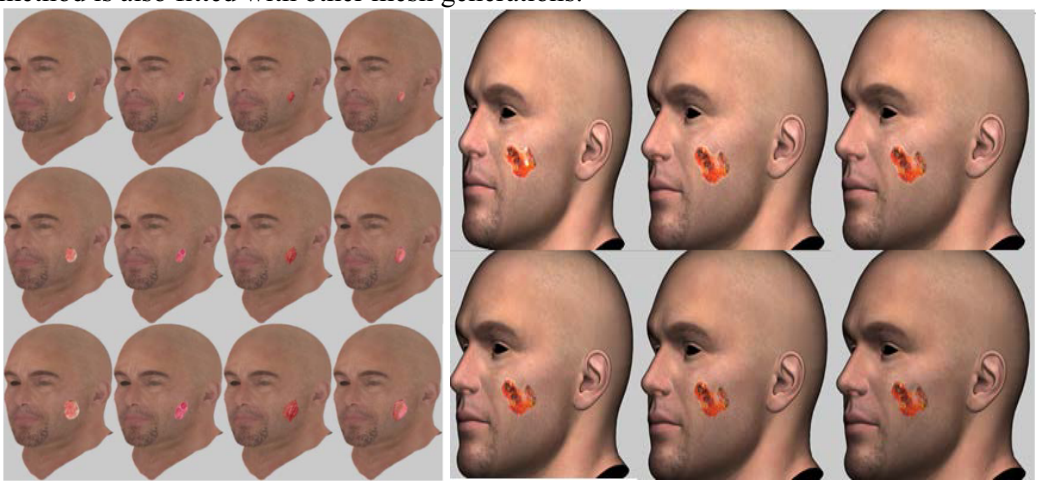

Fig. 9. Various types of synthesized wounds with extended levels showing a low level (first row), a medium level (second row), and a high level (third row) in the left figure. Synthesized wounds rendered by improved Oren-Nayar model with shine coefficients 1.8, 1.5, and 1.2 (first row) and 0.9, 0.6, and 0.3 (second row) in the right figure.

Acknowledgment

This research was supported by Basic Science Research Program through the NRF funded by the Ministry of Education (2010-0008673).

References

[1] P. Alliez, M. Meyer and M. Desbrun, Interactive geometry remeshing, ACM Transactions on Graphics, Special issue for SIGGRAPH Conference **21** (2002), 347–354.
[2] K. Hormann, U. Labsik and G. Greiner, Remeshing triangulated surfaces with optimal parameterizations, Computer-Aided Design **33** (2001), 779–788.
[3] P.J. Frey and H. Borouchaki, Geometric surface mesh optimization, Computing and Visualization in Science **1** (1998), 113–121.
[4] A. Rassineux, P. Villon, J.-M. Savignat and O. Stab, Surface remeshing by local Hermite diffuse interpolation, International Journal for Numerical Methods in Engineering **49** (2000), 31–49.
[5] G. Turk, Re-tiling polygonal surfaces, Computer Graphics SIGGRAPH '92 Proceedings **26** (1992), 55–64.

[6] C. Lee, S. Lee and S. Chin, Multi-layer structural wound synthesis on 3D face, Journal of Visualization and Computer Animation **22** (2011), 177–185.
[7] R. Aras, Y. Shen and J. Li, Procedural wound geometry and blood flow generation for medical training simulators, SPIE Proceedings, 8316, Medical Imaging 2012, San Diego, California, 2012.
[8] S. Chin, S. Lee and S. Kim, Abrasion and burn patch synthesis for a rough and round shaped body part, International Journal of Digital Contents Technology and Applications **6** (2012), 303–309.
[9] M. Campen, M. Attene and L. Kobbelt, A Practical Guide to Polygon Mesh Repairing, Eurographics 2012 tutorial, 2012.
[10] M. Attene, M. Campen and L. Kobbelt, Polygon mesh repairing: An application perspective, ACM Computing Surveys **45** (2013), 1–33.
[11] P. Alliez, G. Ucelli, C. Gotsman and M. Attene, Recent advances in remeshing of surfaces, Shape Analysis and Structuring Mathematics and Visualization (2008), 53–82.
[12] A. Sheffer, E. Praun and K. Rose, Mesh parameterization methods and their applications, Foundations and Trends in Computer Graphics and Vision **2** (2006), 105–171.
[13] S. Moon, S. Lee, S. Kim, H. Hyun and S. Chin, Cinematic wound synthesis optimized for real-time gameplay, International Journal of Software Engineering and its Applications **7** (2013), 221–228.
[14] Y. Shen, J. Seevinck and E. Baydogan, Realistic irrigation visualization in a surgical wound debridement simulator, Stud Health Technol Inform **119** (2006), 512–514.
[15] P. Oppenheimer, J. Berkley and S. Weghorst, Virtual image grafting: Image based generation and visualization of virtual skin defects, Studies in Health Technoly and Informatics **85** (2002), 321–327.
[16] B. Stam, M.J. Gemert, T.G. Leeuwen and M.C.G. Aalders, 3D finite compartment modeling of formation and healing of bruises may identify methods for age determination of bruises, Medical and Biological Engineering and Computing **48** (2010), 911–921.
[17] Y. Gotanda, Beyond a simple physically based Blinn-Phong model in real-time, SIGGRAPH 2012 course, 2012.