J. KSIAM Vol., No., 1–11,

PAPER TEMPLATES FOR TRIANGULATED SURFACES

CHOHONG MIN

MATHEMATICS DEPARTMENT, EWHA WOMANS UNIVERSITY, SEOUL, KOREA 120-750
E-mail address: chohong@ewha.ac.kr

ABSTRACT. We introduce an algorithm that acutomatically generates paper templates of a triangulated surface. The surface can be built by cutting, folding, and pasting the paper templates. The algorithm is branched to two strategies : one is to select the longest neighboring edge among many choices, and the other is to select the largest neighboring triangle. Three surfaces, whose triangulation sizes widely range, are successfully built by the algorithm. The two strategies are empirically evaluated in building the surfaces with respect to paper consumption, a measure of cost efficiency, and boundary length, a measure of speed efficiency. Strategy 1 performs in most cases better than the other one with respect to boundary length, but sometimes wins and sometimes loses with respect to paper consumption.

1. INTRODUCTION

In many cases, a surface is represented as a triangulation, i.e. a set of non-overlapping triangles whose union is hole free [13]. The triangulation is efficient in storing and maneuvering, since it deals with only the triangle vertices. In some cases, surface is implicitly represented by a three dimensional function as the zero level set of the function. The implicit representation has many advantages in calculating surface normal, curvature, and other surface geometries. The level set function is sampled on a grid, and stored and manipulated by a three dimensional array. In the implicit representation, surface resolution can be easily controlled; surface is described with more details and accuracy with finer grid. There have been developed successful algorithms such as Marching Cubes [7, 9] and Marching Tetrahedrons [2, 8] that can transform the level set representation to triangulation. In other cases, a surface is given just with densely sampled points on it. It often happens, when real objects such as sculptures and architectures are scanned. There also have been many successful algorithms converting the dense points to triangulation. There have also been developed successful algorithms converting the dense points to a triangulation such as Delanay triangulation [1, 6], level-set method [12, 11], and radial-basis-function interpolation [3, 4].

Among the surface representations, the triangulation represents a surface by triangles which can be printed out in flat material such as paper and metal sheets. There are fore-mentioned

2010 *Mathematics Subject Classification.* 65D18.

Key words and phrases. paper templates, simplex, computational geometry, triangulation.

This work was supported by Priority Research Centers Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2010-0028298).

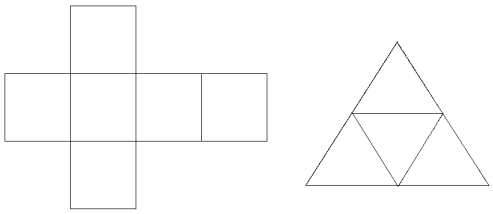

FIGURE 1. Paper templates for cube and tetrahedron

conversion algorithms from the other representations to the triangulation. With these reasons, we assume surfaces to be given as triangulations throughout this paper.

Consider the paper templates in fig 1 of which cube and tetrahedron can be made by cutting, folding and pasting the templates. In this paper, like-wisely we introduce an algorithm that generates paper templates for triangulated surfaces, enabling their manufacturing by cutting, folding, and pasting.

2. ALGORITHM

Assume a triangulated surface that consists of triangles T_1, T_2, $\cdots$, T_N. In this section, we present an algorithm that generates paper templates for the surface, and enables building the surface by cutting, folding, and pasting the templates. By a paper template, we mean a set of triangles that are connected to each other along their shared edges. For clarity, we mathematically define it below.

Definition 1. A set of triangles in $\mathbb{R}^2$ is called *triangle template*, if each triangle shares at least one edge with another one in the set, and their union has one connected component.

We assume paper sheets of uniform dimension $w \times h$ on which paper templates are printed. The dimension is assumed to be large enough so that each triangle in the triangulation can be printed in a paper sheet. A trivial algorithm is to consume a paper sheet for each triangle. Then each edge is cut once on each of its two adjacent triangles, and the two triangles are pasted together along the edge in building the three dimensional surface. When the two triangles were printed together, the work of twice cutting and pasting could be alleviated by one folding. We are aimed to construct an algorithm to meet two objects: one is to decrease the length of the edges that need cutting and pasting, to be effective in speed, and the other is to consume less paper sheets, to be effective in cost.

Our algorithm starts with a triangle, for example T_1, from the triangulation and form a paper template with it as described in section 2.2. It has three neighboring triangles by which the template can be extended. Each neighbor is tested if the enlarged template does not exceed the rectangle of dimension $w \times h$ by the method in section 2.1. If none passes the test, the template is done and printed out on a paper sheet. Among the passed ones, one is selected and added to the template. The process of selection and addling goes on until the paper sheet can contain the template. In the addition process, each neighboring triangle should increase the

template within the rectangle dimension, checked by the method in 2.1, and should not overlap the existing triangles, checked by the method in 2.2. In the selection process, we suggest the following two strategies that are very simple to implement.

Strategy 1 : select the neighboring triangle with the longest edge shared with the template.

Strategy 2 : select the neighboring triangle with the largest area.

Strategy #1 is divided to reduce the length of edges needing cut-and-paste pursuing the speed efficiency, and strategy #2 is devised in the sense that smaller triangles would more easily intervene triangles. Algorithm 1 shows the overall process.

Algorithm 1 Main algorithm

take a triangle from the triangulation and queue it to a list L_{todo}
while L_{todo} is not empty **do**
  repeat
    dequeue T from L_{todo}
  until T was not printed.
  form a paper template $S = \{T\}$.
  mark that T is printed.
  repeat
    prepare a list $L^{ngbd} = \emptyset$.
    for each neighboring triangle T_{ngbd} of S **do**
      $b_{bounding}$: $S \cup T_{ngbd}$ can be printed within $w \times h$. (section 2.1)
      $b_{overlapping}$: T_{ngbd} does not overlap the triangles of S. (section 2.3)
      if T_{ngbd} is not printed and $b_{bounding}$ and $b_{overlapping}$ **then**
        queue T_{ngbd} to L_{todo}
        queue T_{ngbd} to L_{pass}
      end if
    end for
    select one T_{ngbd} among L_{ngbd} by the strategy.
    mark that T_{ngbd} is printed.
    add T_{ngbd} to S (section 2.2)
  until L_{ngbd} is empty
  print out the template S
end while

In the algorithm, the queue operation adds an element to a list, and the dequeue operation takes out from the list the element added in the earliest [5]. Unselected neighboring triangles are queued to the list L_{todo}, and a new paper template begins with the dequeued triangle from the list, so that the new template is connected to the previous one of the templates. Thus the printed paper templates are cut and folded, and then sequentially built: the second template is

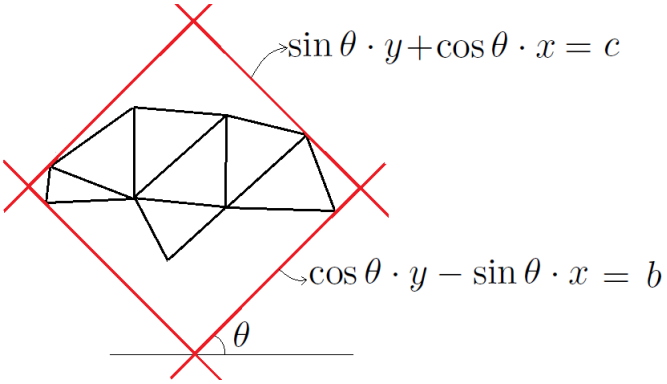

FIGURE 2. The bounding rectangle slanted with angle θ : two parallel lines in the diagonal direction satisfy $\cos\theta \cdot y - \sin\theta \cdot x = b$, and the other two satisfy $\sin\theta \cdot y + \cos\theta \cdot x = c$.

pasted to the first one, and the third one is pasted to the union of the first and the second, and this sequential building goes on. The details of the main algorithm are described as follows.

2.1. **Bounding Test.** In this section, we introduce a test to check whether a paper template can be placed inside a rectangle of dimension $w \times h$ or not. Fig 2 depicts a bounding rectangle slanted with angle θ to the x-axis. A pair of parallel lines satisfies equation $\cos\theta \cdot y - \sin\theta \cdot x = b$ and the other pair does $\sin\theta \cdot y + \cos\theta \cdot x = c$. The template is inside the rectangle if and only if all of its vertices are so, thus we measure the ranges of b and c for the vertices.

$$\begin{aligned} \text{convex\_hull}\,\{b = \cos\theta \cdot y - \sin\theta \cdot x |\,(x,y) \text{ is a vertice of template}\} &= [b_{min}, b_{max}] \\ \text{convex\_hull}\,\{c = \sin\theta \cdot y + \cos\theta \cdot x |\,(x,y) \text{ is a vertice of template}\} &= [c_{min}, c_{max}] \end{aligned}$$

The dimension of the bounding rectangle is $(b_{max} - b_{min}) \times (c_{max} - c_{min})$. The template can be placed inside the rectangle with slanted angle θ if and only if $b_{max} - b_{min} \leq w$ and $c_{max} - c_{min} \leq h$, or $b_{max} - b_{min} \leq h$ and $c_{max} - c_{min} \leq w$. This test needs be done for each angle $\theta \in \left[0, \frac{\pi}{4}\right]$. In all our examples, we tried 20 uniformly sampled θ's in the interval, which practically worked well.

2.2. **Embedding Formula.** In this section, we introduce a formula embedding a triangle $T \subset \mathbb{R}^3$ into a paper template in $\mathbb{R}^2$. Let T have its neighbor $T_{ngbd} \subset \mathbb{R}^3$ along edge $\overline{P_1P_2}$, and T_{ngbd} be already embedded as $\tilde{T}_{ngbd} \subset \mathbb{R}^2$, see figure 3 for the setting. let the other vertice of T be $P_3 \in \mathbb{R}^3$ and that of T_{ngbd} be $P_4 \in \mathbb{R}^3$, accordingly $T = \Delta P_1P_2P_3$ and $T_{ngbd} = \Delta P_1P_2P_4$, and let $\tilde{T}_{ngbd} = \Delta\tilde{P}_1\tilde{P}_2\tilde{P}_4$. To preserve the common edge, the vertices P_1 and P_2 should be mapped to $\tilde{P}_1$ and $\tilde{P}_2$. The other vertice P_3 is mapped to $\tilde{P}_3$ such that $T = \Delta P_1P_2P_3$ and $\tilde{T} = \Delta\tilde{P}_1\tilde{P}_2\tilde{P}_3$ are equivalent.

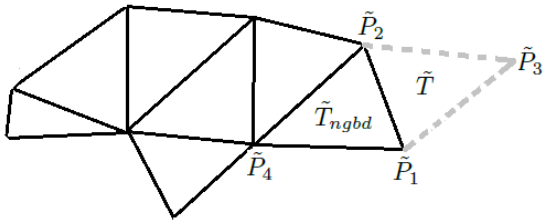

FIGURE 3. Embedding a triangle $T = \Delta P_1P_2P_3 \subset \mathbb{R}^3$ into a paper template: $T_{ngbd} = \Delta P_1P_2P_4 \subset \mathbb{R}^3$ is its neighborhood and embedded as $\tilde{T}_{ngbd} = \Delta\tilde{P}_1\tilde{P}_2\tilde{P}_4$ into the paper template. $\tilde{P}_3$ is decided from the equivalence of T and $\tilde{T}$.

$$\begin{aligned} d &= \left(l_2^2 + l_3^2 - l_1^2\right) / \left(2l_3\right) \\ h &= \sqrt{l_2^2 - d^2} \\ \tilde{P}_3 &= \frac{d}{l_3}\tilde{P}_2 + \left(1 - \frac{d}{l_3}\right)\tilde{P}_1 \pm \frac{h}{l_3}\,(0,0,1) \times \left(\tilde{P}_2 - \tilde{P}_1\right) \end{aligned}$$

Here l_i denotes the length of the edge opposite P_i in triangle T for each $i = 1, 2, 3$. The cross product $(0,0,1) \times \left(\tilde{P}_2 - \tilde{P}_1\right)$ is evaluated as a vector in $\mathbb{R}^2$. Since $\tilde{P}_3$ and $\tilde{P}_4$ are placed in the opposite sides to edge $\overline{\tilde{P}_1\tilde{P}_2}$, the sign is taken to be positive if $\left(\tilde{P}_4 - \tilde{P}_1\right) \cdot (0,0,1) \times \left(\tilde{P}_2 - \tilde{P}_1\right) < 0$, and negative otherwise.

In the case when the triangle T is the first one embedded into the paper template, we arbitrarily set $\tilde{P}_1 = (0,0)$ and $\tilde{P}_2 = (l_3, 0)$, and the other vertice $\tilde{P}_3$ is set by the above formula.

2.3. **Overlapping Test.** In the previous section, a triangle $T \subset \mathbb{R}^3$ is embedded into a paper template. The embedded triangle $\tilde{T} \subset \mathbb{R}^2$ may overlap the existing triangles of the paper template, in which case the embedding is not possible. In this section, we introduce a test to check whether two triangles in $\mathbb{R}^2$ overlap or not. The intersection of $\tilde{T}$ and triangles of the template is then tested by applying the algorithm to the pair of $\tilde{T}$ and each triangle in the template. Let two triangles $\Delta\tilde{P}_1\tilde{P}_2\tilde{P}_3$ and $\Delta\tilde{Q}_1\tilde{Q}_2\tilde{Q}_3$ be given in $\mathbb{R}^2$. We denote their coordinates by $\tilde{P}_i = (x_i, y_i)$ and $\tilde{Q}_i = (a_i, b_i)$ for $i = 1, 2, 3$. The two triangles may overlap in two ways: one triangle is included in the other, or a pair of their edges intersect. The inclusion can be tested as follows.

$$\begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} \lambda_{i1} \\ \lambda i_2 \\ \lambda_{i3} \end{bmatrix} = \begin{bmatrix} 1 \\ a_i \\ b_i \end{bmatrix}$$

	Str #1	Str #2
number of sheets	4	3
boundary length	28.02	20.63

TABLE 1. Evaluation of the two strategies in building the sphere

$\Delta\tilde{Q}_1\tilde{Q}_2\tilde{Q}_3 \subsetneqq \Delta\tilde{P}_1\tilde{P}_2\tilde{P}_3$ if and only if $\lambda_{ij} > 0$ for each i and j . The other direction $\Delta\tilde{P}_1\tilde{P}_2\tilde{P}_3 \subsetneqq \Delta\tilde{Q}_1\tilde{Q}_2\tilde{Q}_3$ can be similarly checked. Two edges $\overline{\tilde{P}_1\tilde{P}_2}$ and $\overline{\tilde{Q}_1\tilde{Q}_2}$ intersect each other if and only if

$$\begin{cases} [(y_1-y_2)(a_1-x_1)-(x_1-x_2)(b_1-y_1)]\cdot[(y_1-y_2)(a_2-x_1)-(x_1-x_2)(b_2-y_1)]<0 \\ [(b_1-b_2)(x_1-a_1)-(a_1-a_2)(y_1-b_1)]\cdot[(x_1-x_2)(x_2-a_1)-(a_1-a_2)(y_2-b_1)]<0 \end{cases}.$$

Similarly tested are the other eight pairs: $\overline{\tilde{P}_1\tilde{P}_2}$ and $\overline{\tilde{Q}_2\tilde{Q}_3}$, $\overline{\tilde{P}_1\tilde{P}_2}$ and $\overline{\tilde{Q}_3\tilde{Q}_1}$, $\overline{\tilde{P}_2\tilde{P}_3}$ and $\overline{\tilde{Q}_1\tilde{Q}_2}$, and et cetera.

3. EXAMPLES

Three surfaces are tested: sphere, bunny-shaped surface, and dinosaur-shaped surface. Their triangulation sizes widely range from tens of triangles to thousands. Their level set functions are given either by formula or by sampled data on grids. The surfaces are triangulated by isosurfacing algorithms. All of their paper templates were printed in A4 paper and actually built. To do so, the ratio of the dimensions of paper sheets is taken 1.5, the ratio of the A4 dimensions. In each example, the two strategies are evaluated with respect to the number of consumed paper and the length of edges needing cut-and-paste. There is a randomness in choosing the initial triangle in the main algorithm, so several ones are tried as the initial one to see if the evaluation depends on the randomness.

3.1. **Sphere.** The level function $\sqrt{x^2+y^2+z^2}-1$ is sampled on a domain $[-1.5, 1.5]^3$ with uniform grid 5^3. Its isosurface, which is a sphere, is then triangulated by Marching Cubes algorithm [9]. The triangulation has 56 triangles and 30 vertices. Its paper templates are embedded in paper sheets of dimension 3×2, and then printed in A4 paper and actually built, as seen in figure 6. Figure 4 shows the paper templates generated by strategy 1, and figure 5 by strategy 2. Table 1 shows that the second strategy is more efficient in cost and speed: it consumes fewer paper sheets and the boundary length, needing the time consuming cut-and-paste, is shorter. Sphere is convex everywhere, and the evaluation may be partial. In the following examples, we take more general surfaces and try more thorough evaluations.

3.2. **Bunny.** The Stanford bunny is one of the most commonly used test models in computer graphics and computer modeling [10]. The surface consists of 69451 triangles, which are too many in its actual building. Its signed distance function is numerically calculated on a grid $150\times150\times120$ taking computational domain as $[-0.094690, 0.061009]\times[0.032987, 0.187321]\times$

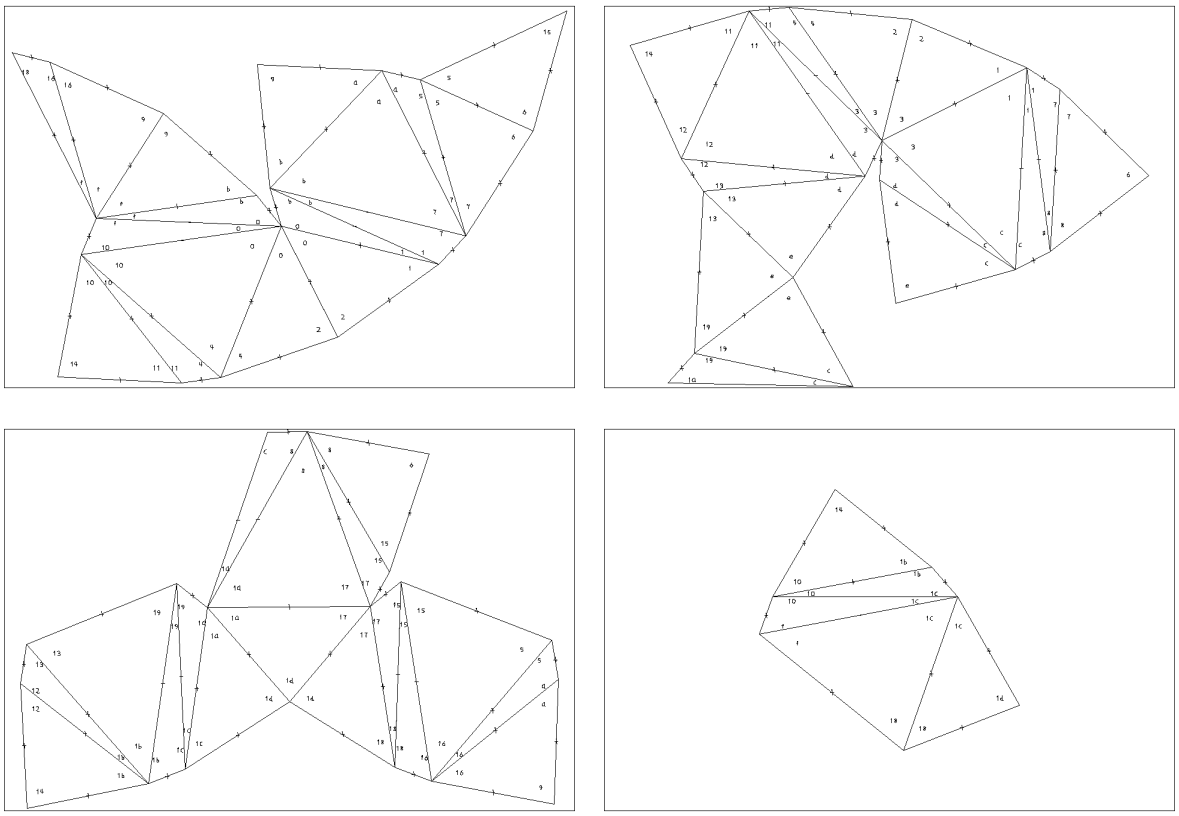

FIGURE 4. Paper templates of the sphere generated by strategy 1 : the vertices are indexed in hexadecimal , reducing printing space than decimal, and each edge is marked $+$(or $-$) if it goes upward (or downward) in the three dimensional folding.

$[-0.061874, 0.058800]$ [12]. The function is then resampled on a coarser grid $18\times18\times15$. The function values in the raster-scan order can be downloaded at www.math.ewha.ac.kr/˜chohong/bunny.txt. Its isosurface is triangulated by Marching Cubes [9]. Now, the triangulation size is reduced to 1628 triangles and 816 vertices. Its paper templates are embedded in paper sheets of dimension 0.0681×0.0454. Figure 6 shows the actual building of the bunny surface. Table 2 evaluates the two strategies applied to building the bunny surface. The algorithm was tried with six different initial triangles. Strategy 1 generated shorter boundary edges in all the trials, and consumed fewer paper sheets five times out of six.

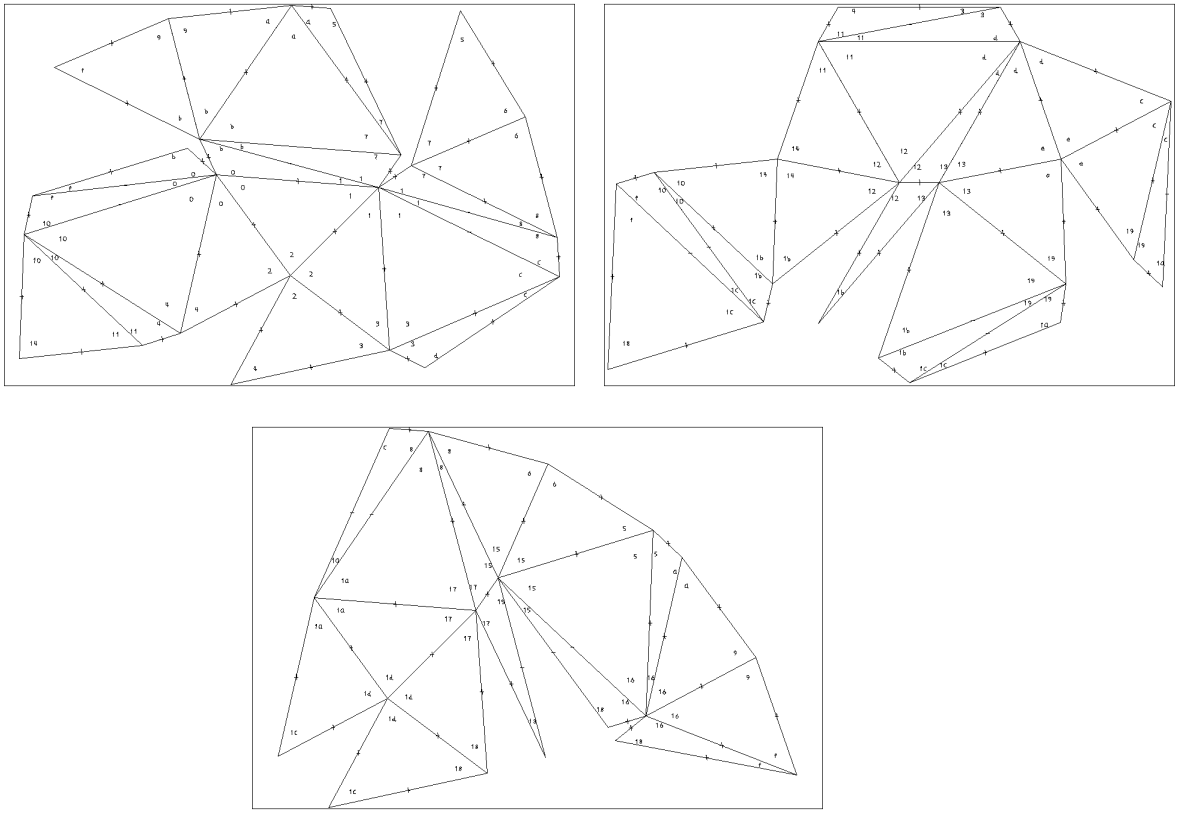

FIGURE 5. Paper templates of the sphere by strategy 2 : vertices are indexed and edges are marked $\pm$ as in figure 4.

	begin with T_1		begin with T_{51}		begin with T_{101}	
	Str #1	Str #2	Str #1	Str #2	Str #1	Str #2
number of sheets	89	92	87	79	77	88
boundary length	9.10	10.20	9.02	9.85	8.54	9.68

	begin with T_{151}		begin with T_{201}		begin with T_{251}	
	Str #1	Str #2	Str #1	Str #2	Str #1	Str #2
number of sheets	80	89	82	94	85	98
boundary length	9.17	10.03	8.82	10.25	9.41	10.31

TABLE 2. Evaluations of the two strategies in building the bunny surface : the algorithm was tried with six different initial triangles.

FIGURE 6. The sphere and bunny surfaces actually built: vertices are indexed and edges are marked $\pm$ as in figure 4.

3.3. **Dinosaur.** We try a dinosaur-shaped surface. As described in the previous example, its level set function is numerically calculated and sampled on a coarse grid. The computational domain is $[-0.0246, 0.0246] \times [-0.0631, 0.0631] \times [-0.0541, 0.0541]$ with grid resolution $11 \times 30 \times 26$. The function values in the raster-scan order can be downloaded at www.math.ewha.ac.kr/~chohong/dinosaur.txt. Its isosurface is triangulated by the simplicial isosurfacing algorithm [8]. The triangulation has 5496 triangles and 2750 vertices. Its paper templates are embedded in paper sheets of dimension 0.0349×0.0233. Figure 7 shows the actual building of the dinosaur surface. Table 3 evaluates the two strategies applied to building the surface. The algorithm was tried with six different initial triangles. Strategy 1 generated shorter boundary edges in all the trials, and consumed fewer paper sheets four times out of six.

4. CONCLUSION

We have introduced an algorithm that automatically generates paper templates of a triangulated surface. The paper templates enable the building of the surface by cutting, folding, and pasting. The algorithm is branched into two strategies : strategy 1 takes the longest neighboring edge among many choices, and strategy 2 takes the largest neighboring triangle. Three surfaces, whose triangulation sizes widely range, were successfully built by the algorithm. The two strategies were evaluated in building the surfaces with respect to paper consumption, a measure of cost efficiency, and boundary length, a measure of speed efficiency. Strategy 1 performed in most cases better than the other one with respect to boundary length, and sometimes won and sometimes lost with respect to paper consumption.

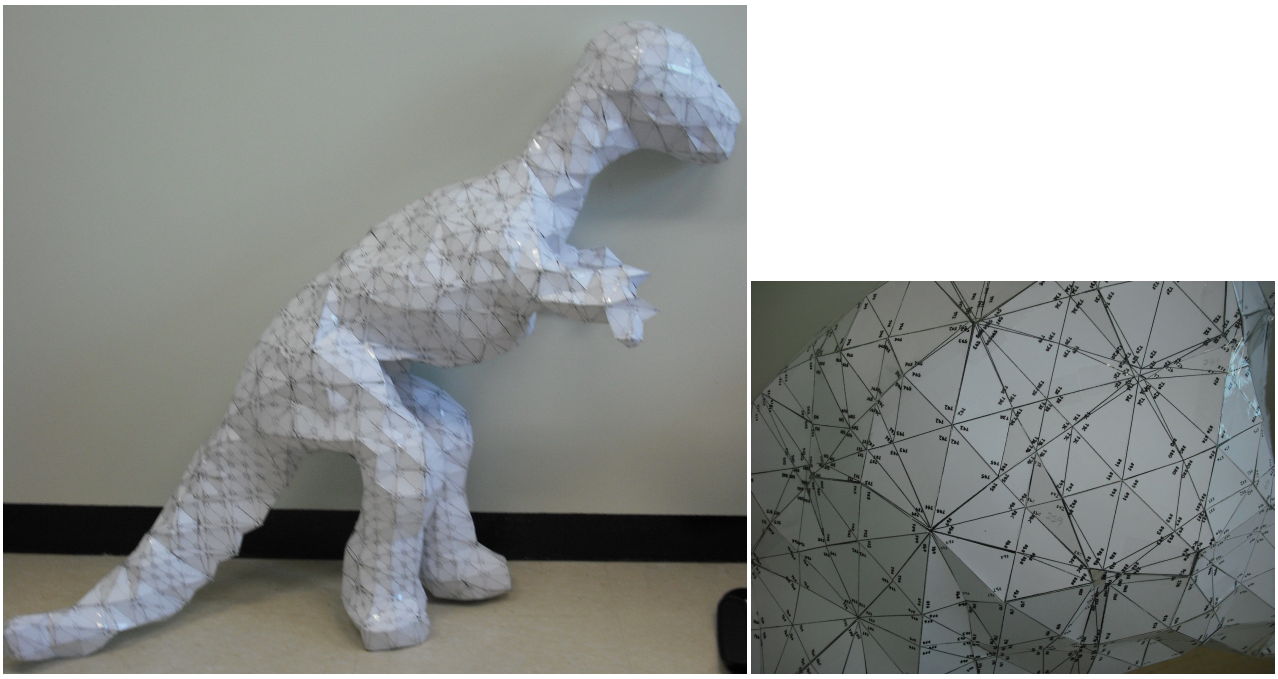

FIGURE 7. The dinosaur surface built and a detailed view around its right shoulder : vertices are indexed and edges are marked $\pm$ as in figure 4.

	begin with T_1		begin with T_{1001}		begin with T_{2001}	
	Str #1	Str #2	Str #1	Str #2	Str #1	Str #2
number of sheets	333	323	330	337	354	337
boundary length	7.76	8.91	7.81	8.91	7.83	8.91

	begin with T_{3001}		begin with T_{4001}		begin with T_{5001}	
	Str #1	Str #2	Str #1	Str #2	Str #1	Str #2
number of sheets	330	331	327	330	320	333
boundary length	7.90	9.11	7.84	9.04	7.60	8.90

TABLE 3. Evaluations of the two strategies in building the dinosaur surface : the algorithm was tried with six different initial triangles.

In cases when surfaces need be built fast, our empirical evidences suggest strategy 1. In general, we suggest trying both strategies with different initial triangles and selecting the choice suiting one's needs. We implemented the algorithm by C++ language and ran it on a regular PC. It would take only a few minutes in generating the paper temples of the examples. The time of running the algorithm is negligible to that of building the surface by cutting, folding,

and pasting. The manual work with our clumsy hands took about twenty mininutes for the sphere, five days for the bunny, and about thirty days for the dinosaur.

As far as we know, this paper is one of the papers shedding a new light on the subject of building triangulated surfaces with a viewpoint of algorithms and mathematics. We hope this work provides a foundation for future research in the area.

ACKNOWLEDGEMENT

I would like to thank Prof. Myungjoo Kang and Prof. June-Yup Lee for their helpful suggestions and advices. This work was supported by the Korea Research Foundation Grant funded by the Korean Government (KRF-2011-0013649).

REFERENCES

[1] N. Amenta, M. Bern, and M. Kamvysselis. A new voronoi-based surface reconstruction algorithm. *Proc. SIGGRAPH 98*, pages 425–421, 1998.

[2] B.P. Carnerio, C. Silva, and A.E. Kaufman. Tetra-cubes: An algorithm to generate 3d isosurfaces bases upon tetrahedra. *Anais do IX SIBGRAPI*, pages 205–210, 1996.

[3] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. *Comput. Graph. (SIGGRAPH Proc.)*, pages 67–76, 2001.

[4] J. C. Carr, R. K. Beatson, B. C. McCallum, W. R. Fright, T. J. McLennan, and T. J. Mitchell. Smooth surface reconstruction from noisy range data. In *Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, GRAPHITE '03, pages 119–ff, New York, NY, USA, 2003. ACM.

[5] A. Drozdek. Data structures and algorithms in c++. *Course Technology*, 2004.

[6] H. Edelsbrunner. Shape reconstruction with delaunay complex. 1380:119–132, 1998. 10.1007/BFb0054315.

[7] W. Lorensen and H. Cline. Marching cubes: A high-resolution 3d surface construction algorithm. *Comput. Graph. (SIGGRAPH Proc.)*, 21:168–169, 1987.

[8] C. Min. Simplicial isosurfacing in arbitrary dimension and codimension. *J. Comput. Phys.*, 190:295–310, 2003.

[9] C. Montani, R. Scateni, and R. Scopigno. A modified look-up table for implicit disambiguation of marching cubes. *The Visual Computer*, 10:353–355, 1994. 10.1007/BF01900830.

[10] G. Turk and M. Levoy. Zippered polygon meshes from range images. *Computer Graphics, SIGGRAPH*, pages 311–318, 1994.

[11] H. K. Zhao, S. Osher, B. Merriman, and M. Kang. Implicit and non-parametric shape reconstruction from unorganized points using variational level set method. *Comput. Vis. Image Unders.*, 80:295–319, 2000.

[12] H.-K. Zhao, Stanley Osher, and Ronald Fedkiw. Fast surface reconstruction using the level set method. In *1st IEEE Wrkshp. on Variational and Level Set Meth., 8th Int. Conf. on Comput. Vis.*, pages 194–202, 2001.

[13] G. M. Ziegler. Lectures on polytopes. *Graduate texts in Math. Springer-Verlag*, 1995.