Comput Syst Sci & Eng (2019) 6: 339–355

International Journal of
Computer Systems
Science & Engineering

Improving Initial Flattening of Convex-Shaped Free-Form Mesh Surface Patches Using a Dynamic Virtual Boundary

Erdem Yavuz[1] *, **Rıfat Yazıcı**[2], **Mustafa Cem Kasapbaşı**[2], **Turgay Tugay Bilgin**[1]

[1] *Department of Computer Engineering, Faculty of Engineering and Natural Sciences, Bursa Technical University, Yildirim, Bursa, Turkey*
[2] *Department of Computer Engineering, Faculty of Engineering, Istanbul Commerce University, Istanbul, Turkey*

This study proposes an efficient algorithm for improving flattening result of triangular mesh surface patches having a convex shape. The proposed approach, based on barycentric mapping technique, incorporates a dynamic virtual boundary, which considerably improves initial mapping result. The dynamic virtual boundary approach is utilized to reduce the distortions for the triangles near the boundary caused by the nature of convex combination technique. Mapping results of the proposed algorithm and the base technique are compared by area and shape accuracy metrics measured for several sample surfaces. The results prove the success of the proposed approach with respect to the base method.

Keywords: initial flattening, surface parameterization, triangular mesh, barycentric mapping, virtual boundary

1. INTRODUCTION

Finding planar counterparts of curved surfaces is a commonly known problem in the field of production (such as ships, toys, clothes, shoes and furniture industries). In the conventional method, during manufacturing, 2D pieces are sheared and assembled to form the final product. Ideally, strain factor should be taken into account in these shearing and assembling processes, because the strain produces an elastic energy that reduces quality and creates material fatigue in the final product. The conventional design process used in these industries is based on the trial-and-error method. The designer sketches 2D pieces on paper and makes a prototype of the desired product. If the result is unsatisfactory, the designer changes the patterns based on his experience and makes another prototype. This prototyping and modification steps are repeated, which are ineffective, expensive and exhausting process [1].

The design of 3D surface patches for the shape of products and the determination of their 2D blank patterns for production by surface flattening method cannot give the desired production quality due to the stresses in the materials today. The surface flattening technique is independent of strain only for surface meshes with isometric mapping properties [2]. Current commercial CAD/CAM (Computer Aided Design/Manufacturing) systems do not have the functions to model any complex, free-form surfaces, and current approaches to developable surfaces in the literature are insufficient to model free-form surfaces [1].

A number of approaches [3,4,5] have been proposed regarding the problem of building curved surfaces from planes (development). This problem is trivial when the surface has zero curvature, but when the surface has a curvature larger than zero, the problem becomes complicated. There may be many different planar development approaches for the same surface. A large

*Corresponding author: E-mail: erdem.yavuz@btu.edu.tr, ryazici@ticaret.edu.tr, mckasapbasi@ticaret.edu.tr, turgay.bilgin@btu.edu.tr

number of complex surface products manufactured today can be represented using the cloud of points. The choice of which method to use for the planar development is a sensitive issue, because there are lots of matters to consider such as production method, material properties, surface geometry.

Surface texture characterization formulations for conventional Euclidean surfaces are well-established. For example, Euclidean surfaces are defined by a single height value for each point in the plane. Free-form surfaces have a more complicated structure by nature [6]. This domain is no longer a plane and contains points with a non-zero curvature value. Therefore, such surfaces are referred to as non-Euclidean surfaces. According to Gauss' theorem of differential geometry, that is, the Egregium Theorem, surfaces with the same curvature can be projected between each other without any distortion. On the other hand, surfaces with different curvatures cannot be projected without distortion. For example, the Earth cannot be displayed on a planar surface without distortion [7]. Thus, a freeform Euclidean surface cannot be projected without distortion or without loss of some surface information. As a result, freeform surfaces can no longer be represented as height values on a two-dimensional grid. As another example, when trying to represent a hemisphere on a 2D grid, it causes distortion in the geodesic distances on the surface. That is, when projected onto the 2D grid, the geodesic distances between the different points on the real surface will be distorted and shrunk [6]. By the way, the geodesic distance between two points on a given surface is the shortest distance between two points on that surface.

With 3D scanning technology advancing rapidly, free-form surfaces quickly come to light and are used in numerous industrial product designs in many different applications. Free-form surfaces play an important role in today's graphics applications. Results of applications related to freeform surfaces, such as surface tessellation, surface rendering and surface sampling, are highly dependent on the surface parametrization. In these applications, it is often desired that the parametrization be shape preserving and done quickly.

2. RELATED WORK AND MOTIVATION

The process of flattening a complex surface is the phenomenon of projecting a surface of 3D space onto a planar surface according to certain rules [8]. A lot of studies have been done so far, and when examined, the flattening of complex surfaces can be summarized under three basic flattening approaches: geometric flattening, mechanic flattening and the hybrid flattening, which is a combination of the first two [9]. The presented study deals mainly with geometric flattening techniques.

In the study by Bodduluri and Ravani [10], freeform surfaces were first divided into simple sub-surfaces and then approached to developable surfaces. In that work, a new technique has been proposed for the geometric design of developable surfaces based on rational Bezier and B-spline curves.

In another study [11], a grid approximation flattening algorithm was proposed. In that study, an approximate local coarse projection relationship was established between a given parametric surface patch and its triangular area by minimizing the projection error function.

Ping presented a flattening algorithm [12] for complex surfaces in 1997. In the study, the metal sheet was divided into a series of strip areas and then approximated to a striped surface. Each striped surface was then divided into triangular meshes and each triangular element was mapped onto plane. From the results obtained, it is seen that the smoothed surface contains overlap and gaps. In the following year, along with some researchers, mainly by the same author, a study [13] was conducted to address the overlaps and gaps arising between stripes.

A deep research was done about interactive parametrization design in 2002 [14]. The novel family of intrinsic parametrizations of surface meshes was introduced in the study. The study produced least-distorted parametrizations by proposing a robust and fast technique. The reported results show that technique is effective in automatically designing optimal maps with or without boundary conditions.

In the early 2000s, two theoretical research findings were released by Floater. In the first one [15], he used Mean Value Theorem in the calculation of harmonic functions simplifying and improving methods for parametrization and morphing. The other study [16] was about using discrete form of Rado-Kneser-Choquet theorem for harmonic maps. Provided that the boundary of the triangulation is homeomorphic to a disk, then such mappings are one-to-one fulfilling a discrete maximum principle, stated in the study.

In another study, an innovative technique that alleviates the drawback of convex combination approach was presented [17]. The technique proposed using a virtual boundary serving to get rid of the high distortions produced by convex combination approach for the triangles near the boundary. The results reported in that study shows that the technique produced a parametrization of 3D triangular mesh surfaces with less distortion than the original approach.

In the work of Yang et al. [18], an optimization algorithm using the rational bilinear re-parametrization was proposed to improve the uniformity and orthogonality of iso-parametric curves for general rational Bézier surfaces. The reported results show that their algorithm generates more uniform and orthogonal iso-parametric curves across the rational Bézier surfaces.

In a study [19], Liu et al. proposed a fast mesh parametrization algorithm based on subdivision technique. Their method based on 4-point interpolatory subdivision approximation is indeed an extension of the chordal parameterization to surface case. Their algorithm reportedly avoids computation of linear system of equations, which makes it computationally efficient than previous methods. The proposed subdivision algorithm which is easy to implement is of high efficiency, especially in case of large number of vertices.

A simple and fast low-stretch mesh parametrization method was proposed in 2004 [20]. The study based on Floater's shape preserving parametrization and took its success one step further. As the parameter domain is a convex-shaped polygon, the optimization process in their proposed method does not generate flipped triangles as reported in that study. The efficiency and speed of the method was tested on several complex surfaces and compared with other techniques in literature.

In this work, an efficient technique has been developed to utilize the barycentric mapping theory and a dynamic virtual boundary technique in order to ensure that the length information of arbitrary 3D mesh surfaces is preserved as much as possible

and to find the 2D planar equivalents. The parametrization of the 3D surface can be described as an operation to produce a function that will correspond to a 2D planar surface. This concept plays an important role in geometry processing, since it allows to open the way to convert 3D modeling problems into 2D space, and doing it here is relatively easy. Owing to the advances in range scanning and 3D printing technologies, effective parametrizations are important for large meshes since complex 3D surfaces are widely used everywhere. In this context, an efficient algorithm for improving flattening result of triangular mesh surface patches homeomorphic to a disk is presented in this study. The proposed method incorporates a dynamic virtual boundary to reduce the distortions for the triangles near the boundary caused by the nature of convex combination approach. The proposed algorithm produces good parameterization outputs with relatively low distortions, which can be presented as input to a further energy releasing process by an energy-based approach.

The rest of this work is organized as follows: Section 3 gives the theory and concept of barycentric mapping. The proposed initial mapping approach is elaborated in Section 4. Metrics for measuring accuracy of planar developments are given in Section 5. An illustrative example about the proposed approach is given in Section 6. Experimental results and discussion are presented in Section 7. Finally, Section 8 concludes the paper.

3. BARYCENTRIC MAPPING THEORY

Barycentric mapping is a very efficient mapping method to produce parameterization of triangular surfaces [21]. This approach is based on Tutte's barycentric mapping theory [22] and also provides a basis for surface parameterization and approximation studies [23] of Floater, who is renowned for its successful works in the fields. This theorem states that for a triangular surface homeomorphic to a disk, if boundary nodes are on a convex polygon and internal nodes are a convex combination of their neighbors, then all (u, v) planar coordinates form a valid parameterization [21, 23].

A scheme is needed for determining specification of convex combinations which guarantees a planar shape preserving local shapes of the 3D surface. Let $\boldsymbol{S}(\boldsymbol{X},\boldsymbol{F})$ be a triangulated surface defined by vertices ($\boldsymbol{X}$=$\{(\boldsymbol{x}_i = (x_i, y_i, z_i))\}$, $1 \leq \quad i \quad \leq N$) and facets $\boldsymbol{F}$ (a matrix storing indices of triangles generated by Delaunay Triangulation). Considering the definitions here, the barycentric mapping method is composed of the following steps.

Step 1: Boundary node coordinates of the polygon on which the mapping will be done is specified. This polygon may be unit circle, square or an arbitrary enclosure resembling the original 3D surface. To summarize,

$$\boldsymbol{u}_{n+1}, \ldots, \boldsymbol{u}_N \tag{1}$$

are determined to be on the polygon (which is defined in $\mathrm{D} \subset \mathrm{R}^2$) in an anti-clockwise manner.

Step 2: Choose $\lambda_{i,j}$ as real values for $j = 1,\ldots N$ for each $i \in \{1, \ldots, n\}$ satisfying the following conditions [21, 23].

$$\left.\begin{array}{l} \lambda_{i,j} > 0 \\ -\lambda_{i,i} = -\sum_{j\neq i}^{N} \lambda_{i,j} \\ \lambda_{i,j} = 0 \end{array}\right\} \begin{array}{l} \text{if } i^{\text{th}} \text{ and } j^{\text{th}} \text{ nodes are} \\ \text{connected by an edge,} \\ \lambda_{i,j} = 0 \quad \text{otherwise} \end{array} \tag{2}$$

Step 3: $\boldsymbol{u}_1,\ldots,\boldsymbol{u}_n$ are defined as the solution of the following linear equations where

$$\boldsymbol{u}_i = \sum_{j=1}^{N} \lambda_{i,j}\, \boldsymbol{u}j, \quad i = 1, \ldots, n \tag{3}$$

Solving these equations yields $\boldsymbol{P} = \boldsymbol{P}(\boldsymbol{F}, \boldsymbol{U}_b, \wedge)$ where $\boldsymbol{U}_b = \{ \boldsymbol{u}_{n+1}, \ldots, \boldsymbol{u}_N \}$ and $\wedge = (\lambda_{i,j})_{i=1,\ldots,n;\, j=1,\ldots,N}$. Here $\boldsymbol{P}$ is an embedding of $\boldsymbol{S}(\boldsymbol{X},\boldsymbol{F})$ in R^2 and $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N$ represent positions of nodes connected to each other by straight lines.

Tutte studied how to draw straight lines of planar graph [24] years ago. He suggested for large scale graphs including triangulated ones that Eq. 3 be run as $\lambda_{i,j} = 1/d_i$ for neighboring nodes ($i = 1, \ldots, n$) where d_i represents number of neighbors of ith node. Therefore, $\boldsymbol{u}_i$ can be regarded as barycenter of neighbors. He has proven that there exists a unique solution, and $\boldsymbol{P}$ is a solution consisting of straight lines [23]. In other words, $\boldsymbol{u}_1,\ldots,\boldsymbol{u}_n$ are distinct guaranteeing that no two edges intersect except at common end points of triangles [24]. Considering the general case in Eq. 2, the equations can be rewritten as follows in order to show that Eq. 3 has a unique solution [20, 22].

$$\boldsymbol{u}_i - \sum_{j=1}^{n} \lambda_{i,j}\boldsymbol{u}_j = \sum_{j=n+1}^{N} \lambda_{i,j}\boldsymbol{u}_j\,, \quad i = 1, \ldots, n \tag{4}$$

The reason for arranging the equation as above is that boundary nodes are fixed on border line of the polygon and then internal nodes are obtained by solving Eqs. 2 and 3. When considering that $\boldsymbol{u}_i$ has two separate components u and v, this equals two matrix equations:

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{b}_1, \quad \boldsymbol{A}\boldsymbol{v} = \boldsymbol{b}_2 \tag{5}$$

These two linear system of size n (number of internal nodes) are solved in order to find two column vectors storing positions of internal nodes: $(u_1, \ldots, u_n)^T$ and $(v_1, \ldots, v_n)^T$. Right hand side of the equations, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ store weighted coordinates of neighboring nodes. $\boldsymbol{A}$ is square matrix of size $n \times n$ and its elements are:

$$a_{i,i} = 1, \quad a_{i,j} = -\lambda_{i,j}, \quad j \neq i \tag{6}$$

That is to say, the matrix takes the following form.

$$A = \begin{bmatrix} 1 & -\lambda_{1,2} & -\lambda_{1,n} & \\ -\lambda_{2,1} & 1 & \cdots & -\lambda_{2,n} \\ \vdots & & \ddots & \vdots \\ -\lambda_{n,1} & -\lambda_{n,2} & \ldots & 1 \end{bmatrix} \tag{7}$$

The existence of unique solution of Eq. 3 is dependent on the non-singularity of the matrix $\boldsymbol{A}$. In other words, this is obtained by computing the inverse of $\boldsymbol{A}$. There are a few ways to solve Eq. 3. Sparse iterative and direct methods are the ones which are most effective for large-scale meshes. On the other hand, Gauss-Siedel solvers can be used for reasonable small meshes [21].

4. PROPOSED INITIAL MAPPING APPROACH

The proposed initial mapping method for computing planar equivalents of 3D surface patches is based on Floater's

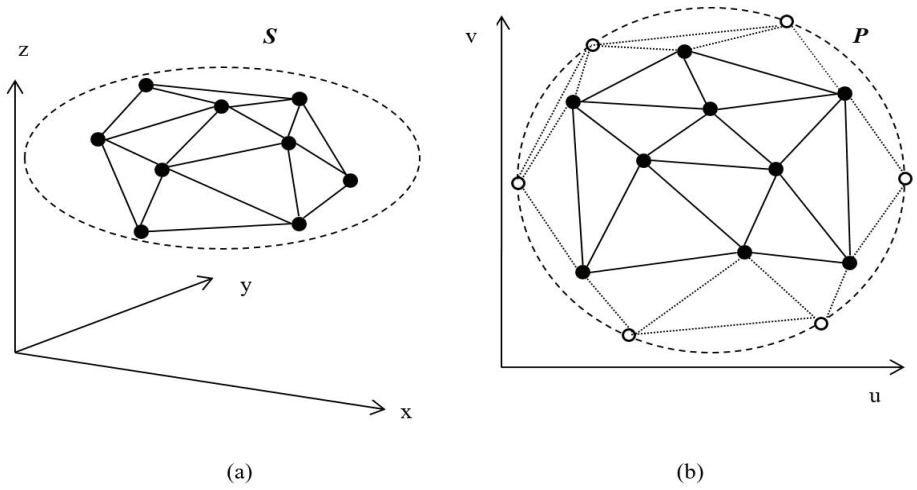


Figure 1 Virtual boundary in 3D and 2D space:mesh surface patch and virtual boundary (a), and its equivalent on the plane (b).

barycentric mapping theory, mathematical foundations of which is given in the previous chapter. The proposed approach effectively improves outputs of the base method by incorporating virtual boundary nodes computed dynamically depending upon the complex surface patch to be flattened. Fig. 1 illustrates virtual boundary nodes created dynamically in order to improve flattening result of 3D surface patch and its planar equivalent.

In the convex combination approach, fixing the real boundary nodes of a surface patch directly on a polygon causes the triangles near the boundary in 2D parametric space to be highly deformed and distorted [17] . So, a set of virtual nodes encompassing real nodes of the 3D surface patch are adaptively created. For the ease of computability, virtual nodes are placed on the trajectory of a circle, whose diameter is determined according to total triangular area of the 3D surface patch to be flattened. Radius of the circle that hosts dynamic virtual nodes is computed according to the formula:

$$r_v = \frac{\sum_{i=0}^{n-1} A_i}{\pi} \tag{8}$$

where A_i is the area of the ith triangle in the facet model, n is the triangle number in the facet model. As the boundary polygon in the barycentric mapping method fixes the boundary nodes of planar equivalent of the 3D surface patch at the beginning, these boundary nodes remain constant throughout the planar development process. Thus, determination of the bounding polygon is critically important for the output of planar development to be of high fidelity to its original 3D counterpart.

Number of virtual boundary nodes to be placed on the trajectory of the circle affects the extent of similarity to the shape of original 3D surface patch. Therefore, number of virtual boundary nodes should be large enough to provide reasonable resolution on the trajectory. It is determined to be proportional to total node number of the 3D surface patch. How the determination of number of virtual boundary nodes affects the initial flattening result is explored for several resolution values later in the study. Fig. 2 shows virtual boundary polygons created for a particular 3D surface patch at various resolutions, such as the number of boundary nodes $n_v = 4$, 6, 8 and 16 respectively in figures (a) through (d).

Boundary nodes are positioned on the trajectory of virtual circle of radius r_v at equal angle intervals α_v determined by the following equation:

$$\alpha_v = \frac{360^\circ}{n_v} \tag{9}$$

The proposed initial mapping approach with a dynamic virtual boundary is presented in Algorithm 1. Globally, the proposed algorithm takes two arguments $\boldsymbol{X}$ and $\boldsymbol{F}$ as input, and produces the planar equivalent $\boldsymbol{P}$ of $\boldsymbol{S}(\boldsymbol{X},\boldsymbol{F})$ as output. The output $\boldsymbol{P}$ consists of n nodes in the u-v coordinate system, i.e. planar coordinates.

In the barycentric mapping approach, neighborhood weights are computed by the simple expression $\lambda_{i,j} = 1/d_i$, which evenly distributes weighting shares inversely proportional to the number of neighbors. In the proposed study, these weights are determined by computing reciprocal of the distances between neighbors and then normalizing the result by a scaling factor. For this purpose, the expression $\lambda_{i,j} = 1/d_i$, used by Tutte for representation of graphs[19], is replaced by the two expressions given in steps 10 and 11 of Algorithm 1. The Euclidean distance between each neighboring vertex in the 3D triangular mesh is computed by the expression in the step 10. The expression in the step 11 normalizes the result dividing d_{ij} by the total cost of neighbors of the each central vertex. Thus, it is aimed to preserve shape of the triangular mesh surface locally in the parametric space [25], which is substantiated by the test results in the following sections.

5. ACCURACY MEASUREMENT

Generally, two metrics are used in order to evaluate accuracy of the planar surface being developed. These metrics are area and shape accuracies, details of which are explained under the following subheadings.

5.1 Area Accuracy

During the development process, planar surface area may change, and the final relative area difference is computed by the formula [26]:

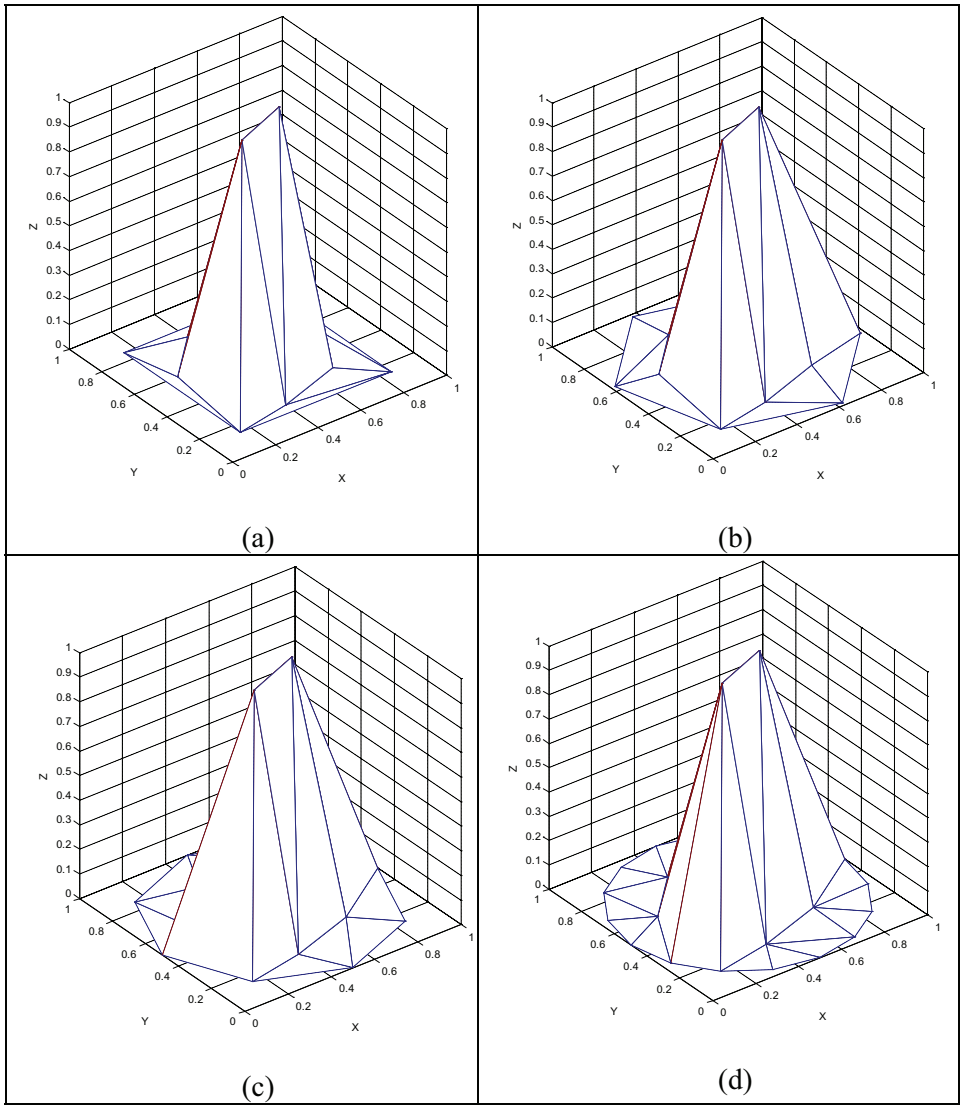


Figure 2 Virtual boundary polygons at various resolutions: number of boundary nodes (a) $n_v = 4$; (b) $n_v = 6$; (c) $n_v = 8$; (d) $n_v = 16$.

$$E_S = \frac{\sum |A - A'|}{\sum A} \tag{10}$$

where A is the real area of the original mesh surface patch before the development and A' is the area of the corresponding surface patch on the planar surface after the development. A relatively complex relation of the double integral must be solved in order to calculate the area of a surface defined by S(u, v) [26]. Since the facet model is adopted in order to represent a complex surface in the approach, it is not possible to determine the analytical form of the surface S in this study. Therefore, the area A can approximately be calculated simply by adding up area of each triangle in the facet model using the formula:

$$A = \sum_{i=0}^{n-1} A_i \tag{11}$$

where A_i is the area of the ith triangle and n is the number of triangles.

5.2 Shape Accuracy

Shape accuracy represents the total difference rate between the length of arc on the surface mesh and its corresponding edge length [26]. The final relative shape accuracy is computed by the formula:

$$E_C = \frac{\sum |L - L'|}{\Sigma L} \tag{12}$$

where L is the real length of an arc on the original mesh surface patch before the development and L' is the length of the corresponding edge on the plane after the development. Assuming that the curve segment L is on the curve defined by the formula $C = S$ (u(t), v(t)) on the surface S, computing the segment length requires to solve a complex double integral problem. Similar to the aforementioned problem related to the adopted model, it is not possible to determine the analytic form of the curve C. Therefore, the length L can approximately be calculated by simply adding up lengths of each triangle in the facet model using the formula:

$$L = \sum_{i=0}^{m-1} L_i \tag{13}$$

Algorithm 1 Initial Map With Virtual Boundary

Input: A given 3D triangular mesh surface $\boldsymbol{S}(\boldsymbol{X},\boldsymbol{F})$ where $\boldsymbol{X}$ contains 3D points (nodes) and $\boldsymbol{F}$ contains vertex indices of triangles.

Output: The initial 2D shape $\boldsymbol{P}$ of the surface $\boldsymbol{S}$.

1: Compute the radius of the virtual boundary circle r_v according to the Eq. (8).
2: Determine n_v, the number of virtual boundary nodes according to the total number of original surface
3: Compute the step angle of the virtual nodes α_v according to the Eq. (9).
4: Append computed virtual nodes to the original mesh surface $\boldsymbol{S}(\mathbf{X},\boldsymbol{F})$ to obtain the augmented surface $\boldsymbol{S}^*(\mathbf{X},\boldsymbol{F})$.
5: Examine the 3D mesh surface $\boldsymbol{S}^*(\mathbf{X},\boldsymbol{F})$ and determine boundary nodes $(n{+}1,\ldots,N)$.
6: Sort $\boldsymbol{X}$(size of which is N x 3)in an order so that the boundary nodes follow internal nodes.
7: Choose $\boldsymbol{u}_{n+1}, \ldots, \boldsymbol{u}_N$to be K-sided convex polygon D $\subset \mathrm{R}^2$ in an anticlockwise sequence, corners of which are placed at angular intervals α_v on the virtual circle.
8: **for** i from 1 to n **do**
9: **for** j from 1 to N **do**
10: $d_{i,j} = 1/\|x_{i,i} - x_{i,j}\|$
11: Set $\lambda_{i,j} = d_{i,j}/\Sigma_{j=1}^{N}(d_{i,j})$ obeying to the general rules in Eqs.(2, 6).
12: **end for**
13: **end for**
14: Compute inverse of the matrix $\boldsymbol{A}$ (holding $\lambda_{i,j}$ values) using one of the aforementioned iterative methods.
15: Multiply both $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$in Equation (5) by the inverse matrix $\boldsymbol{A}^{-1}$ in order to get final planar coordinates of internal nodes $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$.

where L_i is the length of the ith triangle and m is the number of triangles.

6. NUMERICAL EXAMPLE

The proposed initial mapping algorithm has been run on a simple surface patch of 7-vertices (named *Surf1*) in order for the intermediate results to be tracked easily. For this purpose, the simple surface patch given in Fig. 3 is presented to the algorithm.

The cloud of points, defined by $\boldsymbol{X}$, corresponding to the vertices of the mesh surface patch *Surf1* and the topology data $\boldsymbol{F}$ defining its triangular facets are as follows:

$$\boldsymbol{X} = \begin{bmatrix} 0.3 & 0.4 & 1 \\ 0.6 & 0.5 & 1 \\ 0.2 & 0.2 & 0 \\ 0.1 & 0.6 & 0 \\ 0.4 & 0.9 & 0 \\ 0.8 & 0.7 & 0 \\ 0.7 & 0.3 & 0 \end{bmatrix}, \boldsymbol{F} = \begin{bmatrix} 6 & 5 & 2 \\ 6 & 2 & 7 \\ 1 & 5 & 4 \\ 1 & 2 & 5 \\ 4 & 3 & 1 \\ 3 & 7 & 1 \\ 1 & 7 & 2 \end{bmatrix} \tag{14}$$

The number of virtual boundary nodes is arbitrarily chosen to be $n_v = 8$ for this sample surface patch. Augmented surface patch $\boldsymbol{S}^*$, which is obtained by incorporating a virtual boundary polygon is given in Fig. 4.

The augmented surface with virtual boundary vertices, $\boldsymbol{S}^*$ represented by $\boldsymbol{X}^*$ and $\boldsymbol{F}^*$ is given by

$$\boldsymbol{X}^* = \begin{bmatrix} \mathrm{rx}+0.3 & \mathrm{ry}+0.4 & 1 \\ \mathrm{rx}+0.6 & \mathrm{ry}+0.5 & 1 \\ \mathrm{rx}+0.2 & \mathrm{ry}+0.5 & 1 \\ \mathrm{rx}+0.2 & \mathrm{ry}+0.6 & 0 \\ \mathrm{rx}+0.4 & \mathrm{ry}+0.9 & 0 \\ \mathrm{rx}+0.8 & \mathrm{ry}+0.7 & 0 \\ 0.28 & 0.28 & 0 \\ 1.05 & 0.28 & 0 \\ 1.65 & 1.65 & 0 \\ 0.96 & 1.93 & 0 \\ 0.28 & 1.65 & 0 \\ 0 & 0.96 & 0 \end{bmatrix}, \quad \boldsymbol{F}^* = \begin{bmatrix} 15 & 4 & 14 \\ 5 & 12 & 13 \\ 13 & 14 & 5 \\ 5 & 14 & 4 \\ 11 & 12 & 6 \\ 6 & 5 & 2 \\ 12 & 5 & 6 \\ 7 & 10 & 11 \\ 11 & 6 & 7 \\ 7 & 6 & 2 \\ 9 & 10 & 7 \\ 15 & 8 & 3 \\ 3 & 4 & 15 \\ 3 & 8 & 9 \\ 9 & 7 & 3 \\ 1 & 7 & 2 \\ 1 & 3 & 7 \\ 4 & 3 & 1 \\ 2 & 5 & 1 \\ 1 & 5 & 4 \end{bmatrix} \tag{15}$$

Herein, the new vertices given in green color in the matrix $\boldsymbol{X}^*$, corresponding to the points in 3D space are the vertices augmented to the original mesh surface patch by using a virtual boundary polygon. Translation offset values added to positions of original vertices are calculated to be rx=ry=r_v/2=0.48 for this sample surface patch. When the augmented surface is presented as the input to Algorithm 1, the matrix $\boldsymbol{\Lambda}$ is computed to be as the following:

$$\boldsymbol{\Lambda} = \left[\begin{array}{ccccccc|cccccccc} -1 & 0.46 & 0.14 & 0.14 & 0.13 & 0 & 0.13 & 0 & 0.0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.53 & -1 & 0 & 0 & 0.15 & 0.16 & 0.16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.1 & 0 & -1 & 0.25 & 0 & 0 & 0.2 & 0.18 & 0.14 & 0 & 0 & 0 & 0 & 0.14 & \\ 0.11 & 0 & 0.27 & -1 & 0.26 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.17 & 0.19 \\ 0.08 & 0.08 & 0 & 0.21 & -1 & 0.2 & 0 & 0 & 0 & 0 & 0 & 0.11 & 0.16 & 0.16 & 0 \\ 0 & 0.11 & 0 & 0 & 0.25 & -1 & 0.28 & 0 & 0 & 0 & 0.17 & 0.19 & 0 & 0 & 0 \\ 0.09 & 0.09 & 0.19 & 0 & 0 & 0.24 & -1 & 0 & 0.12 & 0.14 & 0.13 & 0 & 0 & 0 & 0 \end{array}\right] \tag{16}$$

Size of matrix $\boldsymbol{\Lambda}$ is determined both by the numbers of original and augmented mesh surface patches. For this sample surface, its size becomes 7×15. First seven vertices, coordinates of which are given in black color, of the matrix given in Eq. 15 correspond to internal nodes. On the other hand, the remaining eight vertices, coordinates of which are given in green color are the virtual boundary nodes computed dynamically for the sample surface. Then, the square matrix $\boldsymbol{A}$ is formed using the elements of the matrix $\boldsymbol{\Lambda}$ as in the way defined in Eq. 7. Flattening result (in u-v parameter space) for the subject augmented surface is given in Fig. 5a.

Ultimate flattening result of the original mesh surface patch *Surf1*, which is given in Fig. 5b is obtained by removing the virtual nodes from the flattening result of the augmented surface. At the end of initial planar development process, area and shape accuracies for the subject mesh surface are obtained as $E_S = 0.7570$ and $E_C = 0.5825$ under the chosen conditions.

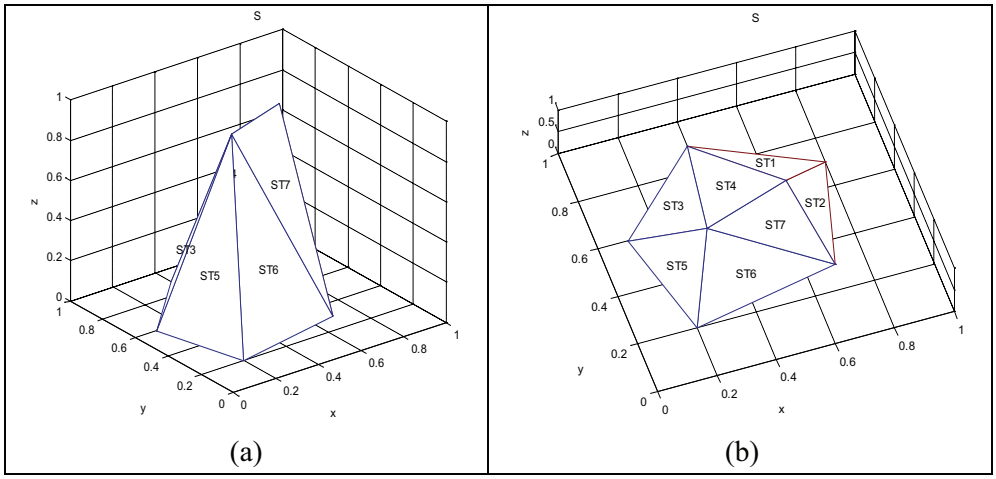


Figure 3 A sample surface patch of 7-vertices (*Surf1*), two different viewpoints (a, b)

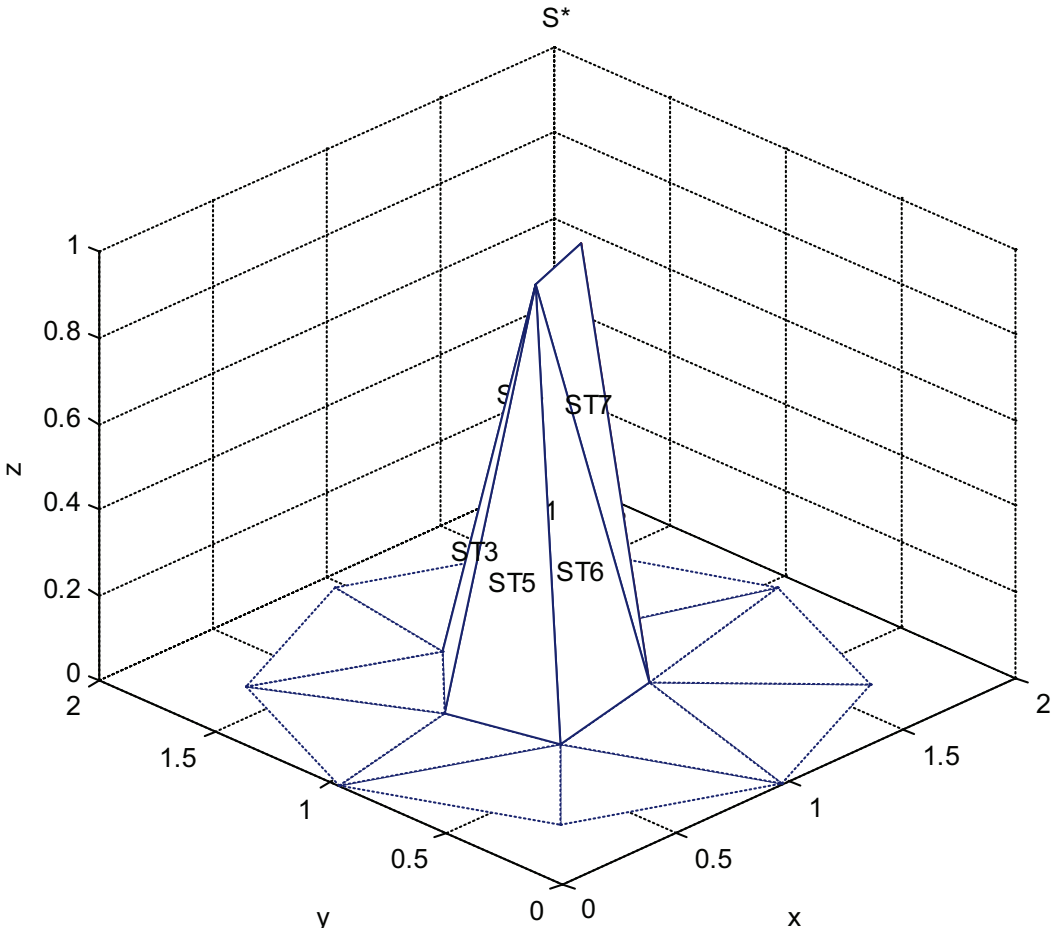


Figure 4 Augmented surface obtained by adding 8 virtual boundary vertices to the original surface patch *Surf1*.

7. EXPERIMENTAL RESULTS AND DISCUSSION

The proposed initial flattening algorithm is tested on several arbitrary triangular mesh surface patches of various sizes and only three of them are included in this section. One of the triangular mesh surface patches is *Surf1*, given in the previous section and the other two are *Surf2* and *Surf3*, seen in Figs. 6 and 7.

The cloud of points, defined by $\boldsymbol{X}$, corresponding to the vertices of the mesh surface patch and the topology data $\boldsymbol{F}$ defining its triangular facets for *Surf2* and *Surf3* are given in Eq.17 and 18 respectively.

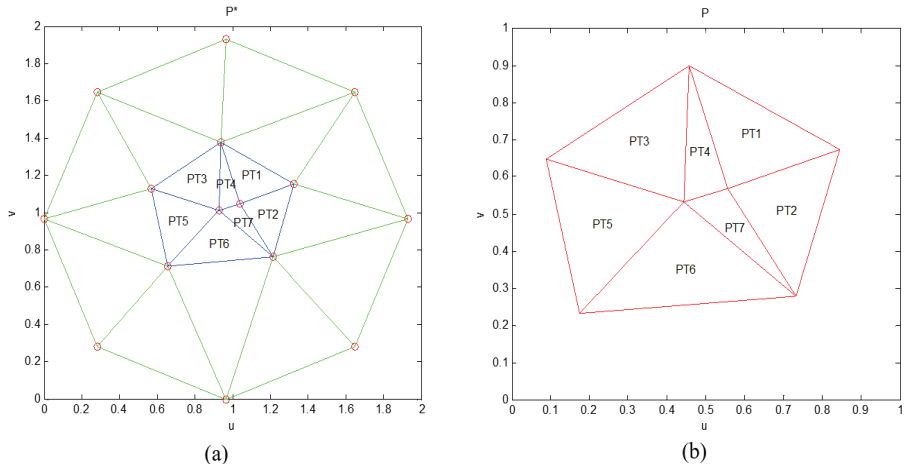


Figure 5 Flattening result of mesh surface patch for *Surf1*:the augmented surface (a) and its ultimate result obtained by removing virtual nodes (b).

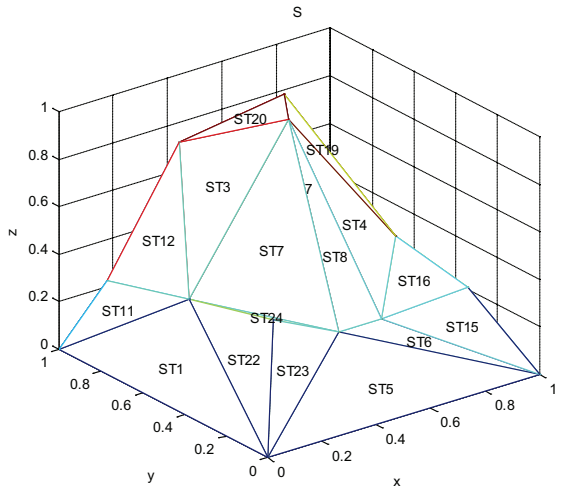


Figure 6 An arbitrary sample surface patch of 15-vertices (*Surf2*).

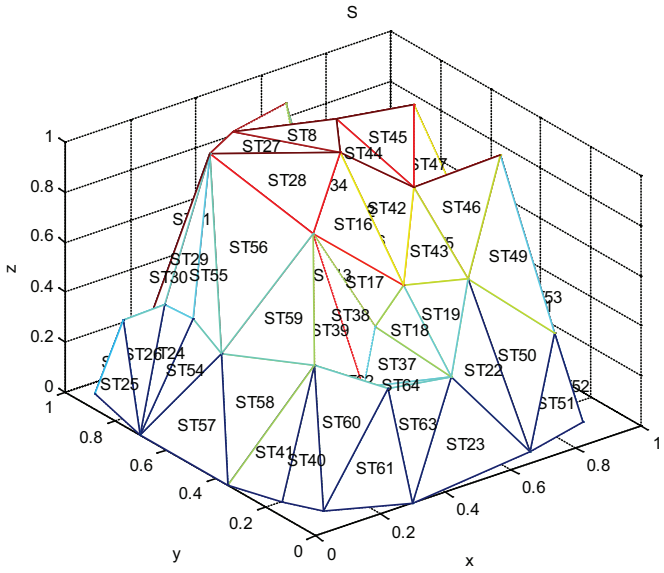
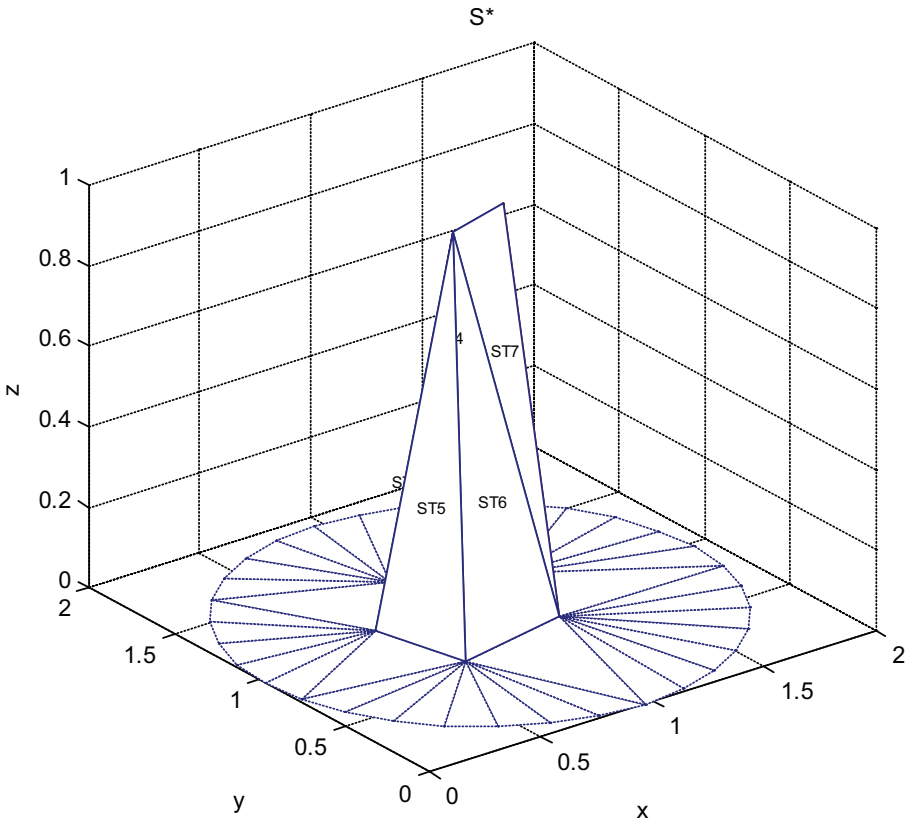


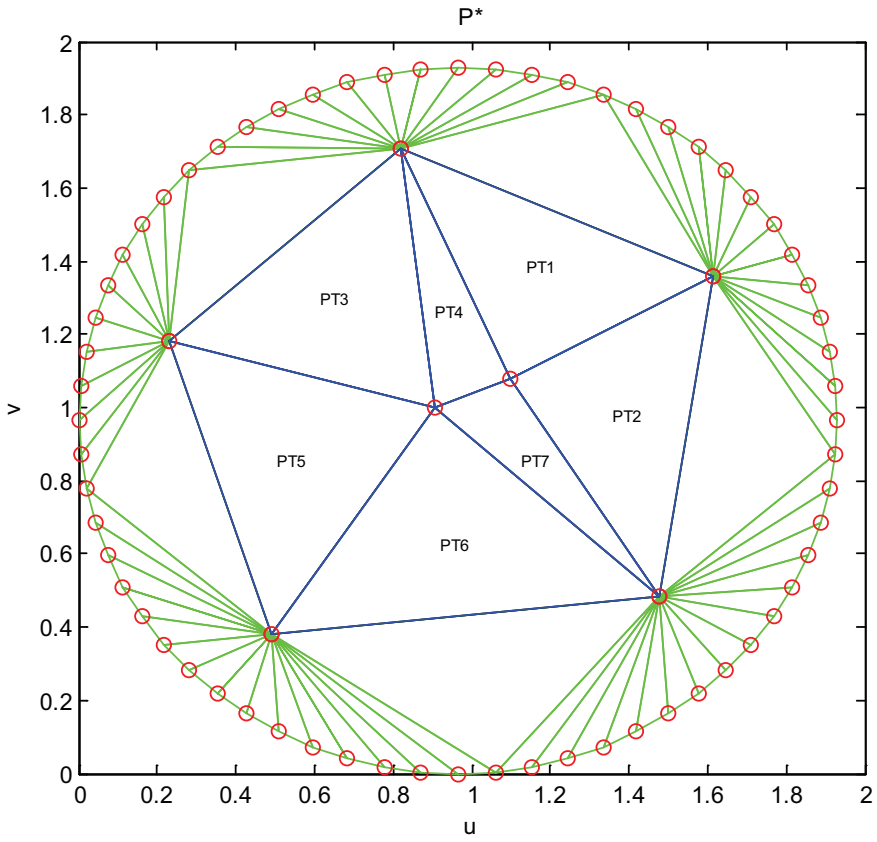
Figure 7 An arbitrary sample surface patch of 40-vertices (*Surf3*).

$$\boldsymbol{X} = \begin{bmatrix} 0.1 & 0.1 & 0.5 \\ 0.1 & 0.9 & 0.3 \\ 0.65 & 0.82 & 0.45 \\ 0.7 & 0.3 & 0.55 \\ 0.3 & 0.05 & 0.4 \\ 0.85 & 0.15 & 0.35 \\ 0.5 & 0.55 & 1 \\ 0.6 & 0.7 & 1 \\ 0.08 & 0.48 & 0.42 \\ 0.48 & 0.08 & 0.38 \\ 0.25 & 0.75 & 0.9 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} 9 & 13 & 12 \\ 14 & 13 & 3 \\ 11 & 9 & 7 \\ 7 & 10 & 4 \\ 12 & 15 & 5 \\ 7 & 9 & 5 \\ 5 & 10 & 7 \\ 2 & 3 & 13 \\ 2 & 11 & 3 \\ 13 & 9 & 2 \\ 9 & 11 & 2 \\ 6 & 15 & 14 \\ 14 & 4 & 6 \\ 6 & 10 & 15 \\ 6 & 4 & 10 \\ 14 & 3 & 8 \\ 8 & 4 & 14 \\ 7 & 4 & 8 \\ 8 & 11 & 7 \\ 3 & 11 & 8 \\ 1 & 9 & 12 \\ 12 & 5 & 1 \\ 1 & 5 & 9 \end{bmatrix} \tag{17}$$

$$\boldsymbol{X} = \begin{bmatrix} 0.15 & 0.2 & 0.5 \\ 0.1 & 0.9 & 0.3 \\ 0.15 & 0.8 & 0.4 \\ 0.2 & 0.75 & 0.35 \\ 0.7 & 0.9 & 0.45 \\ 0.7 & 0.3 & 0.55 \\ 0.6 & 0.8 & 0.5 \\ 0.65 & 0.75 & 0.6 \\ 0.3 & 0.1 & 0.4 \\ 0.3 & 0.2 & 0.35 \\ 0.85 & 0.15 & 0.35 \\ 0.5 & 0.55 & 1 \\ 0.6 & 0.7 & 1 \\ 0.6 & 0.9 & 0.95 \\ 0.08 & 0.48 & 0.42 \\ 0.48 & 0.08 & 0.38 \\ 0.25 & 0.75 & 0.99 \\ 0.4 & 0.85 & 0.95 \\ 0.8 & 0.65 & 1 \\ 0.3 & 0.4 & 0.85 \\ 0.8 & 0.3 & 1 \\ 0.4 & 0.28 & 0.5 \\ 0.9 & 0.6 & 0.6 \\ 0.9 & 0.8 & 0.45 \\ 0.51 & 0.31 & 0.6 \\ 0.65 & 0.45 & 0.85 \\ 0.1 & 0.1 & 0 \\ 0.05 & 0.95 & 0 \\ 0.95 & 0.95 & 0 \\ 0.9 & 0.1 & 0 \\ 0.3 & 0 & 0 \\ 0.7 & 0.05 & 0 \\ 0.2 & 1 & 0 \\ 0.7 & 1 & 0 \\ 0 & 0.35 & 0 \\ 0 & 0.7 & 0 \\ 1 & 0.45 & 0 \\ 1 & 0.75 & 0 \\ 0.95 & 0.2 & 0 \\ 0.05 & 0.2 & 0 \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} 24 & 38 & 29 \\ 33 & 28 & 2 \\ 37 & 38 & 23 \\ 23 & 24 & 19 \\ 38 & 24 & 23 \\ 5 & 24 & 29 \\ 33 & 2 & 3 \\ 12 & 13 & 18 \\ 18 & 14 & 33 \\ 8 & 13 & 19 \\ 19 & 24 & 8 \\ 19 & 24 & 8 \\ 24 & 5 & 8 \\ 34 & 5 & 29 \\ 14 & 5 & 34 \\ 33 & 14 & 34 \\ 12 & 20 & 25 \\ 25 & 20 & 22 \\ 25 & 22 & 16 \\ 16 & 6 & 25 \\ 37 & 23 & 21 \\ 21 & 39 & 37 \\ 32 & 6 & 16 \\ 16 & 31 & 32 \\ 4 & 3 & 36 \\ 36 & 2 & 28 \\ 36 & 3 & 2 \\ 12 & 18 & 17 \\ 17 & 20 & 12 \\ 17 & 3 & 4 \\ 33 & 3 & 7 \\ 17 & 18 & 33 \\ 14 & 18 & 7 \\ 7 & 18 & 13 \\ 13 & 8 & 7 \\ 7 & 5 & 14 \\ 7 & 8 & 5 \\ 16 & 22 & 10 \\ 22 & 20 & 10 \\ 20 & 1 & 10 \\ 27 & 1 & 40 \\ 40 & 1 & 35 \\ 12 & 25 & 26 \\ 26 & 25 & 6 \\ 26 & 13 & 12 \\ 19 & 13 & 26 \\ 6 & 21 & 26 \\ 26 & 23 & 19 \\ 26 & 21 & 23 \\ 11 & 21 & 6 \\ 6 & 32 & 11 \\ 11 & 32 & 30 \\ 30 & 39 & 11 \\ 39 & 21 & 11 \\ 4 & 36 & 15 \\ 15 & 17 & 4 \\ 20 & 17 & 15 \\ 15 & 36 & 35 \\ 35 & 1 & 15 \\ 15 & 1 & 20 \\ 9 & 1 & 27 \\ 27 & 31 & 9 \\ 9 & 10 & 1 \\ 9 & 31 & 16 \\ 16 & 10 & 9 \end{bmatrix} \tag{18}$$



(a)



(b)

Figure 8 The augmented surface patch of *Surf1* for $n_v = 64$ (a) and its planar equivalent (b).

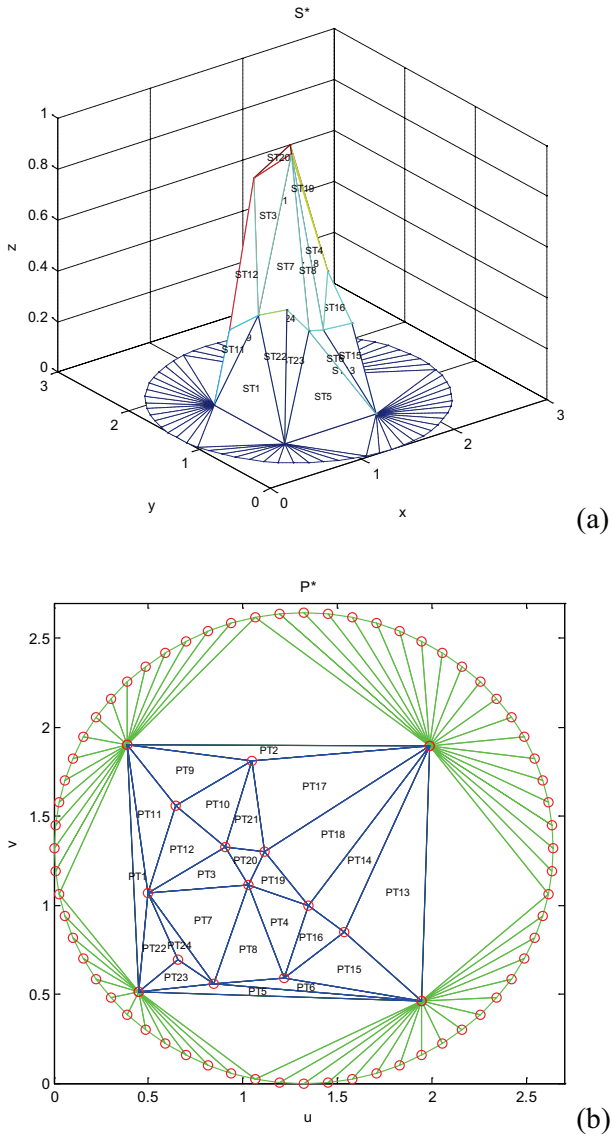


Figure 9 The augmented surface patch of *Surf2* for $n_v = 64$ (a) and its planar equivalent (b).

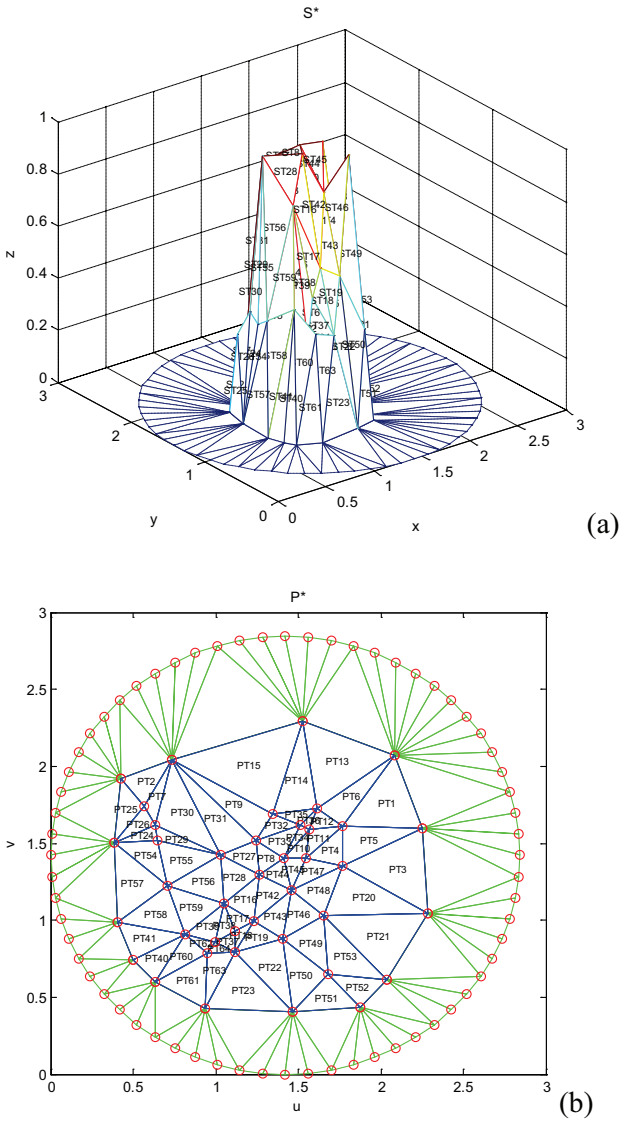


Figure 10 The augmented surface patch of *Surf3* for $n_v = 64$ (a) and its planar equivalent (b).

7.1 Effect of Virtual Boundary Node Number on Flattening Accuracy

The three sample mesh surfaces mentioned above are given as input to Algorithm 1 and initial flattening results are obtained for several number of virtual boundary nodes. However, only one flattening result is given for each sample surface patch due to the page limitation. Flattening results of the sample surface patches for $n_v = 64$ are given in Figs. 8, 9 and 10.

For all of the sample surface patches, virtual boundary polygons with 4, 6, 8, 12, 16, 32 and 64 nodes have been formed and their mappings to the plane have been realized. Area and shape accuracies measured for the obtained flattening results are tabulated in Table 1.

Area and shape accuracies for the sample surface patch *Surf1* are also illustrated in the bar graph in Fig. 11. When the number of virtual boundary nodes is $n_v = 4$, the area accuracy and the shape accuracy is calculated to be $E_S = 0.8875$ and $E_C = 0.7098$, respectively. The accuracies decreased gradually with the increase in the number of virtual boundary nodes and are measured as $E_S = 0.2038$ and $E_C = 0.4202$ for $n_v = 64$.

Area and shape accuracies for *Surf2* are also illustrated in the bar graph in Fig. 12. When the number of virtual boundary nodes is $n_v = 4$, the area accuracy and the shape accuracy is calculated to be $E_S = 0.9211$ and $E_C = 0.6974$, respectively. The accuracies decreased gradually with the increase in the number of virtual boundary nodes and are measured as $E_S = 0.4906$ and $E_C = 0.2697$ for $n_v = 64$.

Area and shape accuracies for *Surf3* are also illustrated in the bar graph in Fig. 13. When the number of virtual boundary nodes is $n_v = 4$, the area accuracy and the shape accuracy is calculated to be $E_S = 0.8400$ and $E_C = 0.6521$, respectively. The accuracies decreased gradually with the increase in the number of virtual boundary nodes and are measured as $E_S = 0.3351$ and $E_C = 0.2905$ for $n_v = 64$. The test results reported here indicates that increasing the number of virtual boundary nodes (based on the size of the surface patches) by a certain extent improves area and shape accuracies of the planar projection obtained in the flattening result.

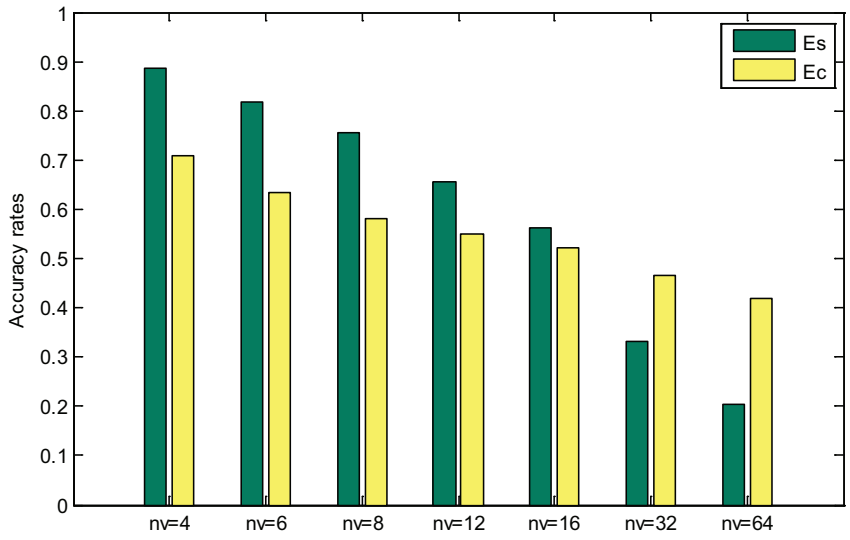


Figure 11 Area and shape accuracies measured for *Surf1* for various number of virtual boundary nodes.

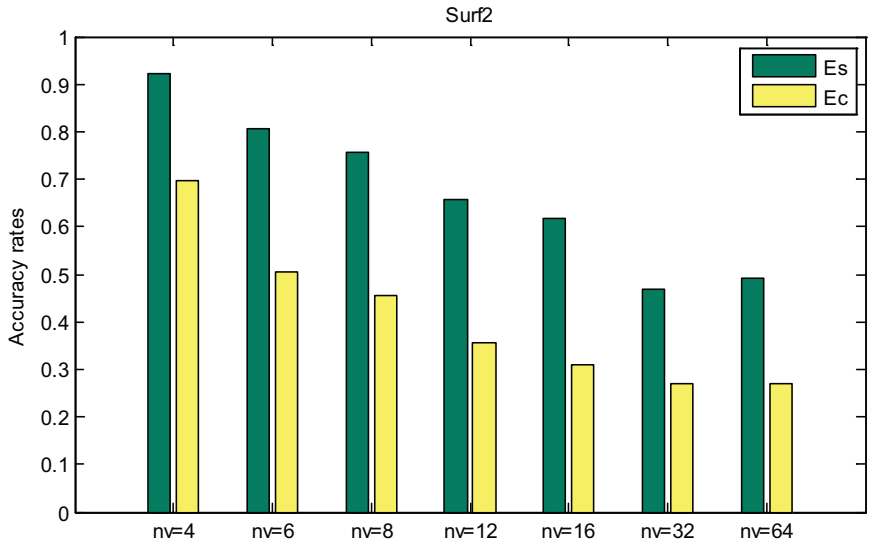


Figure 12 Area and shape accuracies measured for *Surf2* for various number of virtual boundary nodes.

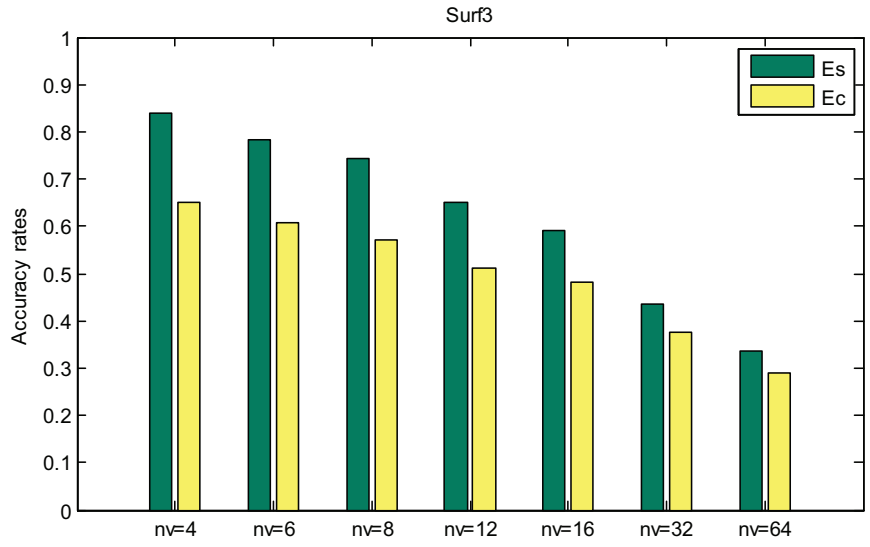


Figure 13 Area and shape accuracies measured for *Surf3* for various number of virtual boundary nodes.

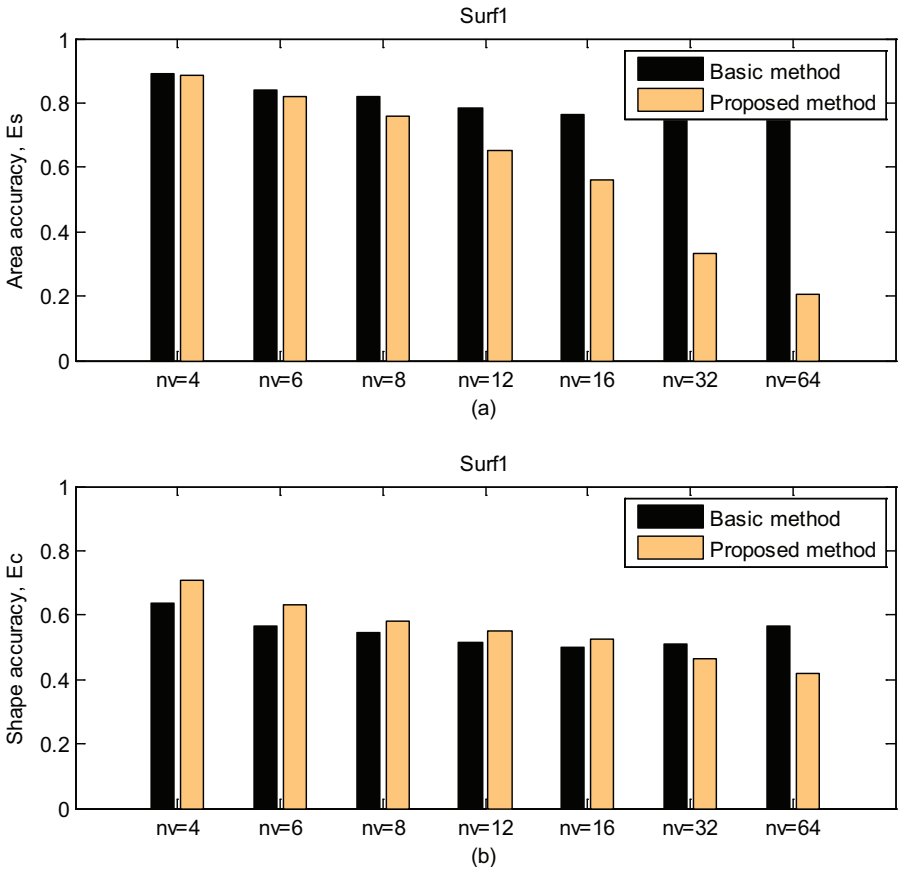


Figure 14 Flattening comparison for *Surf1* regarding area accuracy (a) and shape accuracy (b).

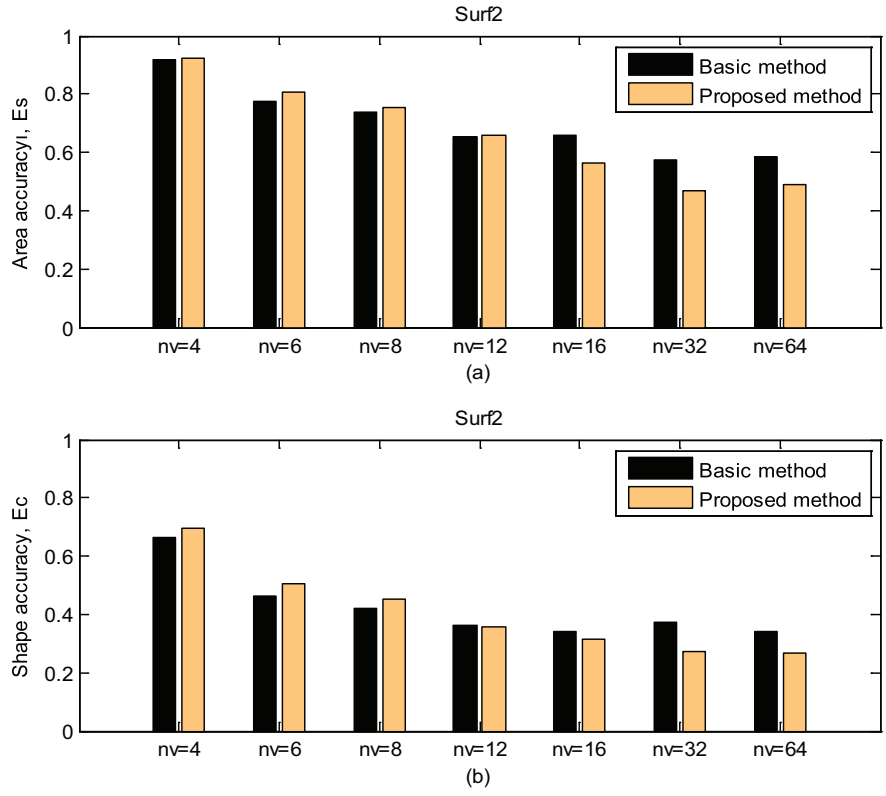


Figure 15 Flattening comparison for *Surf2* regarding area accuracy (a) and shape accuracy (b).

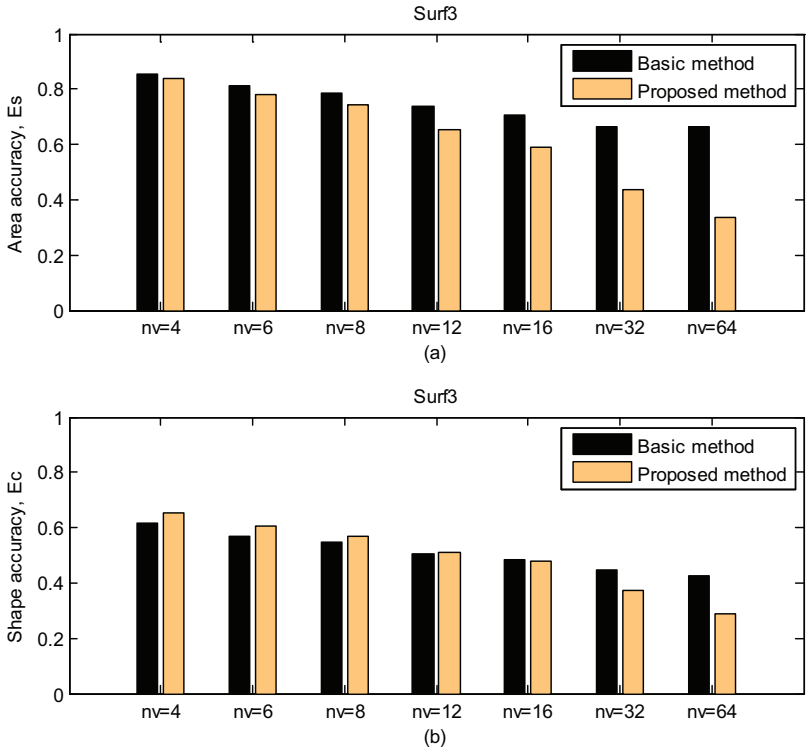


Figure 16 Flattening comparison for *Surf3* regarding area accuracy (a) and shape accuracy (b).

Table 1 Area and shape accuracies measured for the three sample surfaces with various number of virtual boundary nodes.

Sample Surface	***Surf1***		***Surf2***		***Surf3***	
Number of virtual boundary nodes	**Area accuracy, E_S**	**Shape accuracy, E_C**	**Area accuracy, E_S**	**Shape accuracy, E_C**	**Area accuracy, E_S**	**Shape accuracy, E_C**
$n_v = 4$	0.8875	0.7098	0.9211	0.6974	0.8400	0.6521
$n_v = 6$	0.8183	0.6337	0.8056	0.5049	0.7825	0.6071
$n_v = 8$	0.7570	0.5825	0.7565	0.4543	0.7420	0.5700
$n_v = 12$	0.6544	0.5504	0.6579	0.3548	0.6513	0.5104
$n_v = 16$	0.5627	0.5231	0.6178	0.3107	0.5911	0.4810
$n_v = 32$	0.3312	0.4664	0.4685	0.2704	0.4341	0.3745
$n_v = 64$	0.2038	0.4202	0.4906	0.2697	0.3351	0.2905

Table 2 Flattening accuracies of the proposed and basic methods for *Surf1*.

Sample Surface: *Surf1*					
		Basic Method		**Proposed Method**	
100 iterations	**Number of virtual boundary nodes**	**Area accuracy, E_S**	**Shape accuracy, E_C**	**Area accuracy, E_S**	**Shape accuracy, E_C**
	$n_v = 4$	0.8903	0.6355	0.8875	0.7098
	$n_v = 6$	0.8425	0.5678	0.8183	0.6337
	$n_v = 8$	0.8212	0.5462	0.7570	0.5825
	$n_v = 12$	0.7861	0.5146	0.6544	0.5504
	$n_v = 16$	0.7649	0.5020	0.5627	0.5231
	$n_v = 32$	0.7637	0.5122	0.3312	0.4664
	$n_v = 64$	0.8247	0.5676	0.2038	0.4202

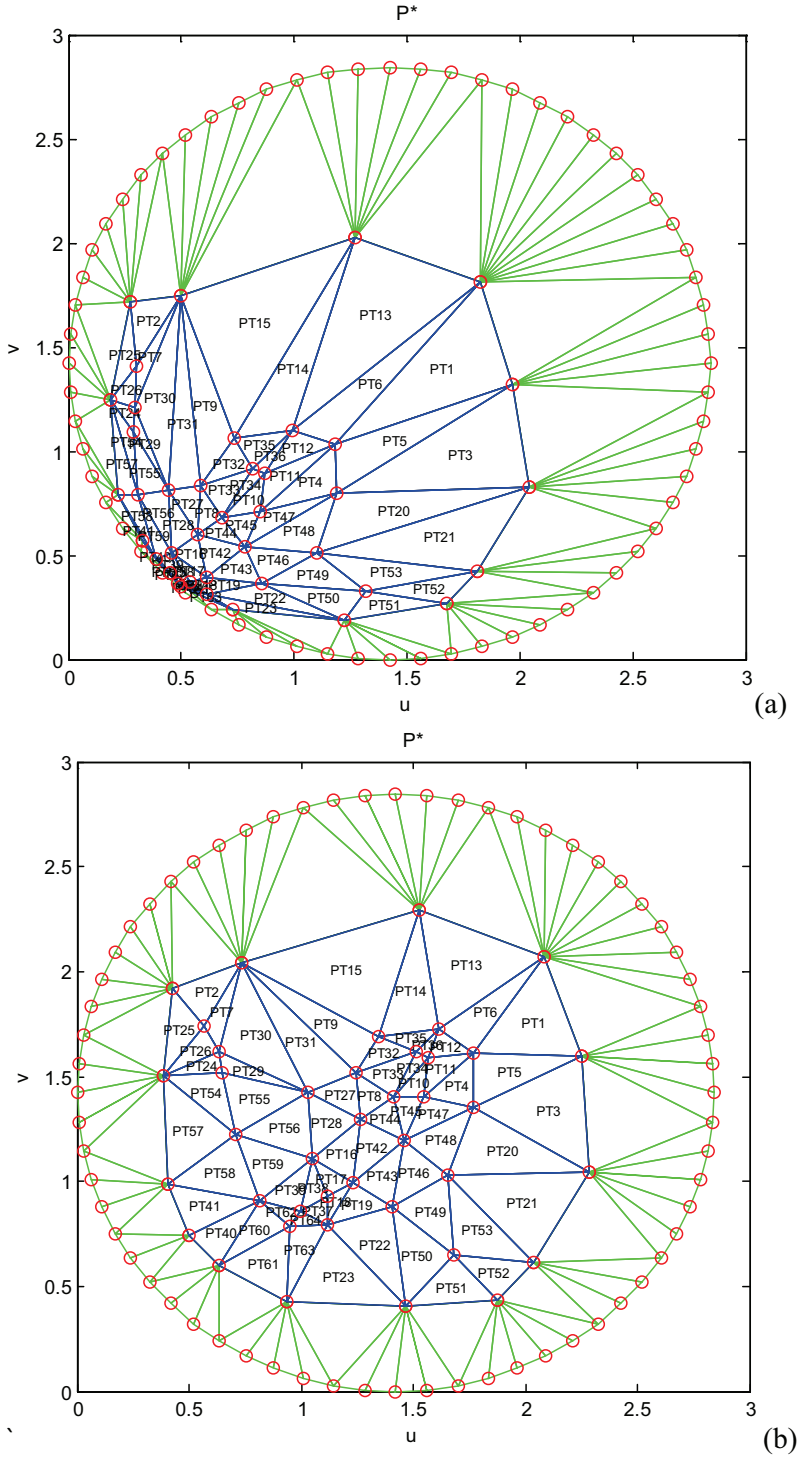


Figure 17 Flattening result of *Surf3* surface patch on the plane by basic method (a) and proposed method (b) .

7.2 Effect of Neighborhood Weights on Flattening Accuracy

In the section describing the principle of the barycentric mapping theory, the method of calculating the neighborhood weight values $\lambda_{i,j}$ for each node has been described in detail. For each node in the basic method proposed by Tutte in the drawing of graph and later adopted by Floater, this value is simply calculated to be inversely proportional to the number of neighbors. However in the proposed method, it is calculated using the inverse of the Euclidean distance between neighboring vertices of a node (with the changes given in the steps 10 and 11 in Algorithm 1). In this way, the local geometry shape in the 3D space is transferred in an efficient way to the parametric space where the planar development is realized. For this purpose, the basic method has been executed for all of the sample surface patches under the same conditions as the proposed method, and the results are given in Tables 2–4.

The area and shape accuracies given in Tables 2–4 for the three sample surface patches are also presented visually in the

Table 3 Flattening accuracies of the proposed and basic methods for *Surf2*.

Sample Surface: *Surf2*					
		Basic Method		Proposed Method	
100 iterations	Number of virtual boundary nodes	Area accuracy, E_S	Shape accuracy, E_C	Area accuracy, E_S	Shape accuracy, E_C
	$n_v = 4$	0.9164	0.6632	0.9211	0.6974
	$n_v = 6$	0.7747	0.4653	0.8056	0.5049
	$n_v = 8$	0.7382	0.4191	0.7565	0.4543
	$n_v = 12$	0.6556	0.3634	0.6579	0.3548
	$n_v = 16$	0.6597	0.3428	0.6178	0.3107
	$n_v = 32$	0.5761	0.3739	0.4685	0.2704
	$n_v = 64$	0.5825	0.3437	0.4906	0.2697

Table 4 Flattening accuracies of the proposed and basic methods for *Surf3*.

Sample Surface: *Surf3*					
		Basic Method		Proposed Method	
100 iterations	Number of virtual boundary nodes	Area accuracy, E_S	Shape accuracy, E_C	Area accuracy, E_S	Shape accuracy, E_C
	$n_v = 4$	0.8535	0.6141	0.8400	0.6521
	$n_v = 6$	0.8148	0.5669	0.7825	0.6071
	$n_v = 8$	0.7834	0.5456	0.7420	0.5700
	$n_v = 12$	0.7403	0.5042	0.6513	0.5104
	$n_v = 16$	0.7082	0.4866	0.5911	0.4810
	$n_v = 32$	0.6665	0.4473	0.4341	0.3745
	$n_v = 64$	0.6623	0.4253	0.3351	0.2905

bar graph in Figs. 14–16. The number of virtual boundary nodes is increased from 4 to 64 at certain intervals to provide the virtual boundary polygon in various resolutions. In each case, the initial projection results of the basic method and the proposed method for the surface patch appear in the bar graph in terms of area and shape accuracy.

It is seen from both Fig. 14 and Table 2 that the difference in area accuracy starts from 0.8903 and decreases to 0.8247 in the basic method, while in the proposed method it starts from 0.8875 and decreases to 0.2038 for this sample surface patch. Similarly, the difference in shape accuracy starts from 0.6355 in the basic method and decreases to 0.5676, while in the proposed method it starts from 0.7098 and decreases to 0.4202.

It is seen from both Fig. 15 and Table 3 that the difference in area accuracy starts from 0.9164 and decreases to 0.5825in the basic method, while in the proposed method it starts from 0.9211 and decreases to 0.4906 for *Surf2* surface patch. Similarly, the difference in shape accuracy starts from 0.6632 in the basic method and decreases to 0.3437, while in the proposed method it starts from 0.6974 and decreases to 0.2697.

It is seen from both Fig. 16 and Table 4 that the difference in area accuracy starts from 0.8535 and decreases to 0.6623 in the basic method, while in the proposed method it starts from 0.8400 and decreases to 0.3351 for *Surf2* surface patch. Similarly, the difference in shape accuracy starts from 0.6141 in the basic method and decreases to 0.4253, while in the proposed method it starts from 0.6521 and decreases to 0.2905.

The coordinates of the nodes on the $u - v$ parametric space are plotted in Fig. 17, with the projection of the sample surface for $n_v = 64$ by the basic method and the proposed method. All these results show that the proposed method is better than the basic method (for the same iteration number in finding the inverse matrix by iterative methods) regarding flattening accuracies. The proposed method produces planar equivalents with relatively high fidelity to their original surfaces when compared to those of the basic method.

8. CONCLUSION

This paper proposes an effective approach for flattening convex-shaped mesh surface patches. The proposed method exploits barycentric mapping theory and a dynamic virtual boundary approach to improve initial flattening result. Barycentric mapping, also known as convex-combination approach is a base method suitable for realizing mappings of convex-shaped mesh surfaces. A dynamic virtual boundary is efficaciously incorporated so as to reduce the distortions of the triangles near the real boundary of the surface. In the proposed approach, accuracy of base barycentric mapping method is improved by using the inverse of the Euclidean distance between neighboring vertices of a node, which is normalized by the total cost of its neighbors. In this way, the local geometry of the surface in 3D the space is transferred efficiently to the parametric space where the planar development is realized. The experimental results prove that the proposed approach effectively improves flattening results regarding area and shape accuracy. In a future work, initial mapping results obtained may further be enhanced by incorporating an energy-based flattening approach to release strain energy inherent in them.

ACKNOWLEDGEMENTS

This article is a part of doctoral study completed in the Department of Computer Engineering, Graduate School of Applied and Natural Sciences at Istanbul Commerce University.

REFERENCES

1. Wang, CCL (2013). Geometric *Modeling and Reasoning of Human-Centered Freeform Products*.London, Springer-Verlag.
2. Carmo, MP (1976). *Differential Geometry of Curves and Surfaces*.Englewood Cliffs, Prentice-Hall.
3. Su, K, Cui, L, Qian, K., Lei, N. et al. (2016).Area-preserving mesh parameterization for poly-annulus surfaces based on optimal mass transportation. *Computer Aided Geometric Design*,**46**76–91.
4. Zheleznyakova, AL (2015). Molecular dynamics-based triangulation algorithm of free-form parametric surfaces for computer-aided engineering.*Computer Physics Communications*,**190**1–14.
5. Yang, YJ, Zeng, W and Meng, XX (2016).Conformal freeform surfaces. *Computer Aided Design*,**81**48–60.
6. Abdul-Rahman, HS, Lou, S, Zeng, W, Jiang, X. et al. (2016). Freeform texture representation and characterisation based on triangular mesh projection techniques.*Measurement*,**92**172–182.
7. Azariadis P andSapidis N (2005).Planar development of digital free-form surfaces.*Advances in geometric modeling*.Wiley.
8. Zhan, W. (2007). Research on Pattern Flattening for Parts with Complex Surfaces, M.Sc. Thesis, Nanjing University, Aeronautics & Astronautics, China.
9. Xu, R, Liu, X andJ. Chen (2002). Development and status of surfaces flattening method.*Die and Mould Technology*, **5**15–18.
10. Boddulur, RMCandRavani, B (1993). Design of developable surfaces using duality between plane and point geometries.*Computer-Aided Design*, **25**(10)621–632.
11. Cho, W, Patrikalakis, NM and Peraire, J (1998). Approximate development of trimmed patches for surface tessellation.*Computer-Aided Design*, **30**(14) 1077–1087.
12. Ping, XI (1997). Geometric approach of 3D surface Development.*Chinese Journal of Computers*, **4**315–322.
13. Ping, X, Zhang, H, Xiong, X and Wei, G (1998).Geometric arithmetic of development of sheet metal parts.*Journal of Engineering Graphics*, **3**94–100.
14. Desbrun, M, Meyer, M and Alliez, P (2002). Intrinsic parameterizations of surface meshes.In *Computer graphics forum*,**21**(3) 209–218.
15. Floater, MS (2003). Mean value coordinates.*Computer Aided Geometric Design*, **20**(1) 19–27.
16. Floater, M (2003). One-to-one piecewise linear mappings over triangulations. *Mathematics of Computation*, **72**(242)685–696.
17. Lee, Y, Kim, HS, and Lee, S (2002). Mesh parameterization with a virtual boundary.*Computers & Graphics*, **26**(5) 677–686.
18. Yang, YJ, Zeng, W, Yang, CL, Deng, B, et al. (2013). An algorithm to improve parameterizations of rational Bézier surfaces using rational bilinear reparameterization.*Computer-Aided Design*, 45(3) 628–638.
19. Liu, C, Luo, Z, Shi, X, Liu, F et al. (2013). A fast mesh parameterization algorithm based on 4-point interpolatory subdivision. *Applied Mathematics and Computation*, **219**(10) 5339–5344.
20. Yoshizawa, S, Belyaev, A and Seidel, HP (2004). A fast and simple stretch-minimizing mesh parameterization.*Shape Modeling Application Proceedings*, 200–208.
21. Botsch, M, Kobbelt, L, Pauly, M, Alliez, P, et al. (2010).*Polygon mesh processing*,New York, CRC press.
22. Tutte, W (1960). Convex Representation of Graphs.*Proceedings of the London Mathematical* Society, **10** 474–483.
23. Tutte, W (1960). Parametrization and Smooth Approximation of Surface Triangulations.*Computer Aided Geometric Design*,**14**474–483.
24. Tutte, W (1963). How to draw a graph.*Proceedings of the London Mathematical Society,*13(1) 743–768.
25. Yavuz, E andYazici, R (2018). A dynamic neural network model for accelerating preliminary parameterization of 3D triangular mesh surfaces", *Neural Computing and Applications*, 1–11.
26. Wang, CC,Smith SS, Yuen, MM (2002). Surface flattening based on energy model.*Computer-Aided Design*, **34**(11)823–833.