









# Editable texture map generation and optimization technique for 3D visualization presentation

Tsung-Chien Wu <sup>a</sup>, Jiing-Yih Lai <sup>a</sup>, Watchama Phothong <sup>a</sup>, Douglas W. Wang <sup>b</sup>, Chao-Yaug Liao <sup>a</sup> and Ju-Yi Lee <sup>a</sup>

<sup>a</sup>National Central University, Taoyuan, Taiwan; <sup>b</sup>Ortery Technologies, Inc., Taiwan

## ABSTRACT

The mesh parameterization method has been widely used in computational geometry. It maps a three-dimensional (3D) model onto the parametric (UV) domain, on which various applications can be developed, such as texture mapping. The texture mapping technique is commonly used to create the 3D color model of an object by combining a 3D model with the object's texture map. However, the resolution and quality of the texture on the color model is important and requires careful consideration. The purpose of this study is to develop an integrated method for generating the 3D color model of an object based on the conformal mesh parameterization and a technique for direct texture mapping from the object images. Furthermore, an optimization process in texture mapping is developed to maintain the photo consistency at the transition of different images on the 3D color model. The proposed 3D color model is integrated with a 3D visualization method for a new easier and smoother presentation mode for 3D product presentation in e-commerce applications. Several realistic examples are presented to demonstrate the feasibility of the proposed method.

## KEYWORDS

Conformal Mapping; Direct Texture Mapping; Mesh Parameterization; Pixel Extraction; Triangular Model

## 1. Introduction

Two-dimensional (2D) images of an object are commonly used for product presentation in e-commerce, mainly because these images can reveal the object detail in high quality and are easy to process. However, only limited viewing angles of an object can be observed from 2D images. Three-dimensional (3D) visualization is an alternative technique for product presentation, in which multiple 2D images showing different viewing angles are integrated. The user can orient a 2D image at a given viewing angle via a viewing interface. However, in such a presentation, the user can only view images that were captured beforehand. Further, owing to limited angles recorded, the orientation process is not fluent enough. In addition, the actual 3D shape and dimensions of the object cannot be obtained using this approach. A 3D model combined with color texture technology, called 3D color model hereafter, is an alternative approach for product presentation in e-commerce. Various techniques can be employed to create a texture map for a 3D model. However, it could be necessary to edit the texture once it is generated. Therefore, it is required to generate an editable texture so that the editor can easily recognize the editing part and perform the required editing.

Furthermore, the photo inconsistency at the transition of different image resources could be a problem as it may cause the distortion of the 3D color model. Therefore, for e-commerce applications, the quality of the texture should be investigated.

The mesh parameterization technique in computational geometry provides several practical applications. Sheffer et al. [16] and Hormann et al. [9] introduced and summarized several typical methods of mesh parameterization and its applications, e.g. texture mapping, normal mapping, detail transfer, morphing, mesh completion, editing, database, remeshing, and surface fitting. The available techniques for mesh parameterization can be divided into types relating to distortion minimization, fixed or free boundary, or numerical complexity. For distortion minimization, an objective function can be formulated in terms of angles, areas or distances, and it is minimized to yield the optimized mapping of the model from the 3D domain to the parametric domain (called UV domain hereafter). The fixed boundary can be obtained by a simple formulation, allowing for an easy solution, but, the distortion in parameterization is quite large. In contrast, the free boundary has less distortion in parameterization, but obtaining the solution is

**CONTACT** Tsung-Chien Wu  [rabbit94577@gmail.com](mailto:rabbit94577@gmail.com); Jiing-Yih Lai  [jylai@ncu.edu.tw](mailto:jylai@ncu.edu.tw); Watchama Phothong  [p\\_watchama@hotmail.com](mailto:p_watchama@hotmail.com); Douglas W. Wang  [dwmwang@gmail.com](mailto:dwmwang@gmail.com); Chao-Yaug Liao  [cyliao@ncu.edu.tw](mailto:cyliao@ncu.edu.tw); Ju-Yi Lee  [juyilee@ncu.edu.tw](mailto:juyilee@ncu.edu.tw)

time-consuming in computation because the boundary is considered as part of the solution. Numerical complexity is divided into linear and nonlinear methods. The nonlinear method is complex and requires more computational time, but it yields less distortion in the result.

The angle-preserving approach, also called conformal mesh parameterization, aims to minimize the angle distortion when unwrapping a 3D model onto the UV domain. Eck et al. [4] proposed a method to convert an arbitrary mesh to a multiresolution form via a harmonic map. The boundary of the harmonic map is fixed and the numerical complexity is linear. Floater [5] proposed an algorithm based on discrete harmonic mapping to preserve the shape in mapping. The positive and symmetric weights were considered to ensure that the parameterization is bijective. The boundary of the shape-preserving method is fixed and the numerical complexity is linear. Floater [6] further developed a method for computing the harmonic map by setting mean-value weights. This approach can yield a planar parameterization with less angle distortion. The boundary of this method is fixed and the numerical complexity is linear. Lévy et al. [10] solved for the meshes on the UV domain based on a least-squares approximation of the Cauchy-Riemann equations. This method can minimize both angle and area distortion, and also establishes the topology of the planar meshes. Desbrun et al. [3] proposed a method, called instinct parameterization, to minimize angle distortion. These two methods have free boundaries and linear numerical complexity. The free boundary can yield less distortion in the UV domain.

Sheffer et al. [14] proposed a mesh parameterization algorithm, called angle-base flattening, to optimize the angles on the UV domain. The topology of triangular meshes is set as constraints in this method. Thus, it can maintain the correctness of the mesh topology on the UV domain. Sheffer et al. [15] proposed another method to make the optimization process more efficient. In addition, the hierarchical algorithm was employed to deal with the case of a large amount of triangular meshes. The computational efficiency can be increased while dealing with this kind of mesh. Zayer et al. [18] proposed a method to apply linear equations for solving the optimization problem. Linear equations were derived from the angle-base flattening approach, in which topological constraints were specified. Zigelman et al. [19] developed an algorithm to generate a geodesic distance map. It can minimize the distance distortion on the UV domain. The texture information can be preserved on both 3D and UV domains. Degener et al. [2] employed an energy function to minimize the angle and area distortion on the UV domain. Mesh topology errors, such as face flip, were

prevented as well. The aforementioned approaches are all free boundary and non-linear parameterization.

The technique of texture map generation not only deals with mesh parameterization, but also solves the texture transferring problem. Niem et al. [12] proposed a procedure of texturing the meshes that includes grouping the meshes using the camera information to find the most appropriate image source, filtering the boundary between two different groups to minimize the color inconsistency, and synthesizing the invisible meshes using the neighboring color. Genç et al. [8] proposed a method to extract the pixels and render the texture dynamically. The extraction is performed by scanning the pixels horizontally and rendering every color onto the meshes. Baumberg [1] proposed an algorithm to process the color difference from two different images using a blending method. The images were filtered into high and low bands. The low band images were averaged to minimize the color difference, whereas the high band images were kept to maintain the outline of the object boundary.

In this study, a method in accordance with conformal mesh parameterization is developed for unwrapping 3D triangular meshes onto the UV domain, and an integrated process is proposed for direct texture mapping. The proposed method can generate an editable texture map for further manual editing. Additionally, the photo inconsistency problem in traditional 3D color model is improved by providing an algorithm to detect and remove the inconsistency in photo at the transitions of different images. The proposed direct texture mapping can be divided into three phases: grouping of 3D triangles, color pixel extraction from object images, and color pixel placement on the texture map. Grouping of 3D triangles is to determine the most appropriate object image for each triangle on the UV domain. The camera viewing angle for each object image and the normal vector for each triangle are known. The grouping rules can be determined by several factors, which will be described later. Color pixel extraction from object images and color pixel placement on the texture map are implemented simultaneously. By projecting 3D triangles onto the image domain and evaluating the relationship of them on the image and UV domains, a texture map is eventually generated to describe a 3D color model. The most critical problem in texture mapping is that the junctions of different images on the texture are usually not properly connected. An algorithm is proposed to deal with the photo inconsistency problem at such junctions. Several examples are presented to demonstrate the feasibility of the proposed method. Several important characteristics of the proposed method in e-commerce application are also addressed.

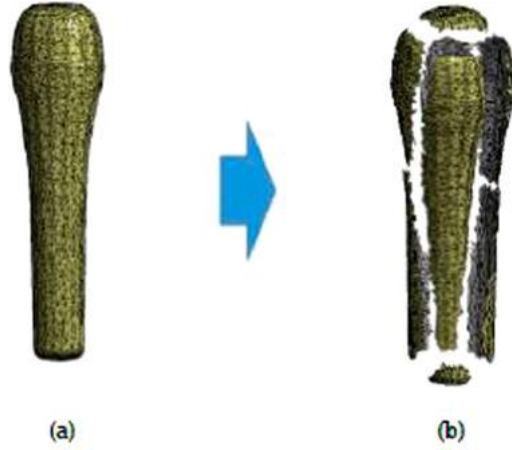


**Figure 1.** Visualization of the steps of the proposed method, (a) Input object images, camera information, and object meshes, (b) divide 3D meshes into segments for mesh parameterization, (c) unwrap 3D meshes onto the UV domain to yield a UV map, (d) grouping of 3D meshes, and (e) final texture map and 3D color model.

To generate a 3D color model, the object images must be captured sequentially in a controlled environment. In addition, the camera information should be obtained by capturing the calibration mat and performing the calibration process [11]. The object's 3D model is constructed by the shape-from-silhouette (SFS) algorithm [13], and the shape and surface of the 3D model are optimized by silhouette and smoothing factors [13], [17]. The aforementioned process can yield a 3D model for texture mapping, where the 3D model is composed of triangular meshes. Fig. 1 is the steps of the proposed method for generating an editable texture map and 3D color model. The first step is to input the optimized 3D triangular meshes, the object images and camera information. Next, the 3D meshes are separated into several segments in accordance with the requirement. Subsequently, each mesh segment is unwrapped onto the UV domain and all planar meshes on the UV domain are packed together to form a UV map. Furthermore, the triangles on 3D meshes are grouped, with each group of meshes unwrapping onto an image with the most appropriate viewing angle. Finally, the color extraction and pixel placement are implemented simultaneously. The pixels covered by each triangle on the UV map are filled in using the pixels extracted from the corresponding image. The 3D color model is then output and saved as an OBJ file. The techniques used to achieve these tasks are described below.

## 2. Editable conformal mesh parameterization

The editable mesh parameterization is based on a separated 3D model; the regions of interest are separated as individual segments and are editable when they are



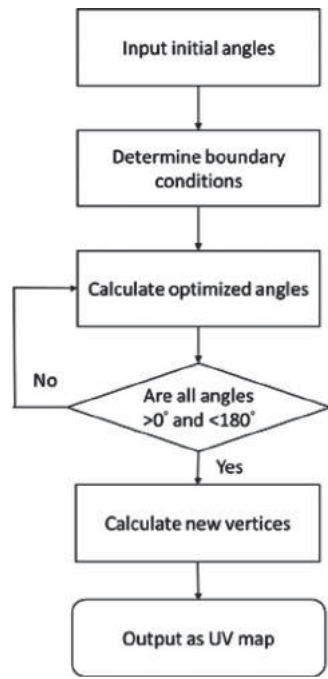
**Figure 2.** 3D meshes subdivision, (a) original 3D meshes, and (b) segments of 3D meshes separated manually.

unwrapped onto the UV domain. Therefore, the 3D meshes should be separated into segments either manually or automatically before mesh parameterization is implemented. Fig. 2 shows an example of separating 3D meshes manually. The meshes are separated based on two criteria. First, a segment to edit should be separated. Second, each segment should be simple in geometry for avoiding distortion in unwrapping. Mesh parameterization is implemented one by one on all segments and each is unwrapped onto the UV domain. The basic idea of the proposed conformal mesh parameterization is to unwrap 3D meshes onto the UV map while preserving the angles of each triangle on the UV domain. A critical issue of the preservation is that all angles of 2D meshes on the UV domain cannot be kept the same as those of 3D meshes. Therefore, an optimization problem is formulated to minimize the deviation of angles and determine the optimize vertices on 2D meshes [14–15], [18]. Fig. 3 shows the flowchart for the optimization and parameterization of 2D meshes on the UV domain, which can be divided into the following four steps: (1) input initial angles, (2) determine boundary conditions, (3) calculate optimized angles, and (4) calculate new vertices in accordance with optimized angles.

First, the angles of all 3D meshes are regarded as the initial angles of the 2D meshes on the UV domain. However, the summation of all angles surrounding a vertex on the 3D domain may exceed  $360^\circ$ , whereas it must be  $360^\circ$  on the UV domain. Therefore, the following equation is employed to adjust each angle on the UV domain so that all angles surrounding a vertex can become  $360^\circ$ :

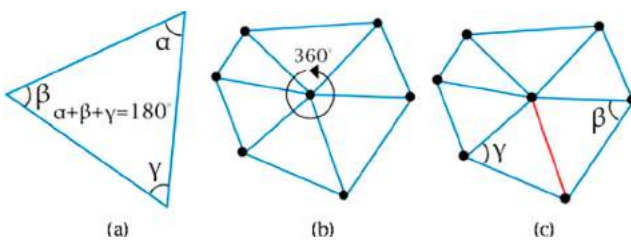
$$\theta_{Adjust} = \theta_{Original} * \left( \frac{360}{\theta_{Sum-Vertex}} \right) \quad (1)$$

where  $\theta_{Adjust}$  denotes the adjusted angle,  $\theta_{Original}$  denotes the original angle, and  $\theta_{Sum-Vertex}$  denotes the summation



**Figure 3.** Flowchart of the optimization and parameterization of 2D meshes on the UV domain.

of all angles surrounding a vertex. There are also some conditions, such as flip in triangles, degenerate triangles, and non-manifold edges, on 2D meshes, which make 2D meshes deviate from 3D meshes. Therefore, additional constraints on angles must be specified to maintain the accuracy of the topology and prevent the occurrence of irrational meshes. In this way, the topology of the meshes on the UV domain after angle adjustment must be kept the same as that on the 3D domain. Three constraints on angles are specified in the proposed algorithm, as described below. The first constraint is to satisfy the triangle consistency. The summation of all angles on a 2D triangle should be  $180^\circ$  (Fig. 4(a)). This constraint is



**Figure 4.** Constraints for conformal mesh parameterization, (a) summation of angles on one triangle should be  $180^\circ$ , (b) summation of angles surrounding a vertex should be  $360^\circ$ , and (c) the lengths of an edge evaluated from two neighboring meshes, respectively, should be equal.

given as:

$$\sum_{i=1}^3 \varepsilon_i = 180 - \sum_{i=1}^3 \theta_i \quad (2)$$

where  $\theta_i$  denotes the angle of a 2D triangle, and  $\varepsilon_i$  denotes the error of the angle on the  $i^{\text{th}}$  2D triangle. Equation (2) describes the total angular errors of all triangles on 2D meshes.

The second constraint is to satisfy the vertex consistency, in which the summation of all angles surrounding an inner vertex should be  $360^\circ$  (Fig. 4(b)). This constraint is given as:

$$\sum_{i=1}^d \varepsilon_i = 360 - \sum_{i=1}^d \theta_i \quad (3)$$

where  $\theta_i$  denotes the angle on the  $i^{\text{th}}$  inner vertex,  $\varepsilon_i$  denotes the error of the angle on the  $i^{\text{th}}$  vertex, and  $d$  denotes the number of inner vertices. Equation (3) describes the total angular error of all inner vertices. It is noted that boundary vertices are not counted in Eq. (3).

The third constraint is to satisfy the wheel consistency, in which the lengths of all edges neighboring a vertex should be equal. This constraint is given as:

$$\begin{aligned} & \sum_{i=1}^d \cot(\beta_i) \varepsilon_{\beta_i} - \cot(\gamma_i) \varepsilon_{\gamma_i} \\ &= \sum_{i=1}^d \log(\sin(\beta_i)) - \log(\sin(\gamma_i)) \end{aligned} \quad (4)$$

In Eq. (4), the lengths of an edge evaluated from two neighboring meshes are computed first (Fig. 4(c)). The lengths of all edges adjacent to a vertex are then computed.

The aforementioned three constraints can be employed to guarantee the topological consistency of the 2D meshes with the 3D meshes. The triangle consistency, vertex consistency, and wheel consistency can prevent the degenerate triangle, triangle flip, and non-manifold edge correspondingly when unwrapping 3D meshes onto the UV domain to generate 2D meshes. The third step of the optimization is to calculate the minimized error. As mentioned before, the conformal mesh parameterization is obtained by minimizing the error between the meshes on the 2D and 3D domains. From the constraint conditions, a set of initial errors on the UV domain has already been determined. By minimizing this set of errors, the optimized angles can be reached. The proposed algorithm employs a linear system to solve for the optimized angles. The initial angles (Eq. (1)) and the constraint conditions (Eqs. (2) to (4)) can be combined to formulate a linear

system as follows:

$$\begin{bmatrix} 10101001 \\ \vdots \\ 10101001 \\ \vdots \\ \cot(\beta) \ 0 \ \cot(\gamma) \ 0 \\ \vdots \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \vdots \\ \epsilon_n \end{bmatrix} = \begin{bmatrix} 360 - (\theta_1 + \dots + \theta_d) \\ \vdots \\ 180 - (\theta_1 + \dots + \theta_3) \\ \vdots \\ (\log(\sin(\beta)) - \log(\sin(\gamma))) + \dots \end{bmatrix} \quad (5)$$

Eq. (5) is essentially  $Ax = b$ , in which the errors  $\epsilon_i, i=1 \dots n$  are minimized to yield the optimized angles  $\theta_i$ . After the calculation, the optimized angles can be obtained using the following expression:

$$\theta_{New} = \theta_{Initial} + \epsilon \quad (6)$$

where  $\theta_{new}$  is the new angle obtained, and  $\theta_{initial}$  is the angle of previous trial. As some of the angles obtained may be larger than  $180^\circ$  or less than  $0^\circ$ , the optimization process should be implemented iteratively. The iteration stops when all angles  $\theta_{new}$  are within the range  $0^\circ - 180^\circ$ . The last step of the mesh parameterization is to calculate the new positions of all vertices in accordance with the optimized angles obtained. The calculation is based on the triangle similarity. As depicted in Fig. 5, let three angles of a triangle be  $\alpha_1, \alpha_2$ , and  $\alpha_3$ . Assume that two vertices  $P_1$  and  $P_2$  are known. The unknown vertex ( $P_3$ ) can be evaluated as follows:

$$P_3^i - P_1^i = \frac{\sin \alpha_2^i}{\sin \alpha_3^i} R_{\alpha_1^i} (P_2^i - P_1^i) \quad (7)$$

where  $R$  denotes the rotation matrix. In this way, the calculation of new vertices on the UV domain can be applied using the least-squares approximation, as shown

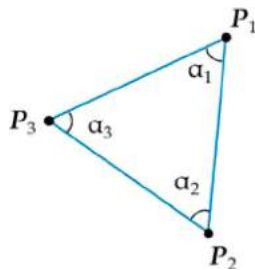


Figure 5. Vertices and angles distribution on a single triangle.

in Eqn. (8).

$$\min \sum_i \left( P_3^i - P_1^i - \frac{\sin \alpha_2^i}{\sin \alpha_3^i} R_{\alpha_1^i} (P_2^i - P_1^i) \right)^2 \quad (8)$$

Eq. (8) employs two known vertices  $P_1$  and  $P_2$  to optimize the remaining unknown vertex  $P_3$ . For all 2D meshes, if the first two vertices on a mesh can be determined first, the remaining vertices can be calculated using the least-squares approximation [10], which is formulated as a linear system  $Ax = b$ . After solving this linear equation, all vertices can be obtained. The topology of all vertices on 2D meshes can be maintained correctly.

### 3. Direct texture mapping algorithm

The 3D color model is generated from a 3D model covered with texture that stores the color information on a texture map. The 3D model with the texture map can make the model more realistic. However, the resolution and quality of the texture plays an important role regarding the feasibility of this technique in practical applications. As mentioned before, the object images are taken sequentially in a control environment. High-quality object images can be obtained by an image-taking

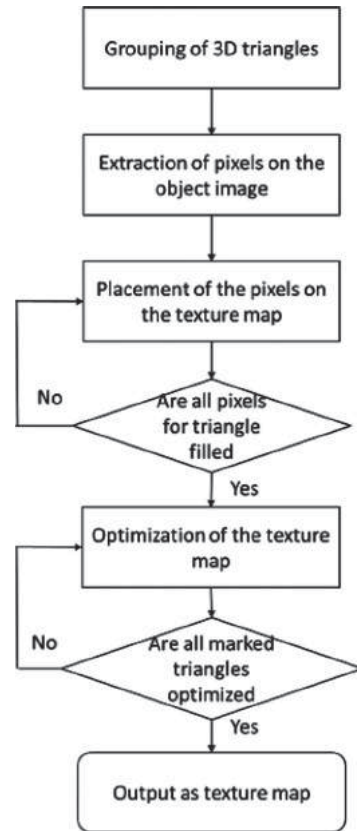
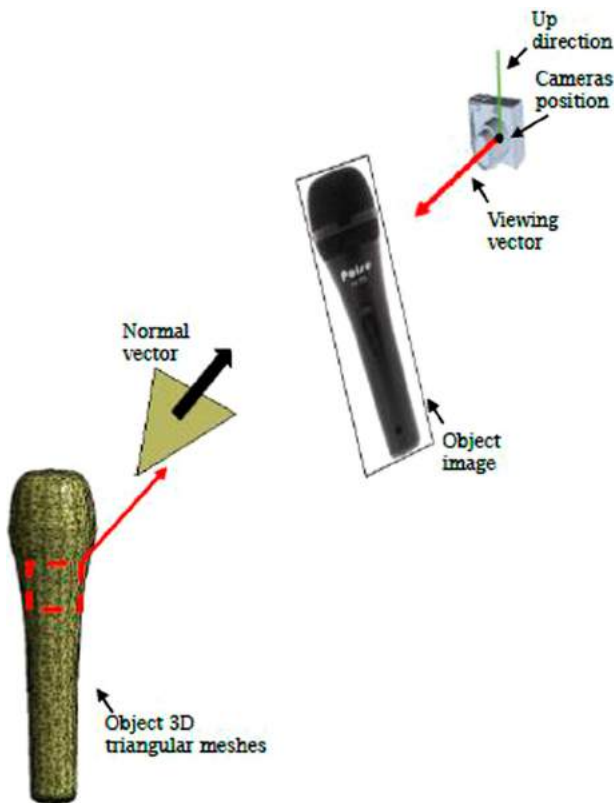


Figure 6. Flowchart of the proposed direct texture mapping algorithm.

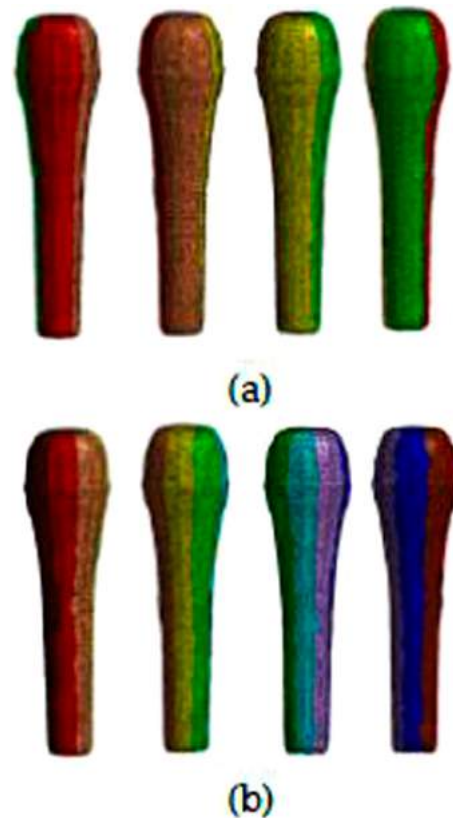


**Figure 7.** The normal vector and the camera viewing vector for grouping.

process. Thus, the main idea of direct texture mapping is to texture the 3D model using the object images directly. Fig. 6 shows the flowchart of the proposed direct texture mapping algorithm, which has following four major steps: (1) grouping of 3D triangles, (2) extraction of the pixels on the object images, (3) placement of the pixels on the texture map, and (4) optimization of the texture map.

First, a series of camera information and 3D meshes are input. The purpose of grouping 3D triangles is to allocate the triangles to the most appropriate image. In this way, the texture quality for each triangle can reach a high-quality texture. The grouping criterion is based on the angle between an image and a triangle. One component of the camera information is the looking direction, which represents the camera viewing vector. The camera viewing vector is perpendicular to the image plane. In addition, all 3D triangles have their own surface normal vectors, as shown in Fig. 7.

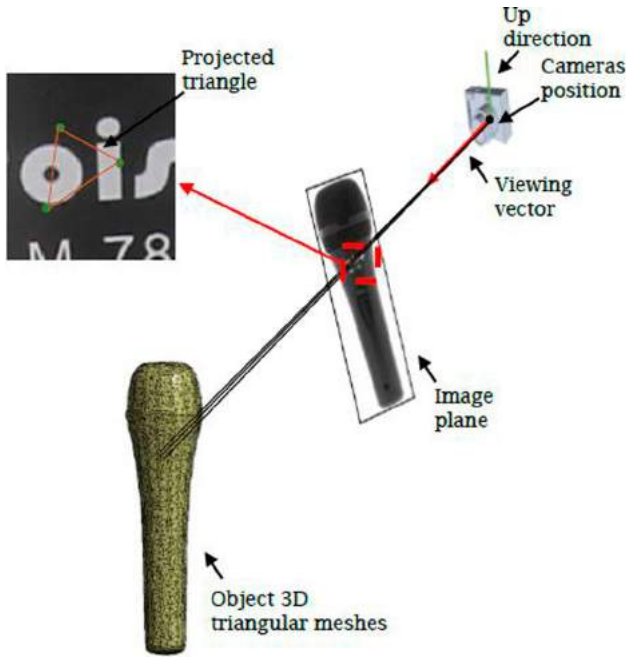
With these two vectors, the angle between a triangle and an image can be calculated. The main image of a triangle is defined as the image that has the minimum angle among all angles evaluated between the viewing vectors of the image and the surface normal vector of the triangle. The visibility is also considered while grouping the triangles. The angle between an image and a triangle should be



**Figure 8.** Result of grouping for a "Mic" sample: (a) 4 viewing images, and (b) 8 viewing images.

less than  $90^\circ$  to ensure that the image faces the front side of the triangle. Further, this triangle cannot be obstructed by other triangles on the same image. The visibility check can also prevent the occurrence of this kind of situation on other images. Fig. 8 shows one result of grouping, where all meshes of the same color are in the same group, and are textured using the same main image.

The second step is extraction of the pixels on the object images. The texture of the 3D color model comes directly from the object images. Thus, in this step, the 3D triangle should be projected back to the image plane of the main image. The projection is based on the prospective projection in accordance with the camera position to project the 3D triangle to the image plane. As Fig. 9 depicts, the triangle on the image represents the projection of one triangle on the 3D triangles. All pixels on and inside the triangle represent the texture corresponding to the 3D triangle. To generate a texture map for all 2D triangles created, the pixels of a 2D triangle on the texture map is obtained using the pixels obtained in this step. The difference is that the area of the pixels corresponding to a 2D mesh does not have the same area as the pixels found in this step. Therefore, a transformation of the pixels between two different pixel domains should be implemented.



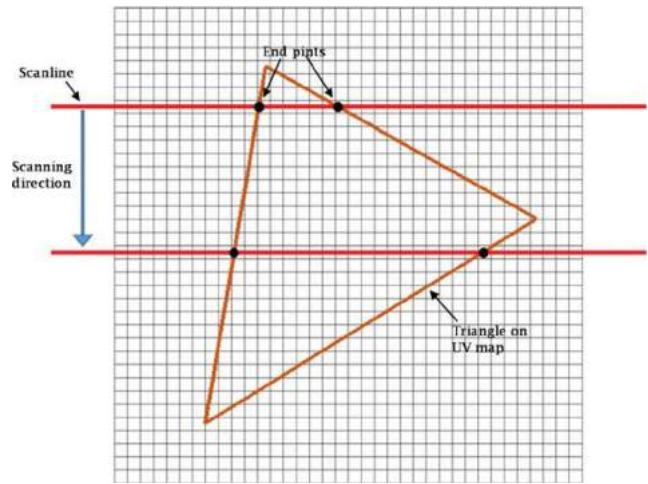
**Figure 9.** The method for projecting a 3D triangle onto an image plane.

The extraction of any pixel inside a triangle is explained below. The sequence of the image pixel is like the grid plane. It is composed of horizontal and vertical lines. Thus, a scanline method can be implemented to evaluate all pixels inside a triangle, as in Fig. 10, which shows a horizontal scanline. This scanline will intersect two triangle edges, which denotes both end points of the scanline. As long as both end points of the scanline are known, all pixels within this scanline can be evaluated in sequence. The following equation shows the computation of the pixels within both end points of a scanline:

$$\delta_x = X_2 - \left( \frac{(Y_2 - Y)(X_2 - X_1)}{(Y_2 - Y_1)} \right) \quad (9)$$

where  $(X_1, Y_1)$  and  $(X_2, Y_2)$  represent two vertices of the triangle edge,  $Y$  is the current scanline vertical value, and  $\delta_x$  denotes the pixel within both end points of the scanline.

The third step is placement of the pixels on the texture map. The pixels corresponding to each 2D triangle come from the previous step. However, the area of the pixels from the previous step are different to those that should be filled on the UV map. Therefore, the main issue is how to extract the correct pixels from the previous step and place them appropriately on the UV map. As each 2D triangle on the UV domain is different from the projected mesh on an image domain, the pixels cannot be directly placed one by one. A transformation algorithm is developed to map the pixels between two pixel domains.



**Figure 10.** The scanline method for extracting and filling the color for all pixels inside a triangle.

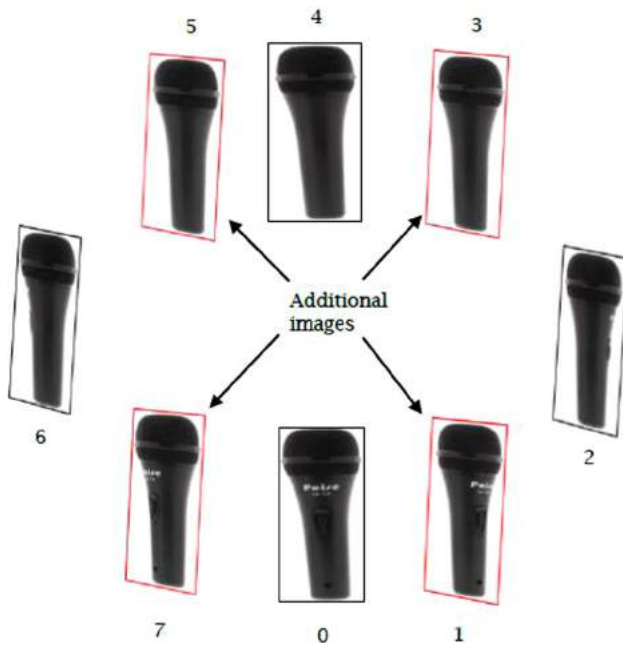
First, the three vertices on the image domain are respectively mapped onto three vertices on the UV domain. The transformation can be expressed as follows:

$$aX + bY + c = X' \quad (10)$$

$$dX + eY + f = Y' \quad (11)$$

where  $X$  and  $Y$  denote the coordinates of a vertex on the image domain, and  $X'$  and  $Y'$  denote the coordinates of a vertex on the UV domain. The unknown parameters  $a$  to  $f$  can be obtained because the coordinates of three vertices on the image domain and the UV domain are all given. Once all parameters  $a$  to  $f$  corresponding to a triangle are obtained, the colors of all pixels within the corresponding triangle can be interpolated by using Eqs. (10) and (11). With this method, all pixels of different triangles on the UV domain can be filled in with the correct color. This finally yields a texture map for all 2D meshes generated.

However, the direct texture mapping uses the most appropriate view of the image to map each pixel onto 3D meshes. The texture on 3D meshes comes from different viewing angles. Hence, the transition of different images on the texture may be inconsistent in photo. This may affect the texture quality of the 3D color model and, hence, its realism for e-commerce applications. This discontinuity issue on the texture has two main causes. First, the camera position is calculated by camera calibration. The position of the camera might not locate on the exact position of the global coordinate, resulting in a texture mapping error that can affect the continuity of the 3D color model. Second, the surface vertices of the 3D model might not lie on the surface of the real object. The texture extraction comes from projecting the surface



**Figure 11.** The additional images for texture optimization.

vertex back to the image domain to extract the texture. However, the position of each vertex on the 3D model is still not 100% correct although the model optimization has already been performed [13],[17]. The vertices lying on the boundary of two different groups would extract different colors from two different images. This kind of situation would cause the boundary discontinuity, which is also known as photo inconsistency.

The fourth step of this approach is to deal with the second problem, the photo inconsistency problem. In this step, the texture of the color model is optimized to maintain the photo consistency on the model. The basic idea of this step is to extend the layer of the image to make the boundary lie on a place where the color difference is small. However, the original number of images used was four. The difference in the viewing angle for two adjacent images is  $90^\circ$ . This might also cause the photo inconsistency problem after extending the layer. Thus, the method to handle this kind of problem is to add more images to group the triangles. In this way, the difference in the viewing angle for two adjacent images is reduced and the photo consistency can be increased. Eight images are used to optimize the texture, as shown in Fig. 11.

In Fig. 11, images 1, 3, 5, and 7 are images added for optimizing the texture. The boundary triangles between 1 and 2, 2 and 3, 5 and 6, and 6 and 7, respectively, are identified. After the identification, the color difference on these series of triangles are calculated. When the color difference on the boundary of a triangle is larger than a threshold, the image with respect to this triangle is replaced by one image added. The new image boundary

between two different image will be determined. In this way, the photo consistency on the 3D color model can be maintained, making the textured model more accurate.

#### 4. Examples and discussion

Several examples were employed to evaluate the proposed method. The inputs were the object images, camera information, and 3D triangular model of the object. The output was an editable texture map and a 3D color model. The editable texture map was used to verify the extension of the application. In addition, the results of the 3D color model before and after optimization results were compared. Furthermore, the integration of the color model and the 3D visualization viewer were presented and demonstrated. The simulations were performed on a personal computer with a 3.20 GHz CPU and 8 GB of RAM.

Fig. 12 depicts the original model and the corresponding 2D meshes on the UV domain for six examples, where the left and right images in each figure panel denote the original 3D meshes and the mapped 2D meshes, respectively. The method to generate the original 3D meshes is shown in Phothong et al. [13]. The model optimization [13], [17] and the mesh simplification [7] have been performed separately as well. The 3D meshes were separated into several regions and each region is unwrapped onto the UV domain individually by the proposed conformal parameterization method. Remarkably, the proposed method can unwrap the 3D meshes individually and minimize the angle distortion on the UV domain. Tab. 1 lists the number of vertices and faces (meshes) of the triangular model, the triangle angle stretch root-mean-square (RMS) error on the UV domain, and the CPU time required for unwrapping 3D meshes to the UV domain for the six examples in Fig. 12. As per Tab. 1, the number of faces in all models was controlled at 4,500, which will not induce any sluggishness during data download and website operation. The errors on angles for all examples were between  $1.032^\circ$ – $3.549^\circ$ , and the required CPU time for all cases was between 8.08–76.84 second.

Fig. 13 depicts the results of the 3D model combined

**Table 1.** Parameters of input 3D model, number of segments, RMS error and CPU time for the six examples.

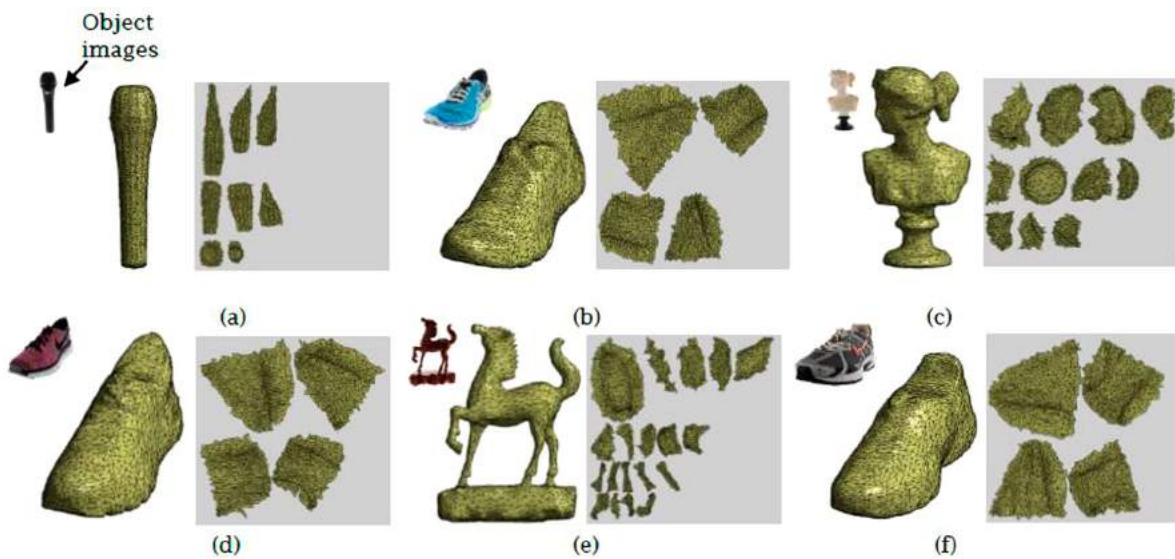
Case	3D model		Number of segments	RMS error for all angles ( $^\circ$ )	CPU time (sec)
	Vertices	Meshes			
Mug	2252	4500	6	1.032	76.84
Sport shoe			4	1.239	32.94
Cat doll			4	1.628	24.19
Sport shoe			4	1.811	24.12
Horse			19	3.549	8.08
Sport shoe			4	1.412	33.66



with the texture map as a 3D color model, where the left and right images in each figure panel denote the texture map and the 3D color model, respectively. The color texture on the texture map is extracted from the object image directly which of size  $5184 \times 3456$ . The size of the texture map image is  $4096 \times 4096$ . Thus, the quality of the original image can be kept on the texture map. In this way, the 3D color model can have a higher resolution for demonstration. As Tab. 1 shows, the numbers of vertices and meshes behind the high resolution 3D color model are only 2252 and 4500, respectively, indicating that this model can easily be displayed on a website viewer. Moreover, the texture map is editable. The part to edit or replace can be isolated and considered as an independent segment. The

user can easily modify the image or replace the pattern on the isolated segment. As Fig. 14 shows, different patterns can be added to the texture map (Fig. 14(a)) or an existing pattern can be replaced by another one (Fig. 14(b)). Figs. 14(c) and (d) depict the results of 3D color models after editing.

The texture of the 3D color model may face the problem of photo inconsistency because it comes from the image by projection and extraction. Figs. 15(a) and (b) show the discontinued parts of two 3D color models, mug and cat doll. The discontinued parts are located at the junction of two different groups of triangles. In the original method, four images from different viewing directions were used to group the triangles. But, the



**Figure 12.** The original model and the 2D meshes on the UV domain for six examples, (a) mic, (b) blue shoe, (c) statue, (d) pink shoe, (e) horse statue, and (f) gray shoe.



**Figure 13.** The 3D color model and the texture map for six examples, (a) mic, (b) blue shoe, (c) statue, (d) pink shoe, (e) horse statue, and (f) gray shoe.

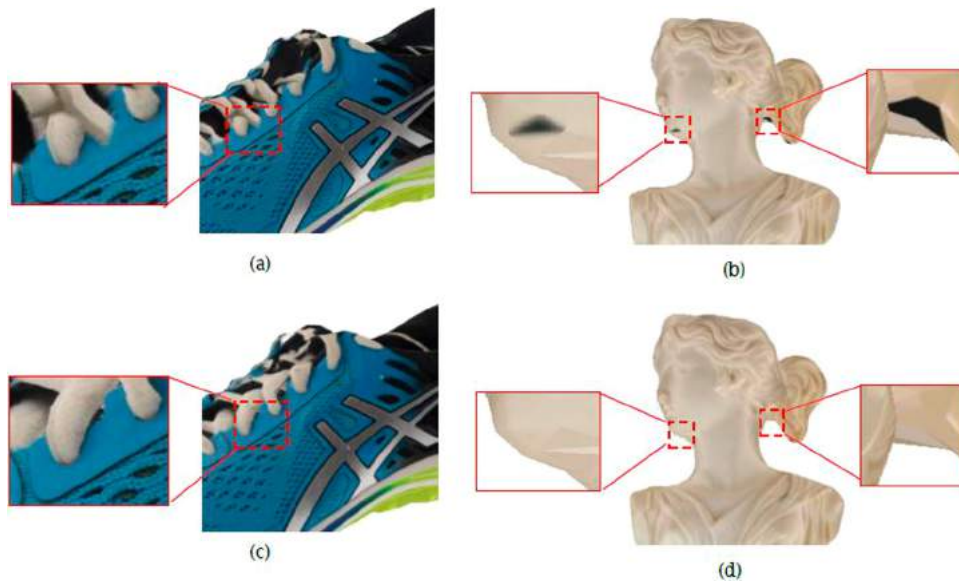
photo inconsistency problem was very serious. In the modified method, four additional images were used to minimize the photo inconsistency problem. As shown in Figs. 15(c) and (d), the color models generated by the proposed process and the texture maps are optimized. The discontinuity problem has been eliminated. The quality



**Figure 14.** The texture editing and replacing for the “Mic” and “Statue” example, (a) texture edited texture map for the Mic, (b) texture replaced texture map for the Statue, (c) texture edited 3D color model, and (d) texture edited 3D color model.

of the color model has also been improved when the 3D color model is displayed on the viewing tool.

The 3D color model can be integrated with the 3D visualization presentation as a new presentation mode. The 3D visualization presentation is integrated with a series of images. The user can drag the screen to change the image from different viewing angles. Each image represents the best quality of the picture that can be displayed. However, when dragging the screen, the rotating might not be fluent enough because limited images were captured. Therefore, the integration of the 3D color model and the 3D visualization presentation is proposed, combining the advantages of both methods. The advantages of the 3D color model are that rotating an object on the screen can be performed smoothly and the object’s image can be displayed continuously on the screen. The advantage of the 3D visualization is that the original quality of the high-resolution images can be kept. Fig. 16 shows the original presentation viewer. Fig. 16(a) depicts the screenshot in rotating, in which the low quality object image is displayed to yield a fluent rotating process. Fig. 16(b) depicts the screenshot in still, in which the original object image is displayed to yield a high-quality image presentation. However, the main problem of this technique is that the rotating is discrete as only limited viewing angles with 2D images can be accessed. To increase the viewing experience in rotating, Fig. 17 shows the new integrated presentation viewer. Fig. 17(a) depicts the screenshot in rotating, in which the 3D color model is displayed. The 3D color model can be rotated continuously. Fig. 17(b) depicts the screenshot in still,



**Figure 15.** The original and optimized 3D color model for the “blue shoe” and “statue” examples, (a) the original 3D color model for the blue shoe, (b) the original 3D color model for the statue, (c) the optimized 3D color model for the blue shoe, and (d) the optimized 3D color model for the statue.



**Figure 16.** The original presentation viewer example for the “blue shoe” example, (a) low-quality object image display when rotating, and (b) object image display when rotation has stopped.



**Figure 17.** The integrated new presentation viewer example for the “blue shoe” example, (a) object 3d color model display when rotating, and (b) object image display when rotation has stopped.

in which the original object image is displayed. With the aforementioned switching mode to display 3D color model in rotating, while 2D image in still, the new presentation method can be more realistic for e-commerce applications.

## 5. Conclusions

In this study, we proposed a method for performing conformal mesh parameterization to unwrap the mesh to a two-dimensional (UV) domain and for direct texture mapping from the extraction of the object image. The proposed algorithm employs angle-based optimization to calculate the triangle angles on the UV domain. The vertices placement was based on the least-squares approximation combined with the optimized angles. The direct texture mapping was obtained by projecting the triangles onto the image domain in accordance with the camera information and extracting the pixel to map to the UV map from the object image. However, the texture from different images might cause the photo inconsistency problem. Therefore, an optimization process was proposed to deal with this kind of problem. The additional images were used to minimize the discontinuity issue and the boundary color was checked for determining the extending criterion. Two examples have been presented to demonstrate the feasibility of the proposed texture optimization method. The texture map result of the proposed method has been presented to demonstrate the

editing extension as well. Additionally, an integrated presentation viewer was proposed to extend the application of the 3D visualization presentation and to solve the disadvantage of the original 3D visualization presentation application in e-commerce.

## ORCID

Tsung-Chien Wu <http://orcid.org/0000-0002-2299-7361>

Jiing-Yih Lai <http://orcid.org/0000-0002-0495-0826>

Watchama Phothong <http://orcid.org/0000-0002-3239-4564>

Douglas W. Wang <http://orcid.org/0000-0002-8039-5027>

Chao-Yaug Liao <http://orcid.org/0000-0001-8203-9520>

Ju-Yi Lee <http://orcid.org/0000-0002-2244-4863>

## References

- [1] Baumberg, A.: Blending Images for Texturing 3D Models, *BMVC*, 3, 2002, 5. <http://doi.org/10.5244/C.16.38>
- [2] Degener, P.; Jan M.; Reinhard, K.: An Adaptable Surface Parameterization Method, *IMR*, 3, 2003, 201–213.
- [3] Desbrun, M.; Meyer, M.; Alliez, P.: Intrinsic parameterizations of surface meshesh, *Computer Graphics Forum*, 21(3), 2002, 209–218. <http://doi.org/10.1111/1467-8659.00580>
- [4] Eck, M.; DeRose, T.; Duchamp, T.; Hoppe, H.; Lounsbery, M.; Stuetzle, W.: Multiresolution analysis of arbitrary meshes, *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 1995, 173–182. <http://doi.org/10.1145/218380.218440>
- [5] Floater, M. S.: Parametrization and smooth approximation of surface triangulations, *Computer aided geometric design*, 14(3), 1997, 231–250. [http://doi.org/10.1016/S0167-8396\(96\)00031-3](http://doi.org/10.1016/S0167-8396(96)00031-3)
- [6] Floater, M. S.: Mean value coordinates, *Computer aided geometric design*, 20(1), 2003, 19–27. [http://doi.org/10.1016/S0167-8396\(03\)00002-5](http://doi.org/10.1016/S0167-8396(03)00002-5)
- [7] Garland, M; Heckbert, P. S.: Surface simplification using quadric error metrics, *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH 97*, 1997, 209–216. <http://doi.org/10.1145/258734.258849>
- [8] Genç, S.; Atalay, V.: Texture extraction from photographs and rendering with dynamic texture mapping, *Image Analysis and Processing*, 1999, 1055–1058. <http://doi.org/10.1109/ICIAP.1999.797737>
- [9] Hormann, K.; Lévy, B.; Sheffer, A.: Mesh parameterization: Theory and practice, *ACM SIGGRAPH 2007 courses on - SIGGRAPH 07*, 2007, 1. <http://doi.org/10.1145/1281500.1281510>
- [10] Lévy, B.; Petitjean, S.; Ray, N.; Maillot, J.: Least squares conformal maps for automatic texture atlas generation, *ACM Transactions on Graphics (TOG)*, 21(3), 2002, 362–371. <http://doi.org/10.1145/566654.566590>
- [11] Liao, C. Y.; Xiong, Y. S.; Wang D.W.; Lai J. Y.; Lee J. Y.: A camera calibration process for 3D digital model reconstruction of huge objects, 2016 *Machining, Materials and Mechanical Technologies*, Matsue Terssa, Japan: 7–11 October 2016. Japan: IC3MT, 2016.

- [12] Niem, W.; Buschmann, R.: Automatic Modelling of 3D Natural Objects from Multiple Views, *Image Processing for Broadcast and Video Production*, 1995, 181–193. [http://doi.org/10.1007/978-1-4471-3035-2\\_15](http://doi.org/10.1007/978-1-4471-3035-2_15)
- [13] Phothong, W; Wu T.C.; Lai, J. Y.; Wang D.W.; Liao C. Y.; Lee J. Y.: 3D Model Reconstruction and Re-meshing, 2016 Machining, Materials and Mechanical Technologies, Matsue Terrsa, Matsue, Japan: 7–11 October 2016. Japan: IC3MT, 2016.
- [14] Sheffer, A.; Sturler, E, de.: Parameterization of Faceted Surfaces for Meshing using Angle-Based Flattening, *Engineering With Computers*, 17(3), 2001, 326–337. <http://doi.org/10.1007/PL00013391>
- [15] Sheffer, A.; Lévy, B.; Mogilnitsky, M.; Bogomyakov, A.: ABF++: fast and robust angle based flattening, *ACM Transactions on Graphics*, 24(2), 2005, 311–330. <http://doi.org/10.1145/1061347.1061354>
- [16] Sheffer, A.; Praun, E.; Rose, K.: Mesh parameterization methods and their applications, *Foundations and Trends® in Computer Graphics and Vision*, 2(2), 2006, 105–171. <http://doi.org/10.1561/0600000011>
- [17] Yemez, Y.; Sahilioglu, Y.: Shape from silhouettte using topology-adaptive mesh deformation, *Pattern Recognition Letters*, 30(13), 2009, 1198–1207. <http://doi.org/10.1016/j.patrec.2009.05.012>
- [18] Zayer, R.; Lévy, B.; Seidel, H. P.: Linear angle based parameterization, *Fifth Eurographics Symposium on Geometry Processing-SGP*, 2007, 135–141. <http://doi.org/10.2312/SGP/SGP07/135-141>
- [19] Zigelman, G.; Kimmel, R.; Kiryati, N.: Texture mapping using surface flattening via multidimensional scaling, *Visualization and Computer Graphics*, 8(2), 2002, 198–207. <http://doi.org/10.1109/2945.998671>