Journal of Computer Sciences

Original Research Paper

Constraints Optimization for Minimizing Stretch in Bounded-Parameterization

[1]Anuwat Dechvijankit, [2]Hiroshi Nagahashi and [2]Kota Aoki

[1]*Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology, Yokohama, Japan*
[2]*Imaging Science and Engineering Laboratory, Tokyo Institute of Technology, Yokohama, Japan*

Article history
Received: 18-05-2016
Revised: 12-07-2016
Accepted: 23-11-2016

Corresponding Author:
Anuwat Dechvijankit
Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology, Yokohama, Japan
Email: dech.anuwat@gmail.com

Abstract: In a mesh parameterization process that generates one-to-one mapping information between a three-dimensional surface and a two-dimensional plane, we need to set some constraint positions in the solving system to define a specific location or size of the mapping plane. In this study, we will discuss how to perform a bounded-parameterization that minimizes distortion based on changing of constraints setting in the solving system. We introduce a series of experiments focusing on constraints optimization to deliver the optimal mapping information with less computational cost and time. Our proposed methods can reduce more than half of calculation costs and times from traditional method while maintaining the optimal mapping information.

Keywords: Mesh Parameterization, Optimization, Heuristic, Computational Geometry, Parallel Computing

Introduction

Mesh parameterization is formulated as a mapping from a 3D triangulated surface to a certain 2D planar domain. Parameterization between two domains generally causes distortion errors. Hence, low stretching is an important criterion for parameterization and its applications.

To perform a mesh parameterization, we first need an input mesh to be disk topology and then define a shape of target planar. There are two categories of planar shape, bounded and natural boundaries. However, both of them need constraint values in their solving systems. In bounded-parameterizations, all boundary vertices are constrained into a certain shape in planar domain and then solve remaining interior vertices. There are two famous constraint shapes; circle and square.

Using a circular parameterization, different constraints settings in a square parameterization can generate different qualities of mapping information as shown in Fig. 1. The easiest way to deliver the lowest distortion is to do parameterizations with all possible constraints settings then check the results. Although it can guarantee the best result, it consumes a lot of computation time and resources.

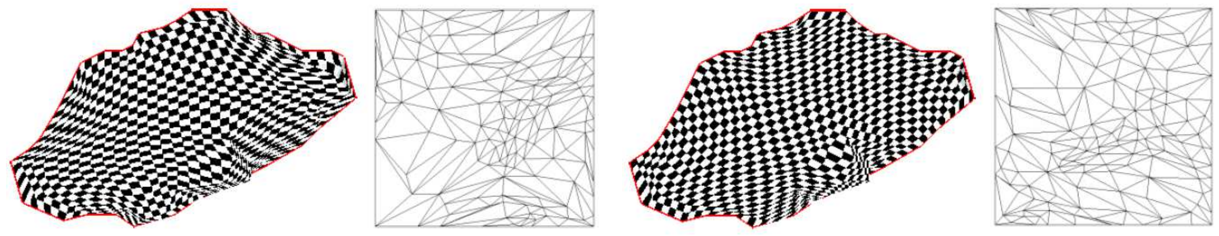
First, we will discuss how to set constraints positions in both circular and square parameterizations. Then, we present various approaches to get the lowest distortion in square parameterization by avoiding the brute-force scheme. Moreover, since multi-core CPU and GPU-computing have been introduced and widely used nowadays, the proposed approaches should support parallel computing scheme as well.

Backgrounds

In 3D computer graphics or computational geometry studies, there are many techniques and applications that involve with a mapping information between 3D and 2D domains.

Generating a mapping information between a 3D surface and a 2D data such as a square image requires a mesh parameterization process. A parameterization result can be represented by planar coordinates (u, v), indicating a position in 2D domain related to a position (x, y, z) in 3D domain.

Many well-known parameterization methods have been proposed to achieve good mapping information. Tutte (1963) used a barycentric mapping theory and created a conformal mapping. More improvement found in Floater's method (Floater, 1997), by using relative angles as a weight in each interior vertex to create barycentric mapping. Later on, stretch-minimizing methods have been proposed to achieve low stretch as possible. Yoshizawa *et al.* (2002) proposed a fast method of stretch-minimizing by re-computing the weight of linear energy-minimizing equations by using previous stretch value as a divisor.



(a) (b)

Fig. 1. Ustica models with checker-board texture mapping using the same square parameterization method with different constraints settings, (a) shows the worst case that has the largest L^2 stretch (1.320295), (b) shows the best case that has the smallest L^2 stretch (1.175196)

Concerning the limitation of bounded-parameterization, Least Squares Conformal Maps (LSCM) by Levy *et al.* (2002) were presented as alternative ways to optimize the boundary positions from fixed-boundary into free-boundary. They used different harmonic energy formulations found in harmonic map (Eck *et al.*, 1995) but still minimized angular distortion. Intrinsic parameterizations (Desbrun *et al.*, 2002) used the same technique found in LSCM to preserve angle distortion and preserved area distortion. Both of them could significantly improve the distortions. However, they aimed to optimize by changing bounded-boundary into natural-boundary parameterization.

Sorkine *et al.* (2002) proposed a bounded-distortion concept with simultaneous seam-cutting and they generated a valid parameterization without local or global fold overs and also controlled each mesh triangle distortion not to exceed a certain threshold. Lipman (2012) also proposed bounded-distortion mapping spaces which can control worst-case conformal distortion, orientation preserving and one-one mapping in various existing mapping algorithms. However, they aimed to control the mappings at unconstrained parts.

On the topic of error metric, Sander *et al.* (2001) presented a method to minimize texture stretch to balance sampling rates over all locations and directions on the surface, called "progressive mesh". To measure the local stretch of mapping in every direction, they defined a new "texture stretch" metric on triangle meshes, known as L^2 stretch metric.

Notation

Before explaining various algorithms, let us define basic notations. We represent a disk topology mesh $\mathcal{M}:= (V,E,F)$, where $V:= \{v_i \in \mathbb{R}^3 | i = 0,\ldots,n_v-1\}$ is a set of n_v vertices. $E:= \{e_i(v_a,v_b) | i = 0,\ldots,n_e-1: v_a, v_b \in V: a \neq b\}$ is a set of ne edges and $F:= \{f_i(e_r,e_s,e_t) | i = 0,\ldots,n_f-1: e_r,e_s,e_t \in E: r \neq s \neq t\}$ is a set of n_f faces.

Let $B:= \{b_i \in V | i = 0,\ldots,n_b-1\}$ and $E_B = \{e(b_0,b_1), e(b_1,b_2),\ldots,e(b_{nb}-1, b_0)\}$ be a set of n_b boundary vertices and boundary edges respectively. Let d equal to the total length of E_B and let a sequence of all boundary vertices be $V_B = (b_0,b_1,\ldots,b_{nb-1})$ sorted in the order of connection of E_B.

Bounded Parameterization

In surface parameterization, there are two major shape categories, i.e., bounded and natural boundaries. In natural boundaries, we need to constrain two vertices in planar domain to fix the rotation and translation of the result (Desbrun *et al.*, 2002), then perform the parameterization to solve unconstrained vertices' coordinates.

In bounded-parameterizations, we constrain all boundary vertices into a certain shape in planar domain then solve remaining interior vertices. There are two famous shapes in bounded-parameterization; circle and square. However, there are no specific algorithms for defining constraint positions. Users can create their own algorithm based on the length or numbers of boundary edges and so on.

We will discuss our constraints setting algorithms in both circular and square shapes from now.

Circular Parameterization

There are two appropriate schemes to constrain boundary positions in circular shape. The first one is averaging circle arc's length by the number of boundary edges and the second one is variation by the length of each edge. Although averaging approach is a simple method, it can have high distortion around boundary areas because the constraints of boundary vertices are not balanced by their edge's lengths. A long edge can be assigned with too short constraint positions and a short edge can be assigned with too long constraint positions. Therefore, we use variation by the length approach for circular constraints.

Suppose that circular plane has radius r and has initial center at planar origin (0,0). First, we constrain $b_0 \in B$ at $(r,0)$ then map the boundary edges E_b on the circular arc in counter-clockwise direction (Fig. 2a). We assign constraint positions of b_i in planar as C_i (S, t) by the following formula:

$$C_i\,(s,t) = \left(r \times \cos\left(2\pi\delta_i\right), r \times \sin\left(2\pi\delta_i\right)\right) \delta_i = \sum_{k=0}^{i-1} \left| \left(b_k, b_{k+1}\right) / d \right|$$

Since we can rotate circular mapping plane to any degree, it has the same meaning as constraining b_0 at other positions in circle arc. Therefore, there is no constraints optimization in circular parameterization.

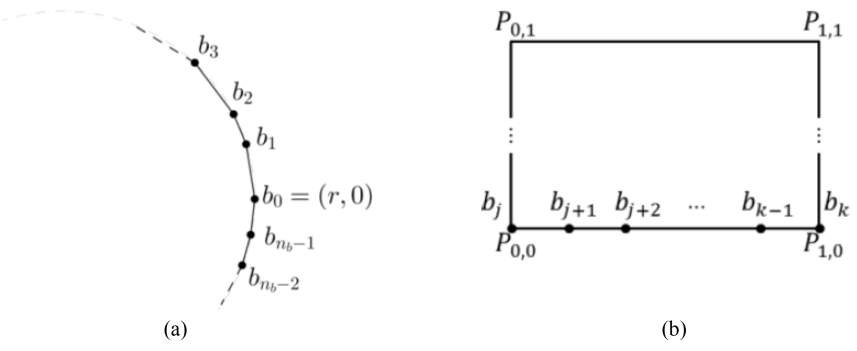


Fig. 2. Constraints mapping sequences on planar points, (a) circular constraints, (b) square constraints

Square Parameterization

Constraining boundary vertices in a square shape is similar to circular shape's approach, i.e., averaging by the number of boundary edges and variation by the length of each edge. With the same reason with circular constraints that we try to deliver the best result, variation by the length approach is chosen.

We assign $b_0 \in B$ as a reference starting point of V_b and E_b. Let $P_{i,j}$ be a corner position in square planar domain as shown in Fig. 2b and μ be the length of each side of square plane. We try to map some edges in E_b onto a side of the plane. In order to achieve low stretch at boundary area as much as possible, one side of square should be mapped on a quarter length of E_b that is, $0.25d$.

Let b_j be a vertex in V_b that we want to constrain onto $P_{0,0}$. Let the total length of edges starting from vertex b_j to vertex b_k be $l = \sum_{i-j}^{k-1} \left| \left(b_i, b_{i+1} \right) \right|$ where $k = (j+m) \bmod n_b$. We try to find the ending vertex bk that satisfy $l = 0.25d$ as a quarter length of E_b. However, in most cases it cannot be equal exactly. Therefore, we find the ending vertex that satisfy $l \approx 0.25dd$. We map the boundary edges $(b_j, b_j + 1, \cdots, b_k)$ onto the square side from $P_{0,0}$ to $P_{1,0}$ relatively on each edge length over μ.

For other sides of the square, we can follow the same basis described above by considering b_k as the starting point from $P_{1,0}$ corner and iterate the constraints setting process to $P_{1,1}$, $P_{0,1}$ and finally back to $P_{0,0}$.

When considering about constraints optimization in square case, different constraints different b_i onto $P_{0,0}$ can give different result. However, we cannot rotate square plane to any degree like circle's case. The optimization problem is how to determine constraints setting that gives the lowest distortion parameterization result.

Square Constraints' Optimization

Considering the problem of square constraints optimization, we map the boundary vertices in the mesh domain onto the square's boundary in the planar domain with some conditions. The method is trying to map a quarter length of the total boundary edges onto one side of the square as mentioned in the previous section. With this condition, a new complexity arises. That is, a constraints setting can have a different number of edges on each side of the square. It is impossible to incorporate these boundary conditions into a linear solving system.

The simplest way to perform is a brute-force approach using a stretch-minimizing parameterization. When considering constraints setting, brute-force approach means every vertex in V_b is mapped onto $P_{0,0}$ then performs a stretch-minimizing parameterization. Although it guarantees the best result, one-time stretch-minimizing parameterization on a fine details mesh can use a lot of computation time and resources. Applying parallel-processing may help on time issue, however it only suits for an environment that has many computation resources.

To find the optimal boundary constraints, we did experiments observing the stretch of the unit square boundary by various fast solving parameterization methods i.e., shape-preserving (Floater, 1997), Tutte's barycentric (Tutte, 1963), mean-value (Floater, 2003) and harmonic-map (Eck *et al.*, 1995). L^2 stretch (Sander *et al.*, 2001) values were compared between the fast solving methods and the stretch-minimizing method by Yoshizawa *et al.* (2002) approach. There was not any relationships among them. Constraints setting that give the lowest stretch in fast solving methods do not guarantee the lowest stretch from a stretch-minimizing method. We should use stretch-minimizing parameterization directly for optimization.

Experimental Environment

We did experiments with two systems with different computing performances. There were a high-performance system (Intel Xeon™ 10 cores running at 2.50-3.30 GHz with 64 GB memory) and a moderate-performance system (Intel Core i7™ 4 cores running at 2.40-3.40 GHz with 8 GB memory).

Experiment Models Information

See Table 4 for details.

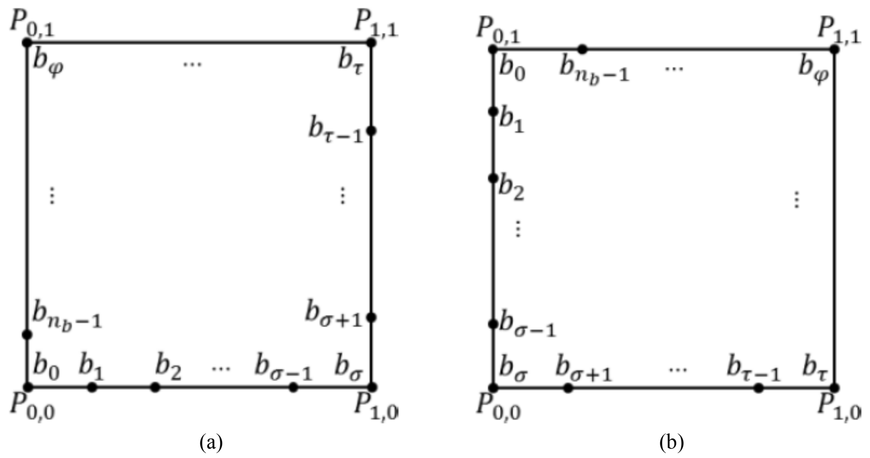


Fig. 3. A similarity of square constraints after shifting constraint at $P_{0,0}$ for a quarter of the total length of boundary edges, (a) b_0 onto $P_{0,0}$, (b) b_0 onto $P_{0,1}$

25 Percent of Brute-Force

Let the first boundary constraints be b_0 onto $P_{0,0}$, b_σ onto $P_{1,0}$ and b_τ onto $P_{1,1}$ (b_0, b_σ, $b_\tau \in V_b$) that are assigned by the boundary constraints algorithm. It means the distance from b_0 to b_σ should be approximately $0.25d$, also the same for distance from b_σ to b_τ.

By starting from b_0, we sequentially constrain a vertex b_i onto $P_{0,0}$ and constrain the following boundary vertices onto the square domain [$P_{0,0}$, $P_{1,0}$]. After repeating the constraints for interval of a quarter of boundary edges, then the vertices b_σ and b_τ might be mapped onto $P_{0,0}$ and $P_{1,0}$ respectively. It is the same as we rotate the first constraints (b_0 onto $P_{0,0}$) 90 degree as shown in Fig. 3.

Even though, we can reduce the number of calculations in brute-force approach to be around 25%. It still takes a lot of calculation time. A single stretch-minimizing parameterization on a fine details mesh could take calculation time more than one minute on a present day high performance computer. That means multiple parameterizations using 25% brute-force approach could take time more than hours or half a day to complete.

From this reason, we set our goal to reducing the number of calculations. If we can pursue an optimization by doing parameterizations with few testing cases, then it will surely be better than 25% brute-force method.

Heuristics

When dealing with a computational optimization problem, it is good to check with other well-known optimization techniques. There are many techniques and generally they are divided into 3 categories. Optimization algorithms such as a Simplex algorithm are suited for linear or quadratic programming solving. Iterative methods such as Newton and Quasi-Newton methods suit for non-linear programming solving. Heuristic algorithms such as genetic algorithms or hill climbing suit for solving the problems that cannot be solved or too slow by classic methods.

Considering the mentioned problem, it concluded that a heuristic algorithm may be the best one for boundary optimization problem. The reason is that the boundary optimization problem has the difficulty of two sub-problems connected together. One sub-problem is concerned with our main problem, i.e., finding best constraints of boundary vertices and edges. Another is a parameterization problem about solving planar locations of interior vertices. It is too difficult to combine the two sub-problems in a problem-solving system since it has the unique conditions for the boundary constraints. Hence, a heuristic method should suit for solving our constraints optimization problem.

We chose a well known "Particle Swarm Optimization" algorithm (PSO) by (Kennedy and Eberhart, 1995) for solving our optimization problem. It is a swarm intelligent technique, originally inspired by social behavior of animal flocking. PSO has been used mainly to solve unconstrained, single-objective optimization problems.

From experiment results, PSO method could reduce the calculation time to around 50 to 75% comparing to "25% brute-force" approach. By changing user-defined parameters, we could improve calculation time in some mesh models. However, we could not gain an expected results from the turning of these user-defined parameters yet. The number of particles plays important roles; more particles could secure the best result (same as brute-force's result) but calculation cost increases. The algorithm itself is based on random processes and it may give the unstable optimal answer if we use a small number of particles. Increasing the number of particles will cost almost the same calculation time as "25% brute-force" approach. Moreover, PSO has a disadvantage point on parallel computing because it updates particle positions based on the global-best position in each iteration which prevents from doing large numbers of parallel-processing simultaneously.

It could conclude that PSO or other heuristic optimizations can improve computation time when using an appropriate number of particles. Even, the performance was still not good as our expectation but its algorithm of checking few positions and focusing around potential optimal answers seem to be appropriate to the optimization problem. The main bottleneck of PSO is a procedure of random search. Avoiding the random search while checking few positions should generate better performance stably.

Step-Sampling

When analyzing square parameterization by looking at brute-force results, the most of the test mesh models' L^2 stretches are changed gradually when we changed b_i at $P_{0,0}$ along V_b (Fig. 5). From this characteristic, we can reduce the number of calculations by focusing on the boundary constraints settings that has high potential to give an optimal result. In order to do such thing, we need to know its appropriate searching scope as potential optimal constraints.

The problem is how to determine the searching scope. A sampling approach is used as a survey of stretch values. Sampling is the reduction of a signal. A common example is the conversion of a sound wave (a continuous signal) to a sequence of samples (a discrete-time signal).

Step-Sampling Algorithm

We propose a simple algorithm named "step-sampling". It samples stretch values from selected boundary constraints. Boundary constraints will be selected by a step defined by a user and performed a stretch-minimizing parameterization. After the sampling is completed, we will get a constraints setting that gives the lowest stretch as the center of the searching area. Then, we do deep-checking in that area. We check its neighbors that are still unchecked for stretch values. The parameterization that has the lowest stretch is the optimal result.

The following Algorithm 1 is the pseudo-code of the step-sampling algorithm.

Algorithm 1. Pseudo-code of step-sampling algorithm

```
Step = 2,3,…
m = number of vertices in first 0.25d of $E_b$
stretch [ ] ← ∞                    //m size array
for (i ← 0 to m−1)
    if (i mod step = 1)
        stretch [i] ← square-param($b_i$ on $P_{0,0}$)
    end if
end for
ω ← index that gives MIN(stretch [ ])
\\Indicate initial optimum and searching scope
J[ ] = [...,ω-2, ω-1, ω+1, ω+2,...] where stretch [ ] = ∞ and close to ω
for (j ← each J) //deep checking
    stretch [j] ← square-param ($b_j$ on $P_{0,0}$)
end for
optimum ← MIN(stretch [ ])
```

Step Value

In the step-sampling algorithm, a proper step value can minimize the calculation time. However, a too large step may result in missing correct searching scope or spending more calculation time than a smaller step because larger step means larger deep-checking processes. Let m be the total test cases (boundary constraints) that can be calculated from the number of vertices in the first $0.25d$ interval. From empirical experiments, the formula for a proper step value is:

$$step \approx \sqrt{n_v \times m / n_f}$$

This formulated step value is suitable for high details models. For low details models, if the parameterizations are performed on a high-performance system then it might be better to use brute-force method or small step value that can guarantee the best result while the computation time is almost the same with high step numbers' results.

Experiment Result

We tested 25% brute-force and our step-sampling algorithms on two systems to see how the performances of proposed methods are on various computation resource conditions. We tested 25% brute-force and our step-sampling algorithms on two systems to see how the performances of proposed methods are on various computation resource conditions as mentioned above.

Both algorithms were optimized to use parallel processing (OpenMP®) as much as possible. However, the number of maximum threads was limited to the number of physical cores. All parameterizations were performed by using the algorithm from Yoshizawa *et al.* (2002) method. For the testing models, if the original models are not disk topology, we need to convert them into disk topology patches before performing the parameterization. We used several methods found in papers by (Gu *et al.*, 2002; Dechvijankit *et al.*, 2012; 2015) for converting a mesh into topological disk patch.

Step values were calculated by proposed formula for all models even coarse-details ones. We recorded the number of how many times stretch-minimizing parameterizations were performed (test counts) and L^2 stretch errors of the optimal results from 25% brute-force and step-sampling approaches. The results are shown in Table 1. Also, computation times on the both systems were measured. The results are shown in Table 3.

Table 1. Brute-force and step-sampling results

	25% brute-force		Step-sampling		
Model	Test count	L^2 stretch	Step value	Test count	L^2 stretch
Torus	16	1.337377	3	11	1.338278
Torus 3 holes	23	1.643867	3	13	1.643867
Hand	17	1.449979	3	10	1.459317
Head	20	1.235226	3	12	1.235226
Eight	45	1.466127	5	18	1.466127
Chains	153	1.452289	8	35	1.452289
Cow	440	14995.2	14	59	14995.2
Bunny 5 holes	166	1.370189	9	36	1.370189
Bunny no hole	108	1.226405	7	29	1.226405
Armadillo	655	1.416241	18	72	1.416241

As the experiment results, our step-sampling approach spent less time and computations comparing to brute- force one. Time different margins were dependent on models' complexity, the step-sampling method could reduce the computational time much more in higher-detail meshes. Also it could deliver optimal results same as brute-force. There were some cases (torus and hand) which step sampling could not deliver the same optimal result compared with brute-force one. However, they were low-detail meshes cases that should use brute-force approach or smallest step value 2 rather than the formulated one.

Circular Mapping Analysis

While different square constraints settings give different results, circular constraints settings have a singular result. Therefore, we analyze the circular parameterization (circular mapping) at its boundary vertices' stretch. We investigated at the boundary vertex that has that the highest stretch in circular parameterization and the best square parameterization's constraints mapping. We noticed that the highest stretch boundary vertex (in the circle) is likely near to a corner in the optimal square parameterization. Therefore, we can use this information to perform square constraints optimization.

There are two approaches for the optimization using the circular stretch. The first one is to constrain the largest stretch boundary vertex in the circle to a corner in the square. Another one is to constrain high stretch boundary vertices to around all four corners of the square as much as possible.

Highest Stretch Boundary Vertex at Corner

This optimization "single highest" can be done by a simple process. First, we perform circular parameterization. Then, we analyze L^2 stretch errors and find the highest stretch vertex. Finally, we analyze all possible square boundary constraints mappings (without doing parameterizations). We assume that the constraints mapping which the largest stretch vertex in the circle be constrained to a corner in the square may be an optimum.

High Stretch Boundary Vertices around Corners

This optimization "many high" is more complex than single highest approach. We need to define a cost function of a square constraints setting. The cost should indicate how high stretch vertices in circular parameterization are mapped in square constraints. If many high stretch vertices were assigned around corners' vicinity, then the cost function should give a high cost value.

In the cost function, the distance between circle's arc and square's side-line is used as the weight in the cost function. Here, we assume that circle's radius $r = 1$ and square side length $\mu = 2$. A vertex be mapped on a corner should have the maximum weight and a vertex be mapped on a middle of a side-line should have the minimum weight. Let u_i be the distance between assigned position of b_i and the nearest corner point (normalized relatively to μ), $stretch_v$ (b_i) be L^2 stretch value of a boundary vertex in circular parameterization also let w_i be the weight of analyzing vertex as the distance from the point on square to circle's arc with 45 degrees angle. Fig. 4 shows the meanings of variables. The cost function of each constraints mapping is formulated as:

$$\cos t = \sum_{i=0}^{b_{n-1}} \left(stretch_v(b_i) \times w_i \right) \; w_i = \frac{2 - u_i - \sqrt{2 - u_i^2}}{\sqrt{2}}$$

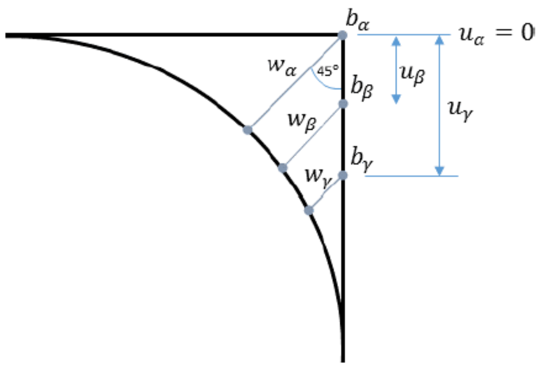


Fig. 4. "High stretch boundary vertices around corners" cost function's variables meaning based on the distance between circle's arc and square's side-line

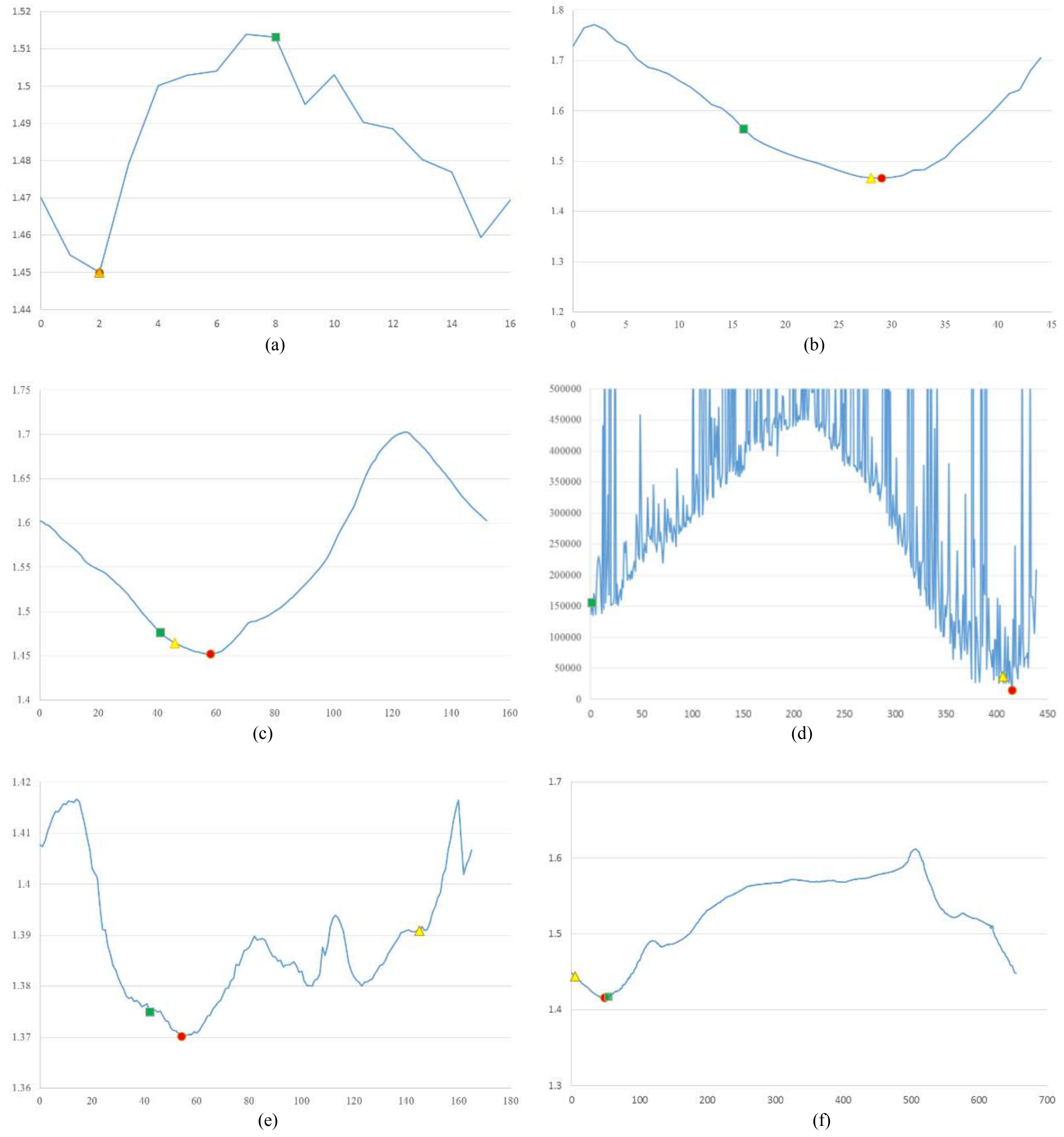


Fig. 5. L^2 stretch values graphs of 25% brute-force approaches. Red circle markers indicate the lowest stretch constraints. Yellow triangle markers indicate constraints defined by "single highest" approach. Green square markers indicate constraints defined by "many high" approach. The x-axes represent constraints mapping ID, the y-axes represent L^2 stretch errors. (a) hand, (b) eight, (c) chains, (d) cow, (e) bunny 5 holes, (f) armadillo

Experiment Result

We tested both approaches with same experimental conditions that be described before. In each testing model, L^2 stretch values from 25% brute-force approach were recorded. Then, stretch-minimizing circular parameterization was performed on it. We analyzed L^2 stretch values and classified which constraints mappings satisfied the conditions on both approaches. The results from some experiments are shown in Fig. 5.

As the results, the prediction using circular parameterization approach could indicate a square boundary constraints mapping which is close to the best result. Either "single highest" constraints mapping or "many high" constraints mapping is close to the 25% brute-force one. However, it still cannot guarantee the same result yet.

Select a Better One

From experiment results, either "single highest" constraints setting or "many high" constraints setting gives a potential to deliver an optimal result. Therefore, these two approaches could be used for square constraints optimization.

By comparing the stretches of both approaches, the one that gives a better result is chosen to be the optimal result. The benefit of this overall approach is to reduce a lot of the computation to one circular parameterization and two square parameterizations. However, it does not give the best result same as 25% brute-force one in most cases but approximately optimal one.

Step-Sampling with Circular Mapping Analysis

In previous sections, two main approaches for square constraints optimization were proposed; step-sampling optimization and prediction-based optimization from circular mapping. Both have advantages and disadvantages.

The first one uses a sampling technique to define an optimal area and then performs deep-checking of the area. The performance is good when running with parallel processing, however it still has high computation cost on high-details models. It is not suitable for a system that has limited computation resource environment.

The second one uses L^2 stretch values of each vertex in circular parameterization and then performs two square parameterizations to get a better result. Although it reduces a lot of computation, it mostly does not deliver the best result same as brute-force or step-sampling yet.

Hybrid Approach

To reduce the computation cost and get the optimal result same as 25% brute-force approach, we can have a constraints optimization method by combining two previous approaches together. We use the advantage of prediction-based optimization using circular mapping to cut the computation cost of finding an optimal area in step-sampling.

First, an initial optimal constraint is defined by the circular mapping analysis approach. Then, the optimal area is defined by searching around the initial optimal constraints. However, the optimal area might not contain the initial optimal constraints.

To define the actual optimal area, we adapted step-sampling technique by sampling the neighbors of the initial optimal constraints. We select to check specific neighbors' constraints by a constant interval. We terminate the sampling if their stretches are increasing. The constant interval is important, for properly and convergently constraints investigation. Too small interval might cause mistakenly termination of sampling process too early before detecting the actual optimal area. Too big interval might cause too large searching scope.

From empirical experiments, a formula to get a proper constant interval is based on the step number formula (see step-value section). The formula for the interval is:

$$interval \approx step - 1 \approx \sqrt{n_v \times m / n_f} - 1$$

After the end of sampling processes, we detect the lowest stretch constraints mapping and define it as a center of the optimal area. As well as the step-sampling basis, we perform deep-checking constraints mappings around the optimal area to get the final optimal result.

The following Algorithm 2 is a pseudo-code of the hybrid algorithm. Also, Fig. 7 shows the flowchart of our hybrid algorithm.

Algorithm 2. Pseudo-code of the hybrid algorithm

```
    $m$ = number of vertices in first $0.25d$ of $E_b$
    $stretch$ [ ] ← ∞ //m size array
    $interval \leftarrow \sqrt{n_v \times m / n_f} - 1$
    $stretch$ [$\gamma$] ← optimum for
                    prediction-from-circular-param( )
    //$\gamma$ is index number that represent as
    // square-param($b_\gamma$ on $P_{0,0}$)
    $\gamma_l, \gamma_r \leftarrow \gamma$
    do
        $\bar{\gamma}_l \leftarrow \gamma_l$
        $\gamma_l \leftarrow \gamma_l$ - $interval$
        $stretch$ [$\gamma_l$] ← square-param($b_{\gamma_l}$ on $P_{0,0}$)
    until $(stretch[\gamma_l] < stretch[\bar{\gamma}_l])$
    do
        $\bar{\gamma}_r \leftarrow \gamma_r$
        $\gamma_r \leftarrow \gamma_r$ - $interval$
        $stretch$ [$\gamma_r$] ← square-param ($b_{\gamma_r}$ on $P_{0,0}$)
    until ($stretch$ [$\gamma_r$]< $stretch$ [$\bar{\gamma}_r$])
    ω ← index that give MIN($stretch$ [ ])
    $J$[ ] = [...,ω-2, ω-1, ω+1,ω+2,...] where $stretch$ [ ] =
    ∞ and close to ω
for ($j$ ← each $J$)                    //deep checking
        $stretch$ [$j$] ← square-param($b_j$ on $P_{0,0}$)
    end for
    $optimim$ ← MIN($stretch$ [ ])
```

Experiment Results

We tested our hybrid approach with the same experimental conditions that be described before. The results are shown in Table 2 and 3. We used formulated constant interval values that used for sampling the neighbors of the initial optimal constraints to find actual optimal area phase.

Table 2. Hybrid approach results. Test numbers in bracket indicate step-sampling test counts

Model	*Interval*	Test count	L^2 stretch
Torus	2	5 (11)	1.337377
Torus 3 holes	2	6 (13)	1.643867
Hand	2	6 (10)	1.449979
Head	2	5 (12)	1.235226
Eight	4	10 (18)	1.466127
Chains	7	18 (35)	1.452289
Cow	13	28 (59)	14995.2
Bunny 5 holes	8	20 (36)	1.370189
Bunny no hole	6	14 (29)	1.226405
Armadillo	17	36 (72)	1.416241

Table 3. Time consuming results on two systems. All time units are second

	High performance system (Xeon): Time			Moderate performance system (i7): Time		
Model	25% brute-force	Step-sampling	Hybrid	25% brute-force	Step-sampling	Hybrid
Torus	0.016	0.009	0.016	0.016	0.016	0.016
Torus 3 holes	0.032	0.015	0.016	0.031	0.031	0.016
Hand	0.062	0.060	0.110	0.110	0.079	0.109
Head	0.047	0.031	0.078	0.078	0.062	0.047
Eight	0.078	0.047	0.063	0.125	0.062	0.063
Chains	4.376	1.125	1.891	14.126	2.969	2.328
Cow	619.438	83.710	50.597	2723.050	357.925	175.010
Bunny 5 holes	130.729	28.393	27.439	573.890	125.255	71.347
Bunny no hole	90.976	25.799	20.251	395.943	105.850	52.847
Armadillo	11298.100	1384.340	904.433	41632.200	5102.450	2903.420

Table 4. Experiment models information

Model	Vertices	Faces	Boundary edges
Torus	233	404	59
Torus 3 holes	246	393	86
Hand	1082	2002	159
Head	713	1357	66
Eight	863	1548	175
Chains	6885	13164	603
Cow	38658	75618	1695
Bunny 5 holes	35073	69465	662
Bunny no hole	35068	69662	471
Armadillo	174296	345952	2637

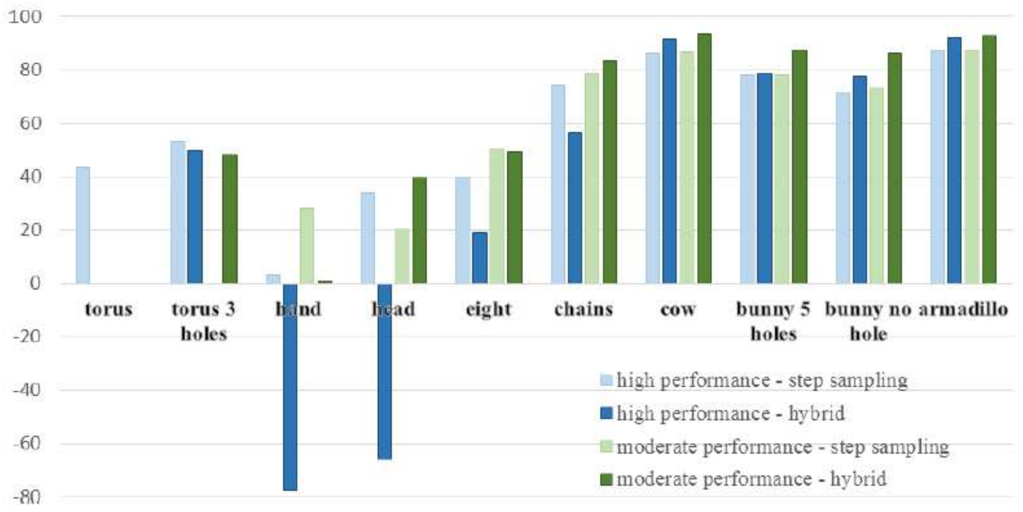


Fig. 6. Percentage of time-consuming reduction on both step-sampling and hybrid approaches compare with 25% brute-force time. The y-axes represent percentage unit, a higher value means more time-consuming reduction (higher is better)

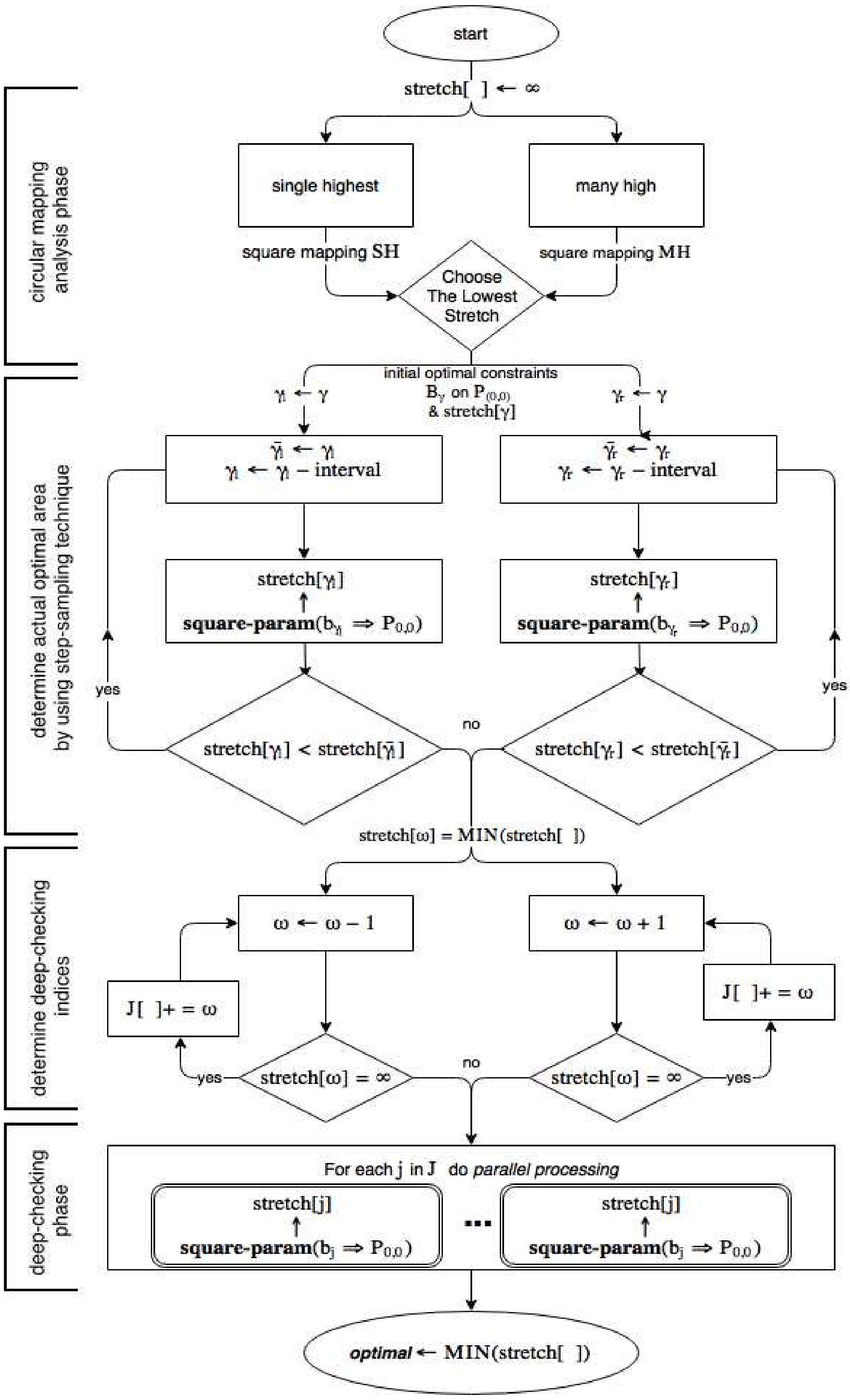

start
stretch[ ] ← ∞
circular mapping analysis phase
single highest
many high
square mapping SH
square mapping MH
Choose The Lowest Stretch
initial optimal constraints
B_γ on $P_{(0,0)}$
& stretch[γ]
$\gamma_l \leftarrow \gamma$
$\gamma_r \leftarrow \gamma$
determine actual optimal area by using step-sampling technique
$\bar{\gamma}_l \leftarrow \gamma_l$
$\gamma_l \leftarrow \gamma_l$ − interval
$\bar{\gamma}_r \leftarrow \gamma_r$
$\gamma_r \leftarrow \gamma_r$ − interval
stretch[γ_l] ↑ square-param($b_{\gamma_l} \Rightarrow P_{0,0}$)
stretch[γ_r] ↑ square-param($b_{\gamma_r} \Rightarrow P_{0,0}$)
yes
stretch[γ_l] < stretch[$\bar{\gamma}_l$]
no
stretch[γ_r] < stretch[$\bar{\gamma}_r$]
yes
stretch[ω] = MIN(stretch[ ])
determine deep-checking indices
ω ← ω − 1
ω ← ω + 1
J[ ]+ = ω
J[ ]+ = ω
yes
stretch[ω] = ∞
no
stretch[ω] = ∞
yes
deep-checking phase
For each j in J do parallel processing
stretch[j] ↑ square-param($b_j \Rightarrow P_{0,0}$)
...
stretch[j] ↑ square-param($b_j \Rightarrow P_{0,0}$)
optimal ← MIN(stretch[ ])

Fig 7. Flowchart of our hybrid algorithm

From the results, our hybrid approach could deliver the best result same as 25% brute-force one. The total amount of square parameterizations was reduced from step-sampling around than half. About the time-consuming, our hybrid algorithm spent time more than step-sampling on low-details models because the hybrid approach must wait until the circular parameterization was finished before performing square parameterizations. On the high-details models such as bunny and armadillo, the hybrid approach could deliver much faster results, especially on a moderate-performance system. Fig. 6 shows the comparison of both approaches comparing to 25% brute-force.

Conclusion

We presented various approaches for constraints optimization in bounded-parameterization especially square one to deliver the lowest stretch error as possible. The easiest way to delivering the lowest stretch square parameterization is to do brute-force approach with a stretch-minimizing method. However, it will consume a lot of computation time and resources.

We proposed our "step-sampling" concept which reduces much calculation time while maintaining a stable optimal result. Although it is a simple algorithm, we can have great performance that reduces computations more than half from brute-force approach. We also proposed a formula to calculate suitable step number based on mesh's complexity that will minimize the calculation time. However, the computational cost is still high and might not suit for running in an environment that has limited resources.

Then, we analyzed stretches in circular parameterization for an optimal square parameterization. We found that constraining the highest stretch boundary vertex in circular mapping onto a square corner or high stretch boundary vertices in circular mapping onto around square corners' vicinity, could give an optimal constraints mapping. Although it could reduce calculation costs to only one circular and two square parameterizations, it still cannot guarantee the best constraints mapping same as brute-force result yet.

Finally, we proposed a hybrid approach by combining step-sampling and circular parameterization analysis approaches. By narrowing the searching scope of the constraints by analyzing circular mapping's stretches, we could get the best result as brute-force, while reducing a lot of computation cost. Our hybrid approach is suitable for applying in a limited resources environment.

For open topics and future works, there are some parts that can be improved more such as how to reduce parameterization time using fast-solving with step-sampling or circular mapping analysis approaches. Also, there are other parameterizations that involve optimization interior vertices such as conformal optimizations in brain surfaces (Gu *et al*., 2003; Lui *et al*., 2010). We can combine our exterior vertices (constraints) optimization with an interior optimization to deliver the optimal mapping in both ways.

Acknowledgement

Models are courtesy of the AIM@SHAPE repository and Stanford University. Special thanks are given to Shin Yoshizawa for parameterization code, to Hugues Hoppe for filled holes bunny and hand models data and to the anonymous reviewers for comments and suggestions.

We acknowledge the support of JSPS KAKENHI (Grant Number 24300035).

Author's Contributions

Anuwat Dechvijankit: Designed, implemented and performed research, wrote manuscript and acted as corresponding author.

Hiroshi Nagahashi: supervised development of research, manuscript evaluation and correction.

Kota Aoki: Supervised development of research.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Dechvijankit, A., H. Nagahashi and K. Aoki, 2012. Fast way to create seam boundary for square parameterization with low-distortion. Proceedings of the International Conference on Computer Graphics Theory and Applications and International Conference on Information Visualization Theory and Applications, Feb. 24-26, Scitepress, Italy, pp: 185-188. DOI: 10.5220/0003811701850188

Dechvijankit, A., H. Nagahashi and K. Aoki, 2015. A homotopy surface cutting using paths crossing in geodesic distance. Proceedings of the 10th International Conference on Computer Graphics Theory and Applications, Mar. 11-14, Scitepress, Germany, pp: 130-137. DOI: 10.5220/0005302601300137

Desbrun, M., M. Meyer and P. Alliez, 2002. Intrinsic parameterizations of surface meshes. Comput. Graph. Forum, 21: 209-218. DOI: 10.1111/1467-8659.00580

Eck, M., T. DeRose, T. Duchamp, H. Hoppe and M. Lounsbery *et al*., 1995. Multiresolution analysis of arbitrary meshes. Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, Aug. 06-11, Los Angeles, CA, USA, pp: 173-182. DOI: 10.1145/218380.218440

Floater, M.S., 1997. Parametrization and smooth approximation of surface triangulations. Comput. Aided Geomet. Des., 14: 231-250. DOI: 10.1016/S0167-8396(96)00031-3

Floater, M.S., 2003. Mean value coordinates. Comput. Aided Geomet. Des., 20: 19-27. DOI: 10.1016/S0167-8396(03)00002-5

Gu, X., S.J. Gortler and H. Hoppe, 2002. Geometry images. ACM Trans. Graph., 21: 355-361. DOI: 10.1145/566654.566589

Gu, X., Y. Wang, T.F. Chan, P.M. Thompson and S.T. Yau, 2003. Genus zero surface conformal mapping and its application to brain surface mapping. Proceedings of the 18th International Conference on Information Processing in Medical Imaging, Jul. 20-25, Ambleside, UK, pp: 172-184. DOI: 10.1007/978-3-540-45087-0_15

Kennedy, J. and R. Eberhart, 1995. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, Nov. 27-Dec. 1, IEEE Xplore Press, Australia, pp: 1942-1948. DOI: 10.1109/ICNN.1995.488968

Levy, B., S. Petitjean, N. Ray and J. Maillot, 2002. Least squares conformal maps for automatic texture atlas generation. ACM Trans. Graph., 21: 362-371. DOI: 10.1145/566654.566590

Lipman, Y., 2012. Bounded distortion mapping spaces for triangular meshes. ACM Trans. Graph., 31: 1-13. DOI: 10.1145/2185520.2185604

Lui, L.M., S. Thiruvenkadam, Y. Wang, P.M. Thompson and T.F. Chan, 2010. Optimized conformal surface registration with shape-based landmark matching. SIAM J. Imag. Sci., 3: 52-78. DOI: 10.1137/080738386

Sander, P.V., J. Snyder, S.J. Gortler and H. Hoppe, 2001. Texture mapping progressive meshes. Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, Aug. 12-17, Los Angeles, CA, USA, pp: 409-416. DOI: 10.1145/383259.383307

Sorkine, O., D. Cohen-Or, R. Goldenthal and D. Lischinski, 2002. Bounded-distortion piecewise mesh parameterization. Proceedings of the Conference on Visualization, Oct. 27-Nov. 1, IEEE Xplore Press, USA, pp: 355-362. DOI: 10.1109/VISUAL.2002.1183795

Tutte, W.T., 1963. How to draw a graph. Proc. London Math. Society, s3-13: 743-767. DOI: 10.1112/plms/s3-13.1.743

Yoshizawa, S., A. Belyaev and H.P. Seidel, 2002. A fast and simple stretch-minimizing mesh parameterization. Proceedings of the Shape Modeling International, Jun. 7-9, IEEE Xplore Press, USA, pp: 200-208. DOI: 10.1109/SMI.2004.1314507