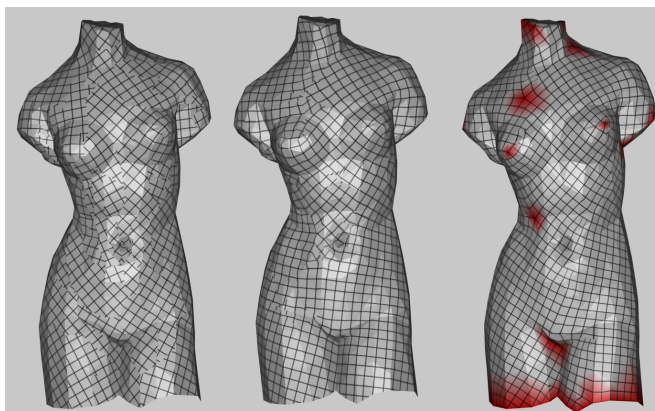


# Interactive and Smooth Parametrization-Based Quadrangulation

Roy Velich  
University of Haifa  
rvelich@campus.haifa.ac.il

Roi Poranne  
University of Haifa  
roiporanne@cs.haifa.ac.il



**Figure 1: Left: Initial mapping of cut mesh. Middle: With increased penalty on seamlessness violation. Right: With increased penalty on integer violation.**

## ACM Reference Format:

Roy Velich and Roi Poranne. 2020. Interactive and Smooth Parametrization-Based Quadrangulation. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Posters (SIGGRAPH '20 Posters)*, August 17, 2020. ACM, New York, NY, USA, 2 pages. <https://doi.org/10.1145/3388770.3407451>

## 1 INTRODUCTION

Many applications in computer graphics, such as character modeling and animation, architectural geometry, and also physical simulation to some extent, call for quad meshes as a representation of the geometry [Bommes et al. 2013b]. However, since triangle meshes are generally more prevalent, they need to be converted via the process known as *quad remeshing*.

Numerous techniques were introduced in recent years, and the most common approach is to split the problem into cross field generation, and field guided parameterization. In this document, we present an interactive, direct parameterization approach, which nullifies the need for intermediate steps.

The idea behind parameterization based methods in general, is to map the mesh to the plane, and create a regular grid layout on it. In order to ensure that the grid on the plane is transformed

into a valid quad mesh on the surface, certain conditions on the parameterization must be fulfilled [Bommes et al. 2013a]:

- (1) **Seamless Condition:** The transition function  $g_{ij}$  between two half-edges  $e_i$  and  $e_j$  on the parameterization domain that corresponds to the same surface edge that is part of a cut seam, has to be an integer-grid automorphism given by:

$$e_j = R_{90^\circ}^{r_{ij}} e_i + \vec{t}_{ij}$$

Where  $r_{ij} \in \{0, 1, 2, 3\}$  and  $\vec{t}_{ij} \in \mathbb{Z}^2$ .

- (2) **Singularity Points Condition:** All singular vertices, which are characterized by a non-zero angular defect on the parameterization domain, have to lie on integer locations. That is, given the set  $S_i$  of all parameterization domain vertices that correspond to the same singular surface vertex  $v_i$ , we require that:

$$\forall u \in S_i : u \in \mathbb{Z}^2$$

- (3) **Consistent Orientation Condition:** All triangles on the parameterization domain should have the same orientation. That is, we should not allow triangle flips after initial mapping is formed.

## 2 OUR APPROACH

Inspired by [Poranne et al. 2017], we employ a direct approach to the problem of quadrangulating a triangle mesh, by formulating and solving a smooth optimization problem. We model smooth penalty functions for the first two conditions mentioned above (Seamless and Singularity Points), and add to it an additional penalty, the Symmetric Dirichlet energy, which prevents triangle flips and penalize triangle distortion. Since all of our penalty functions are smooth, we can analytically derive their gradients and Hessians,

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).  
SIGGRAPH '20 Posters, August 17, 2020, Virtual Event, USA  
© 2020 Copyright held by the owner/author(s).  
ACM ISBN 978-1-4503-7973-1/20/08.  
<https://doi.org/10.1145/3388770.3407451>

and utilize Newton's method to iteratively solve for a mapping that renders a valid quad mesh on the 3D surface.

Our smooth approach allows us to visualize the whole optimization process for the end-user as an interactive design tool, which empowers the user to guide the algorithm to the desired result by gradually changing penalty weights, grid resolution, guiding quads direction using brush tool, and more. The user receives immediate feedback for any change applied to the problem settings. Figure 1 illustrates the three main stages of our approach. First, the 3D surface is cut into the plane. Then, the seamless penalty function weight gradually increased. Finally, the singular points penalty function is turned on to place singular vertices at integer locations.

### 3 DETAILS

#### 3.1 Initialization

We cut the mesh by mapping its dual spanning tree to the parameterization plane isometrically as a triangle soup. We first map an arbitrary initial triangle to the parameterization plane. Then, we map to the plane all neighbour triangles of the initial triangle, such that adjacent triangles share an edge. We continue this process for the next layer of neighbours, till we map all of the mesh's triangles.

#### 3.2 Seamless Penalty Functions

**3.2.1 Angle Penalty.** Given two half-edges  $e_i$  and  $e_j$  on the parameterization domain that corresponds to the same surface edge, we penalize the angle between them as follows:

$$P_{\text{angle}}(e_i, e_j) = \sin\left(4(\theta(e_i) - \theta(e_j)) - \frac{\pi}{2}\right) + 1$$

Where  $\theta(e_i)$  and  $\theta(e_j)$  are the angles of the two half edges  $e_i$  and  $e_j$ , respectively. For any  $k \in \mathbb{Z}$  such that  $\theta(e_i) - \theta(e_j) = \frac{\pi}{2}k$  we have that  $P_{\text{angle}}(e_i, e_j) = 0$ . Therefore,  $P_{\text{angle}}$  will penalize half-edges of which their angle discrepancy differs from a multiple of  $90^\circ$ .

**3.2.2 Length Penalty.** We penalize the length discrepancy between the two half-edges with the following penalty function:

$$P_{\text{length}}(e_i, e_j) = \left(\|e_i\|^2 - \|e_j\|^2\right)^2$$

Only when the two half-edges have the same length, we have that  $P_{\text{length}}(e_i, e_j) = 0$ .

**3.2.3 Translation Penalty.** We penalize for non-integer translation between the two half-edges as follows:

$$P_{\text{translation}}(e_i, e_j) = \sin\left(2\pi\left(x_{e_i} - x_{e_j}\right) - \frac{\pi}{2}\right) + 1 \\ + \sin\left(2\pi\left(y_{e_i} - y_{e_j}\right) - \frac{\pi}{2}\right) + 1$$

Where  $(x_{e_i}, y_{e_i})$  and  $(x_{e_j}, y_{e_j})$  are the coordinates of two corresponding vertices of the two half-edges  $e_i$  and  $e_j$ .

#### 3.3 Singular-Points Penalty Function

To satisfy the singularity points condition, all vertices on the parameterization domain which correspond to the same vertex on the surface, have to lie on integer locations if they are characterized by a non-zero angle defect in the domain. Therefore, we penalize

singular points as follows:

$$P_{\text{singularity}}(S) = \sum_{u \in S} \left(\sin\left(2\pi x_u - \frac{\pi}{2}\right) + 1 + \sin\left(2\pi y_u - \frac{\pi}{2}\right) + 1\right)$$

Where  $S$  is a set of domain vertices with non-zero angle defect, which correspond to the same surface vertex, and  $(x_u, y_u)$  are the coordinates of the domain vertex  $u \in S$ . We weight  $P_{\text{singularity}}$  by the magnitude of the angle defect.

#### 3.4 Consistent Orientation and Distortion Penalty Function

To satisfy the consistent orientation condition, and to minimize triangle distortion, we use the symmetric dirichlet energy [Smith and Schaefer 2015], which prevents triangle flips. The symmetric dirichlet penalty function is given as follows:

$$P_{\text{dirichlet}}(t_i) = \|J(t_i)\|_F^2 + \|J^{-1}(t_i)\|_F^2$$

Where  $J(t_i)$  is the Jacobian of the mapping of triangle  $t_i$  and  $\|\cdot\|_F$  is the Frobenius norm.

#### 3.5 Optimization Process

In order to find a parameterization which forms a valid quad mesh on the 3D surface, we solve the following unconstrained optimization problem:

$$\min_X \sum_{i \sim j} P_{\text{angle}}(e_i, e_j) + P_{\text{length}}(e_i, e_j) + P_{\text{translation}}(e_i, e_j) \\ + \sum_i P_{\text{singularity}}(S_i) \\ + \sum_i P_{\text{dirichlet}}(t_i)$$

Where  $X$  denotes the set of variables of the optimization problem, i.e., the coordinates of the vertices in the parameterization domain. We derive the analytical expressions for the gradient and Hessian of each penalty function and make sure that all Hessians are positive semi-definite by zeroing negative eigenvalues. We solve the optimization problem using Newton's method. In iteration  $n$ , we evaluate the gradient  $g^{(n)}$  and Hessian  $H^{(n)}$  of our total objective function at  $X^{(n)}$ , and solve the Newton equation  $H^{(n)}p^{(n)} = -g^{(n)}$  which yields a search direction  $p^{(n)}$ . The next iterate is obtained by  $X^{(n+1)} = X^{(n)} + \alpha p^{(n)}$  where the optimal  $\alpha$  is found using line search. We use Intel's Pardiso solver to solve the sparse Newton equation.

### REFERENCES

- David Bommes, Marcel Campen, Hans-Christian Ebke, Pierre Alliez, and Leif Kobbelt. 2013a. Integer-Grid Maps for Reliable Quad Meshing. *ACM Transactions on Graphics* 32, 4 (July 2013). <https://hal.inria.fr/hal-00862648>
- David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis Zorin. 2013b. Quad-Mesh Generation and Processing: A Survey. *Comput. Graph. Forum* 32, 6 (Sept. 2013), 51–76. <https://doi.org/10.1111/cgf.12014>
- Roi Poranne, Marco Tarini, Sandro Huber, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Autocuts: Simultaneous Distortion and Cut Optimization for UV Mapping. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)* 36, 6 (2017).
- Jason Smith and Scott Schaefer. 2015. Bijective Parameterization with Free Boundaries. *ACM Trans. Graph.* 34, 4, Article 70 (July 2015), 9 pages. <https://doi.org/10.1145/2766947>