Robust and Controllable Quadrangulation of Triangular and Rectangular Regions

Robust and Controllable Quadrangulation of Triangular and Rectangular Regions

Kenshi Takayama
ETH Zurich

Daniele Panozzo
ETH Zurich

Alexander Sorkine-Hornung
Disney Research Zurich

Olga Sorkine-Hornung
ETH Zurich

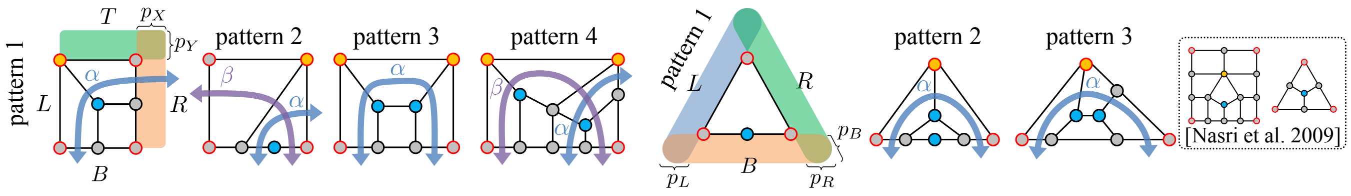

Figure 1: *Basic topological patterns used for quadrangulating rectangular and triangular regions. Vertices with a red circle represent corners. Vertices with orange and blue fill represent irregular vertices with valence three and five, respectively. p_X and p_Y represent a regular grid of quadrilaterals padded to the corresponding side of the rectangle (only shown with pattern 1 for brevity). p_B, p_R, and p_L are similarly defined for the triangle. α and β represent the number of times an edge flow is inserted at these locations. Compared to the patterns proposed by Nasri et al. [2009], all of our patterns have at least one side that has only one edge, which is important for guaranteeing valid quadrangulation of the region even when the specified number of edge subdivisions is very small and extreme.*

Keywords: quad mesh, subdivision surfaces, filling N-sided regions

1 Introduction

Remeshing an input 3D surface geometry into a coarse quad mesh with a desired mesh topology is an important process in many production pipelines, especially in the context of producing movies and video games. Algorithms for automatically generating quad meshes incorporating sparse user constraints have been studied extensively [Bommes et al. 2013]. However, they are not yet able to provide the user with direct and explicit controllability over the resulting mesh topology; the link between the user constraints and the resulting mesh topology is indirect, forcing the user to run the algorithm multiple times with different constraints and parameters to obtain the desired result. The process is tedious since the quadrangulation algorithm is not interactive. These algorithms also tend to perform poorly when coarse mesh resolutions are requested.

For these reasons, many artists today still use rather basic manual modeling tools for coarse quad remeshing that are not much different from placing vertices and faces individually [3D-Coat 2013; ZBrush 2013]. Although these tools provide complete control over the retopology process, they are slow and difficult to use, even for experienced artists. For example, it is highly non-trivial to manually design a pure quad mesh that bridges the gap between two partially quadrangulated regions of a surface.

Recently, a novel interactive system for manual quad remeshing has been proposed [Takayama et al. 2013]. The key idea is to represent a quad mesh as a set of N-sided patches containing multiple quads inside. The user is allowed to sketch patch boundaries freely and to specify any number of edge subdivision at each patch boundary. The system inserts singularities (i.e., vertices shared by more or less than four quads) inside each patch as needed. It is thus important to design an algorithm that allows quadrangulating a polygon with prescribed edge subdivisions.

This problem has already been studied in the literature [Schaefer et al. 2004; Nasri et al. 2009], among which Nasri et al.'s algorithm is the most relevant to us. While being applicable to general N-sided regions, their algorithm does not support arbitrary numbers of edge subdivision at the region boundary. For example, their algorithm fails when one side of the region is given 10 and all the other sides are given 2 as the number of edge subdivision (for any N).

We therefore propose a novel algorithm for quadrangulating rectangular and triangular regions that is provably guaranteed to generate valid quad mesh topology for any even number of edge subdivision at the region boundary.

2 Algorithm

Before describing our algorithm in detail, it is important to exclude unfeasible boundary configurations.

Proposition 1. *The number of boundary edges of any quad mesh is even.*

Proof. Denote a quad mesh as M, the number of its faces as $n_F(M)$, and the number of its boundary edges as $n_E(M)$, respectively. The proposition is proved by induction. When $n_F(M) = 1$ (i.e., M consists of a single quad), clearly $n_E(M) = 4$ and the preposition holds. Assume that the proposition holds for any quad mesh M s.t. $n_F(M) = k$ for some integer k. For a quad mesh M' s.t. $n_F(M') = k + 1$, there exists a quad mesh M s.t. $n_F(M) = k$ and a quad f where M' can be obtained by attaching f to the boundary of M. Denoting the number of edges shared by f and M (i.e., they become internal edges in M') as a, clearly $n_E(M') = n_E(M) + 4 - 2a$. $n_E(M)$ is even according to the assumption, thus $n_E(M')$ is also even. Therefore, the proposition holds for any quad mesh M' s.t. $n_F(M') = k+1$, and by induction, the proposition holds for any quad mesh. $\square$

In the following, we assume that the sum of the numbers of edge subdivisions specified at the boundary of the region is even and call this valid input. Apart from this restriction, it is possible to arbitrarily vary the edge subdivision, including extremely asymmetric subdivision counts, and our algorithm will efficiently generate a corresponding quad mesh connectivity.

2.1 Topological patterns

The basic idea is to use *topological patterns* to generate the patch connectivity given the constraints. We devise four patterns for rectangular patches and three for triangular ones (Fig. 1). Let us denote the desired number of edge subdivisions per side by B, T, R, L (for *Bottom, Top*, etc.) for 4-sided regions and B, R, L for 3-sided regions. The relation between these numbers determines which pattern will be used to construct the connectivity.

Each pattern is a "minimal" instance of a certain graph configuration, from which an infinite family of patch connectivities can be produced by varying the parameters, which are the number of edge flow insertions α and β, and the number of outer paddings: horizontal and vertical paddings for 4-sided patches (called p_X and p_Y) and triangle paddings for 3-sided patches (p_B, p_R, p_L). See Figure 2 for an illustration. A single outer padding step adds an edge loop parallel to the patch side, while insertion of inner edge flows can be performed in certain directions, as shown in Figures 1 and 2 by the blue and purple arrows. For rectangular patches, the outer padding can be performed on both opposing sides (top-bottom, left-right), which effectively translates the irregular vertices in parallel (Figure 3, top). In addition, with patterns 3 and 4 for rectangular regions and patterns 2 and 3 for triangular regions, the padding can be split and inserted in the middle of the patch, which effectively makes the distance between irregular vertices larger (Figure 3, bottom). Most importantly, any variation of these parameters does not change the number of irregular vertices – it is fixed per pattern.

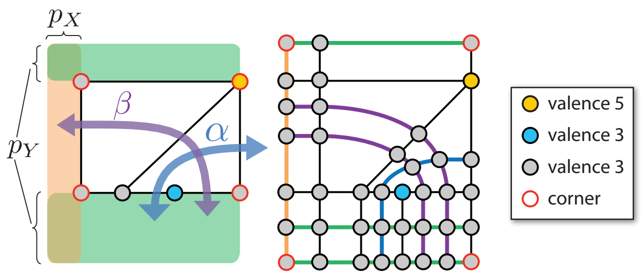

Figure 2: *Using a patch pattern (here, pattern 2, cf. Fig. 1) to generate a quad-only tessellation. Patch corners are circles with red boundary.*

In contrast to the patterns proposed by Nasri et al. [2009], all our patterns have at least one side that has only one edge, which is important when dealing with a very small number of edge subdivisions. Note that while patterns 1 and 2 for rectangular regions and the pattern 1 for triangular regions can be derived from Nasri et al.'s patterns, the other patterns are novel and necessary to support an arbitrary number of edge subdivisions.

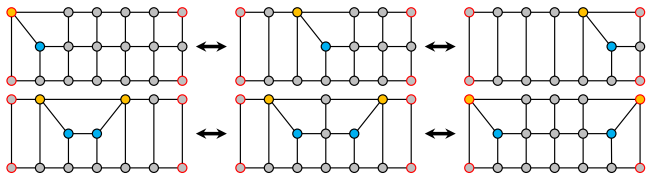

Figure 3: *Moving irregular vertices by changing the distribution of padding added to the opposing sides (top), or to the middle of the pattern (bottom).*

2.2 Pattern selection

Here we describe how a pattern is selected based on the specified number of edge subdivisions for each side of the boundary. For rectangular regions, we denote the differences between the numbers of edge subdivisions between opposing sides by $d_X = B - T$ and $d_Y = R - L$. Without loss of generality, we assume an order of sides such that $d_X \geq d_Y$. An appropriate pattern is selected based on the configuration of d_X and d_Y as follows. For simple cases where $d_X = d_Y = 0$ and $d_X = d_Y > 0$, the regular grid and pattern 1 are selected, respectively. For cases where $d_X > d_Y$, we first check if the specified number of edge subdivisions is *extreme* ($B > R + T + L$). If this is not the case, pattern 2 is selected. Otherwise, patterns 3 and 4 are selected if $d_Y = 0$ and $d_Y > 0$, respectively. For triangular regions, similarly to the above, we assume an order of sides such that $B \geq R \geq L$. We first check if the specified number of edge subdivisions is *extreme* ($B > R + L$). If this is not the case, pattern 1 is selected. Otherwise, patterns 2 and 3 are selected if $R = L$ and $R > L$, respectively.

2.3 Parameter calculation

Once an appropriate pattern is selected, the next step is to calculate its parameters such that it satisfies the specified number of boundary edge subdivisions.

Rectangular region, pattern 1. It is clear from the pattern that:

$$R = L + 1 + \alpha \tag{1}$$

which gives:

$$\alpha = R - L - 1. \tag{2}$$

The padding parameter can be derived as: $p_X = T - 1$ and $p_Y = L - 1$.

Rectangular region, pattern 2. It is clear from the pattern that:

$$R = L + \alpha - \beta \tag{3}$$
$$B = T + 2 + \alpha + \beta \tag{4}$$

which gives:

$$\alpha = (B - T + R - L - 2)/2 \tag{5}$$
$$\beta = (B - T - R + L - 2)/2. \tag{6}$$

The padding parameter can be derived as: $p_X = T - 1$ and $p_Y = R - 1 - \alpha$.

Rectangular region, pattern 3. It is clear from the pattern that:

$$B = T + 2 + 2\alpha \tag{7}$$

which gives:

$$\alpha = (B - T - 2)/2. \tag{8}$$

The padding parameter can be derived as: $p_X = T - 1$ and $p_Y = R - 1$.

Rectangular region, pattern 4. It is clear from the pattern that:

$$R = L + 1 + \alpha \tag{9}$$
$$B = T + 3 + \alpha + 2\beta, \tag{10}$$

which gives:

$$\alpha = R - L - 1 \tag{11}$$
$$\beta = (B - T - 3 - \alpha)/2. \tag{12}$$

The padding parameter can be derived as: $p_X = T - 1$ and $p_Y = L - 1$.

Triangular region, pattern 1. It is clear from the pattern that:

$$\begin{aligned} B &= 2+p_R+p_L && (13)\\ R &= 1+p_L+p_B && (14)\\ L &= 1+p_B+p_R && (15)\end{aligned}$$

which gives:

$$\begin{aligned} p_B &= (R+L-B)/2 && (16)\\ p_R &= (B+L-R-2)/2 && (17)\\ p_L &= (B+R-L-2)/2. && (18)\end{aligned}$$

Triangular region, pattern 2. It is clear from the pattern that:

$$\begin{aligned} B &= 4+p_R+p_L+2\alpha && (19)\\ R &= 1+p_B+p_L && (20)\\ L &= 1+p_B+p_R && (21)\end{aligned}$$

which gives:

$$\begin{aligned} p_B &= (R+L-B+2+2\alpha)/2 && (22)\\ p_R &= (B+L-R-4-2\alpha)/2 && (23)\\ p_L &= (B+R-L-4-2\alpha)/2, && (24)\end{aligned}$$

where α is a user-specified parameter chosen from the range $[(B-L-R-2)/2, (B-4)/2]$.

Triangular region, pattern 3. It is clear from the pattern that:

$$L=1+p_R+p_B,\quad R=2+p_L+p_B,\quad B=3+p_R+p_L+2\alpha, \qquad (25)$$

which gives:

$$\begin{aligned} p_L &= (R+B-L-4-2\alpha)/2 && (26)\\ p_R &= (L+B-R-2-2\alpha)/2 && (27)\\ p_B &= (L+R-B+2\alpha)/2, && (28)\end{aligned}$$

where α is a user-specified parameter chosen from the range $[(B-L-R)/2,\ (L+B-R-2)/2]$.

2.4 Discussion

Note that a quad mesh generated using our algorithm is not the only choice that satisfies certain number of boundary edge subdivisions. For example, Nasri et al. [2009] also considered extreme cases and proposed to attach two of their patterns side by side for rectangular regions, leading to a quad mesh topology different from ours for the same input (4). Compared to their method, our patterns ensure that irregular vertices are connected by the same edge flow, as much as possible, often resulting in more desirable meshing. If a different configuration is needed, the user can always split a single patch into two and adjust the topology as desired.

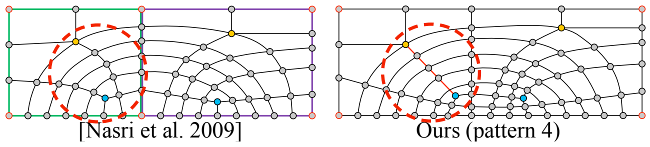

Figure 4: *Quadrangulations of a rectangular region in extreme cases using the pattern of [Nasri et al. 2009] (left) and our pattern 4 (right). Note that our pattern ensures that irregular vertices are connected by the same edge flow as much as possible.*

Extending this approach to higher number of polygonal sides ($N \geq 5$) is left for future work, as the number of possible cases to be considered grows in a combinatorial manner. In practice, the user can easily fill pentagonal or other polygonal regions by combining rectangular and triangular patches.

3 Proof of algorithm generality

In this section, we prove that our algorithm can quadrangulate triangular and rectangular regions with arbitrary number of edge subdivisions specified at the region boundary.

3.1 Patterns for rectangular region

We use the same notation as in Section 2.2: $d_X = B-T, d_Y = R-L$ and assume $d_X \geq d_Y \geq 0$.

If $d_X = 0$, the region can be trivially quadrangulated with a regular grid.

In the following, we consider the condition for a pattern to be able to realize a valid quadrangulation given (B,R,T,L) as input, which we call *realizability condition* hereafter. In general, a pattern's realizability condition is derived by restricting its parameters to be non-negative.

Assuming $d_X = d_Y > 0$, consider the realizability condition of pattern 1:

$$\begin{aligned} \alpha &= R-L-1 \geq 0 && (29)\\ p_X &= T-1 \geq 0 && (30)\\ p_Y &= L-1 \geq 0. && (31)\end{aligned}$$

The first condition always holds under the assumption: $R-L = d_Y \geq 1 > 0$. The second and third conditions obviously hold. Therefore pattern 1 covers all cases where $d_X = d_Y > 0$.

Assuming $d_X > d_Y$, consider the realizability condition of pattern 2:

$$\begin{aligned} \alpha &= (B-T+R-L-2)/2 \geq 0 && (32)\\ \beta &= (B-T-R+L-2)/2 \geq 0 && (33)\\ p_X &= T-1 \geq 0 && (34)\\ p_Y &= R-1-\alpha \geq 0. && (35)\end{aligned}$$

Note that $d_X >= d_Y + 2$ because otherwise (if $d_X = d_Y + 1$) the sum of (B,R,T,L) would become odd, which violates our assumption. This ensures that the first and second conditions always hold. The third condition also obviously holds. The fourth condition gives:

$$B \leq R+T+L \qquad (36)$$

which is the realizability condition of pattern 2. Therefore pattern 2 covers all cases where $d_X > d_Y$ as long as its realizability condition holds: $B \leq R+T+L$. Note that Nasri et al. [2009] also mention this condition.

In the following, we assume $d_X > d_Y$ and $B > R+T+L$. Assuming $d_Y = 0$, consider the realizability condition of pattern 3:

$$\begin{aligned} \alpha &= (B-T-2)/2 \geq 0 && (37)\\ p_X &= T-1 \geq 0 && (38)\\ p_Y &= R-1 \geq 0. && (39)\end{aligned}$$

The first condition always holds because $d_X \geq d_Y + 2 \geq 2$ as shown previously. The second and third conditions obviously hold. Therefore pattern 3 covers all cases where $d_X > d_Y = 0$.

Assuming $d_Y > 0$, which is the last remaining case, consider the realizability condition of pattern 4:

$$\begin{aligned}\alpha &= R - L - 1 \geq 0 && (40)\\ \beta &= (B - T - 3 - \alpha)/2 \geq 0 && (41)\\ p_X &= T - 1 \geq 0 && (42)\\ p_Y &= L - 1 \geq 0. && (43)\end{aligned}$$

The first condition holds from the assumption. The second condition holds because $B-T-3-\alpha = d_X-3-d_Y+1 = d_X-d_Y-2 \geq 0$ as shown previously. The third and fourth conditions obviously hold. Therefore pattern 4 covers all cases where $d_X > d_Y > 0$.

3.2 Patterns for triangular region

As in Section 2.2, we assume $B \geq R \geq L$. The proof is similar to the above section.

Consider the realizability condition of pattern 1:

$$\begin{aligned}p_B &= (R + L - B)/2 \geq 0 && (44)\\ p_R &= (B + L - R - 2)/2 \geq 0 && (45)\\ p_L &= (B + R - L - 2)/2 \geq 0. && (46)\end{aligned}$$

We show that this condition always holds when $B = R$. The first condition holds obviously. The second condition holds because L cannot be smaller than 2 (otherwise the sum of (B, R, L) would become odd). The third condition holds because B cannot be smaller than 2. Therefore pattern 1 covers all cases where $B = R \geq L$.

In the following, we assume $B > R$ (or equivalently $B \geq R + 1$) and consider the realizability condition of pattern 1. The second condition holds because $B + L - R - 2 \geq (R + 1) + L - R - 2 = L - 1 \geq 0$. The third condition holds because B cannot be smaller than 2. The first condition gives:

$$B \leq R + L \qquad (47)$$

which is the realizability condition of pattern 1. Therefore pattern 1 covers all cases as long as its realizability condition holds: $B \leq R + L$. Note that Nasri et al. [2009] also mention this condition.

In the following, we assume $B > R + L$. Note that this means $B \geq R + L + 2$, because otherwise (i.e., $B = R + L + 1$) the sum of (B, R, L) would become odd.

Further assuming that $R = L$, consider the realizability condition of pattern 2. Because pattern 2 has a user-specified parameter α, its realizability condition is parameterized by α:

$$\begin{aligned}p_B &= (R + L - B + 2 + 2\alpha)/2 \geq 0 && (48)\\ p_R &= (B + L - R - 4 - 2\alpha)/2 \geq 0 && (49)\\ p_L &= (B + R - L - 4 - 2\alpha)/2 \geq 0. && (50)\end{aligned}$$

This condition gives one lower bound and two upper bounds for α. In order for α that satisfies this condition to exist, the following condition must hold:

$$\max(B-R-L-2, 0) \leq \min(B+R-L-4, B+L-R-4). \qquad (51)$$

This condition holds because

$$\begin{aligned}\max(B-R-L-2, 0) &= B - R - L - 2\\ &\leq B - 4\\ &= \min(B + R - L - 4, B + L - R - 4)\end{aligned}$$

using the fact that $B \geq R + L + 2$ and $R = L \geq 1$. Therefore pattern 2 covers all cases where $B > R + L$ and $R = L$.

Assuming $R > L$, which is the last remaining case, consider the realizability condition of pattern 3. Similar to the above, its realizability condition is parameterized by α:

$$\begin{aligned}p_L &= (R + B - L - 4 - 2\alpha)/2 && (52)\\ p_R &= (L + B - R - 2 - 2\alpha)/2 && (53)\\ p_B &= (L + R - B + 2\alpha)/2. && (54)\end{aligned}$$

This condition gives one lower bound and two upper bounds for α. In order for α to exist that satisfies this condition, the following condition must hold:

$$\max(B-L-R, 0) \leq \min(B+R-L-4, B+L-R-2). \qquad (55)$$

For the right hand side, we have

$$\begin{aligned}B + R - L - 4 &\geq B + (L + 1) - (R - 1) - 4 && (56)\\ &= B + L - R - 2. && (57)\end{aligned}$$

Finally we show that the condition holds:

$$\begin{aligned}\min(B + R - L - 4, B + L - R - 2) &= B + L - R - 2\\ &\geq B + L - R - 2L\\ &= B - L - R\\ &= \max(B - L - R, 0).\end{aligned}$$

Therefore pattern 3 covers all cases where $B > R + L$ and $R > L$.

References

3D-COAT, 2013. Pilgway. Version V3, `http://3d-coat.com/`.

BOMMES, D., LEVY, B., PIETRONI, N., PUPPO, E., SILVA, C., TARINI, M., AND ZORIN, D. 2013. Quad-mesh generation and processing: A survey. *Comput. Graph. Forum XX*, X.

NASRI, A., SABIN, M., AND YASSEEN, Z. 2009. Filling n-sided regions by quad meshes for subdivision surfaces. *Comput. Graph. Forum 28*, 6, 1644–1658.

SCHAEFER, S., WARREN, J., AND ZORIN, D. 2004. Lofting curve networks using subdivision surfaces. In *Proc. SGP*, 103–114.

TAKAYAMA, K., PANOZZO, D., SORKINE-HORNUNG, A., AND SORKINE-HORNUNG, O. 2013. Sketch-based generation and editing of quad meshes. *ACM Trans. Graph. 32*, 4.

ZBRUSH, 2013. Pixologic, Inc. Version 4.4, `http://www.pixologic.com/zbrush/`.