Eurographics Symposium on Geometry Processing 2014
Thomas Funkhouser and Shi-Min Hu
(Guest Editors)

Volume 33 (2014), Number 5

Pattern-Based Quadrangulation for N-Sided Patches

Kenshi Takayama Daniele Panozzo Olga Sorkine-Hornung

ETH Zurich, Switzerland

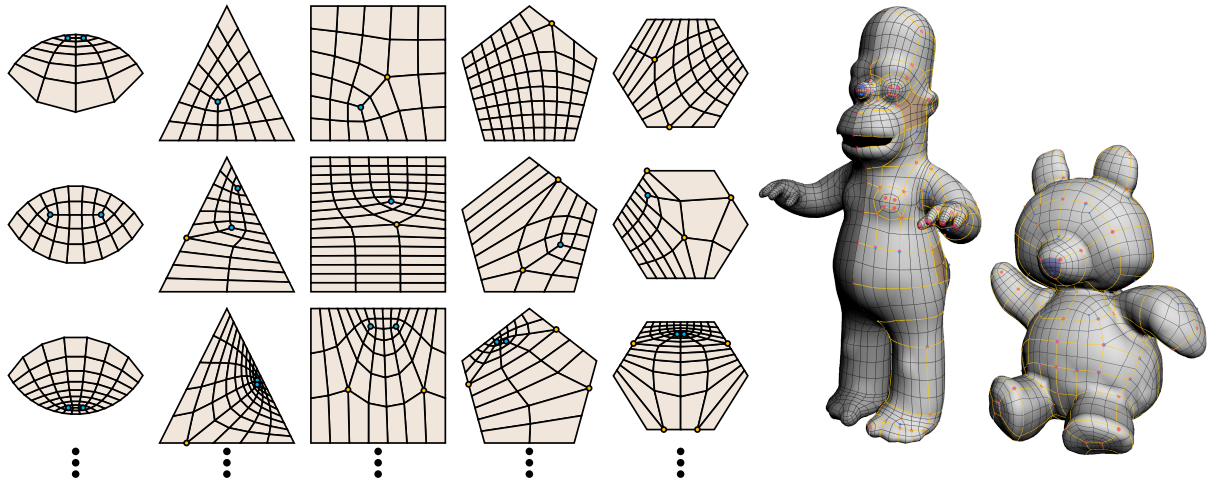


Figure 1: *(Left) Quadrangulations of N-sided patches using our algorithm, where $2 \leq N \leq 6$. Our algorithm is guaranteed to successfully quadrangulate N-sided patches with arbitrary numbers of edge subdivisions specified at the patch boundary. (Right) Integration of our algorithm into the UI framework of Takayama et al. [TPSHSH13b] to efficiently retopologize triangle meshes.*

Abstract

We propose an algorithm to quadrangulate an N-sided patch ($2 \leq N \leq 6$) with prescribed numbers of edge subdivisions at its boundary. Our algorithm is guaranteed to succeed for arbitrary valid input, which is proved using a canonical simplification of the input and a small set of topological patterns that are sufficient for supporting all possible cases. Our algorithm produces solutions with minimal number of irregular vertices by default, but it also allows the user to choose other feasible solutions by solving a set of small integer linear programs. We demonstrate the effectiveness of our algorithm by integrating it into a sketch-based quad remeshing system. A reference C++ implementation of our algorithm is provided as a supplementary material.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

Converting a triangle mesh into a coarse quad mesh with a desired mesh connectivity is an important step in many production pipelines, especially in the context of movies and video games. The process is often called *retopology*. Algorithms for automatic generation of quad meshes incorporating sparse user constraints such as alignment of edges [BZK09, ILS*11] and singularities [MPKZ10] have been studied extensively [BLP*13]. Other works [BLK11, TPP*11] considered the problem of altering the connectivity of existing

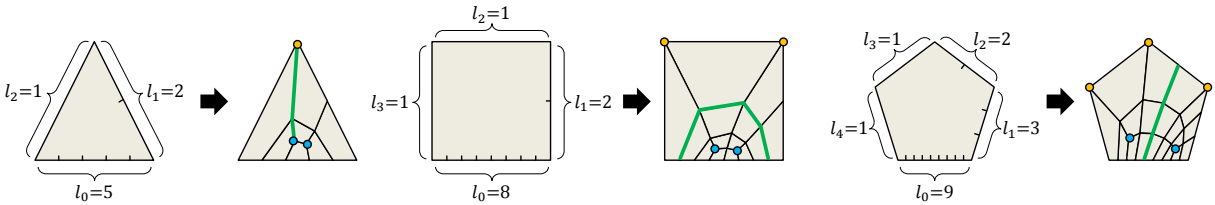


Figure 2: *Examples of input and output of our quadrangulation algorithm. Points with blue and yellow fill represent singularities with valence 3 and 5, respectively. Some of the edge loops are highlighted in green.*

quad meshes to align singularities, thus defining a coarse quad layout. The direct generation of these layouts from the input surface geometry has also been recently considered [CBK12, BCE*13]. Such approaches are implemented in several commercial 3D modeling tools [Pil13, Pix13].

However, most of these methods involve global optimization processes that are computationally expensive. Tweaking their parameters is a time-consuming task, since for each parameter change the entire quadrangulation has to be regenerated. Furthermore, there is no direct intuitive connection between the user constraints and the resulting mesh topology; a local change in the user constraints might generate a globally different connectivity of quadrilaterals.

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 [Pil13, Pix13]. Although these tools provide complete control over the retopology process, they are difficult to use, even for experienced artists, and require time-consuming manual work. For example, it is highly nontrivial to manually design a pure quad mesh that bridges a gap between two partially quadrangulated regions of a surface.

To alleviate this difficulty, Takayama et al. proposed an interactive tool for manual quad remeshing [TPSHSH13b]. Their key idea is to represent a quad mesh as a set of N-sided patches containing multiple quads inside. The user is allowed to freely sketch patch boundaries and specify any number of edge subdivisions at each patch boundary. The system immediately quadrangulates each patch while inserting singularities inside patches as needed. The quadrangulation of an N-sided patch with prescribed numbers of edge subdivisions at the patch boundary is thus fundamental to this system.

While there already exist a few algorithms to solve this problem [SWZ04, NSY09, YNB*13, ZHLB13, dOMM13, PBJW14], none of them provide guarantee to succeed for arbitrary cases; in particular, these algorithms do not seem to be able to handle very asymmetric cases where some sides of the patch are subdivided many times while other sides are not subdivided at all (see Figure 2 for some examples). This is rather a serious shortcoming, since the ability to handle arbitrary cases is a necessary requirement in order to ensure the flexibility of an artist using the sketch-based retopology tool [TPSHSH13b].

Takayama et al. [TPSHSH13a] proposed a robust algorithm for quadrangulating 3-sided and 4-sided patches with arbitrary numbers of edge subdivisions at the patch boundary, and integrated it into their sketch-based retopology system [TPSHSH13b]. While their algorithm also uses a pattern-based approach similar to ours, it does not generalize well to patches with more than 4 sides, because it classifies the input into several cases by directly comparing the numbers of edge subdivisions given to the patch sides, leading to a significantly large number of cases to be considered already for 5-sided patches.

In this paper, we propose a more general algorithm for quadrangulating N-sided patches with $2 \leq N \leq 6$. Our key observation is that the initial problem can be reduced into an equivalent and much smaller problem, which considerably decreases the number of cases that need to be considered (Section 2.1). Based on this observation, we provide a complete set of topological patterns that covers all possible inputs, thus ensuring our algorithm's generality (Section 2.2). We then show that the problem of determining whether a certain topological pattern can satisfy a given input can be concisely formulated as a small integer linear program (Section 2.3). We also provide simple ways to explore the space of possible quadrangulations for the given input (Sections 2.4 and 2.5). As shown in Figure 1, our algorithm can generate quadrangulations for arbitrary input and proves useful when integrated into the interactive system by Takayama et al. [TPSHSH13b]. A reference C++ implementation of our algorithm is provided as a supplementary material.

2. Algorithm

We consider a tessellation of an N-sided patch into a quad mesh with N corner vertices at the boundary, where $2 \leq N \leq 6$. A *side* of the patch is defined as the mesh boundary between two consecutive corner vertices. The *valence* of an interior vertex is defined as the number of its adjacent edges. The valence of a corner boundary vertex is defined as the number of its adjacent edges plus two, and the valence of a non-corner boundary vertex is the number of its adjacent edges plus one. A vertex with valence four is called a *regular* vertex, while a vertex with other valence is called an *irregular*

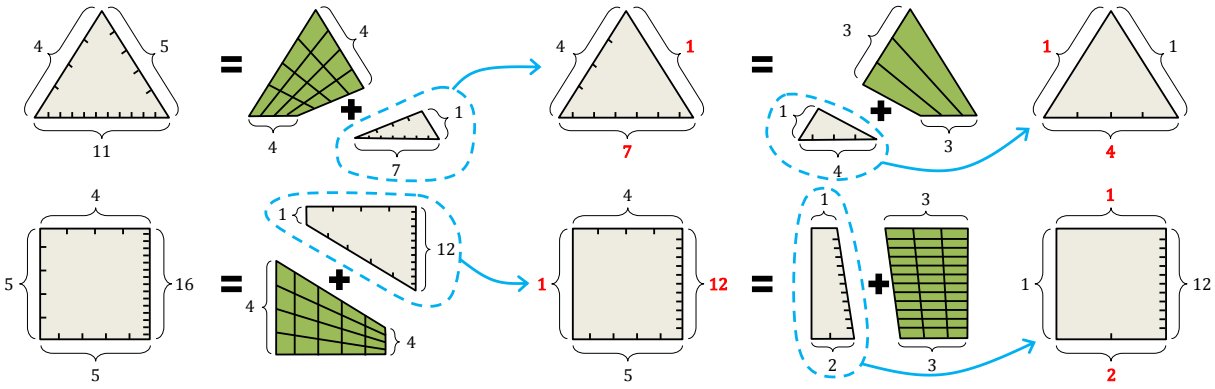


Figure 3: *Examples of maximally reducing the input for 3- and 4-sided patches. After each reduction operation, a pair of sides get smaller numbers of edge subdivisions, shown in red.*

vertex or a *singularity*. An *edge flow* or an *edge loop* is a sequence of edges that starts from and ends at a singularity or a patch boundary and goes through regular vertices such that every pair of consecutive edges does not belong to the same quad.

The input to our algorithm is the number of edge subdivisions for each side $(l_0, \cdots, l_{N-1}) \in \mathbb{Z}_+^N$ where l_i corresponds to the i-th side, and the output is a quad mesh that satisfies the input with as few singularities as possible. In this paper, the side indices are always taken modulo N. Figure 2 shows examples of input and output for the quadrangulation of 3-, 4-, and 5-sided patches.

Note that it is impossible to quadrangulate the input if the sum of the provided numbers of edge subdivisions, $\sum_{i=0}^{N-1} l_i$, is odd. To see this, consider an arbitrary quad mesh patch with F faces, E_b boundary edges, and E_i internal edges. Since every quad has four edges and every internal edge is shared by two quads, these numbers are related as $4F = E_\mathrm{b} + 2E_\mathrm{i}$, meaning that E_b must always be even.

Therefore, we assume in this paper that $\sum_{i=0}^{N-1} l_i$ is even. Apart from this restriction, it is possible to arbitrarily vary the numbers of edge subdivisions l_i, and our algorithm will efficiently generate a corresponding quad mesh connectivity.

Our algorithm only determines the topological information, i.e., the mesh connectivity. For visualization purposes in this paper, we use 2D mesh vertex positions $\{\mathbf{v}_i\} \subset \mathbb{R}^2$ obtained by the uniform Laplacian smoothing:

$$\underset{\{\mathbf{v}_i\}}{\operatorname{argmin}} \ \sum_{(i,j)\in\mathcal{E}} \|\mathbf{v}_i - \mathbf{v}_j\|^2 \tag{1}$$

$$\text{subject to} \quad \mathbf{v}_i = \mathbf{w}_i\,, \quad i \in \mathcal{C} \tag{2}$$

where $\mathcal{E}$ is the set of edges of the mesh, $\mathcal{C}$ is the set of boundary vertices, and $\mathbf{w}_i \in \mathbb{R}^2$ is the predefined 2D boundary position to which the i-th boundary vertex is fixed. In actual retopology scenarios, these 2D vertex positions are mapped to 3D surface positions using the parameterization.

2.1. Reducing the input

It is challenging to consider all possible input configurations, since their number increases exponentially as we increase N. Our key observation is that the problem can be reduced to an equivalent, easier subproblem by *trimming* the patch. This operation drastically reduces the number of cases we need to consider to ensure our algorithm's generality.

Assume an input $(l_0, \cdots, l_{N-1})$ where l_{k-1} and l_{k+1} are both greater than 1 for some k, and let us define $d = \min(l_{k-1}, l_{k+1}) - 1$. Then, the quadrangulation of the original input can be achieved by first quadrangulating another, smaller input $(l'_0, \cdots, l'_{N-1})$ where

$$l'_i = \begin{cases} l_i - d & \text{if } \ i = k \pm 1 \\ l_i & \text{otherwise,} \end{cases}$$

and then attaching a d-by-l_k regular grid to the k-th side of the resulting intermediate quadrangulation. We call this operation *trimming*, and its inverse *padding*. The trimming operation can be repeated until no more trimming is possible, obtaining a *maximally reduced* input. Figure 3 shows examples of maximally reducing the input for 3- and 4-sided patches.

Maximally reduced input can be classified into a small number of cases for $N \leq 6$, as shown in Table 1, where α and β represent arbitrary numbers greater than 1. The requirement for an input to be maximally reduced is that for all k, at least one of l_{k-1} and l_{k+1} must be 1. Note that the cases are classified modulo rotations and inversions, e.g., $(\alpha, 1, 1) \equiv (1, \alpha, 1) \equiv (1, 1, \alpha)$. *Doublets*, i.e., patches with $N = 2$, are handled separately (Section 2.6).

2.2. Topological patterns

Figure 4 shows a complete set of topological patterns that covers all cases of maximally reduced input. The blue and purple arrows with associated symbols x and y shown on top of some patterns are *parameters* of these patterns, indicating

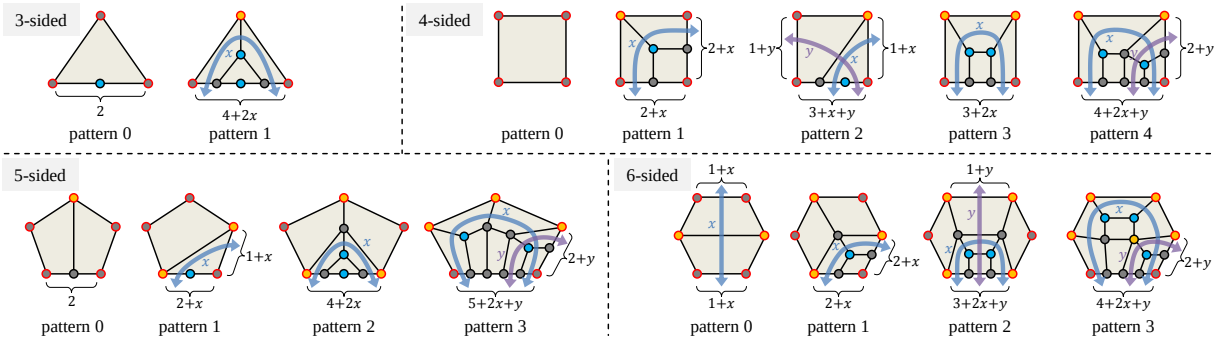


Figure 4: *A complete set of topological patterns that covers all cases of maximally reduced input. Vertices with red borders represent patch corners. The number of edge subdivisions is depicted for each patch side if it is not 1.*

N	Input	Condition	Pattern
3	$(\alpha,1,1)$	$\alpha=2$	0
		$\alpha=4+2x$	1
4	$(1,1,1,1)$		0
	$(\alpha,1,1,1)$	$\alpha=3+2x$	2 $(x=0)$, 3
	$(\alpha,\beta,1,1)$	$\alpha=\beta$	1
		$\alpha=\beta+2+2x$	4
5	$(\alpha,1,1,1,1)$	$\alpha=2$	0
		$\alpha=4+2x$	2
	$(\alpha,\beta,1,1,1)$	$\alpha=\beta+1$	1
		$\alpha=\beta+3+2x$	3
6	$(1,1,1,1,1,1)$		0 $(x=0)$
	$(\alpha,1,1,1,1,1)$	$\alpha=3+2x$	2 $(y=0)$
	$(\alpha,\beta,1,1,1,1)$	$\alpha=\beta$	1
		$\alpha=\beta+2+2x$	3
	$(\alpha,1,1,\beta,1,1)$	$\alpha=\beta$	0
		$\alpha=\beta+2+2x$	2

Table 1: *Exhaustive set of rules that covers all valid maximally-reduced inputs ($3 \le N \le 6$). For each input, the corresponding conditions cover all the valid values of α and β under the assumption that the total number of edge subdivisions is even.*

insertion of edge flows at these locations x and y times ($x \ge 0$ and $y \ge 0$). Figure 5 shows examples of inserting edge flows into some patterns.

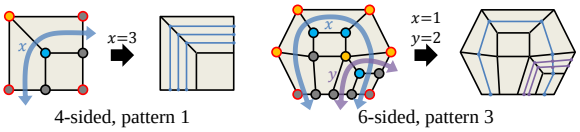


Figure 5: *Examples of inserting edge flows into some patterns.*

Table 1 shows that any possible maximally reduced input can be quadrangulated using at least one of the patterns. As an example, let us consider the input of the form $(\alpha,1,1)$. Only inputs with α being even can be quadrangulated, since the sum of the numbers of edge subdivisions has to be even. We now consider two cases. For the case where $\alpha=2$, we apply pattern 0 directly. For all the other cases where $\alpha=4+2x$, we apply pattern 1. We consider all the other forms of input in a similar manner. In some cases, different patterns can be applied to the same input; we discuss how to select a pattern with minimal number of singularities and how to let the user choose other feasible patterns in Sections 2.3 and 2.5, respectively.

Strategy for designing patterns. We designed the patterns such that a minimal number of singularities of valence 3 or 5 are introduced. We first considered the case with one side having arbitrary number of edge subdivisions, and all the other sides having only one edge, i.e., $(\alpha,1,\cdots,1)$. Then we considered the case with two adjacent sides having arbitrary numbers of edge subdivisions, and all the other sides having only one edge, i.e., $(\alpha,\beta,1,\cdots,1)$. While some of the patterns seem to have similar topological structures in common, we have not yet found a way to automatically generate these patterns for a given N; we designed them manually using the connectivity editing operators [PZKW11].

2.3. Pattern selection formulated as ILP

So far we have shown that it is possible to quadrangulate N-sided patches with arbitrary numbers of edge subdivisions by first maximally reducing the input and then choosing one of the proposed patterns. However, as shown in Figure 6, this procedure may lead to the selection of an unnecessarily complex pattern with more singularities than needed. To select the pattern with a minimal number of singularities, we look for all feasible patterns and pick the one with the fewest singularities.

We observe that each pattern defines a simple linear relation between the input $(l_0,\cdots,l_{N-1})$ and the parameters, i.e., the amount of padding of the i-th side p_i and the number of

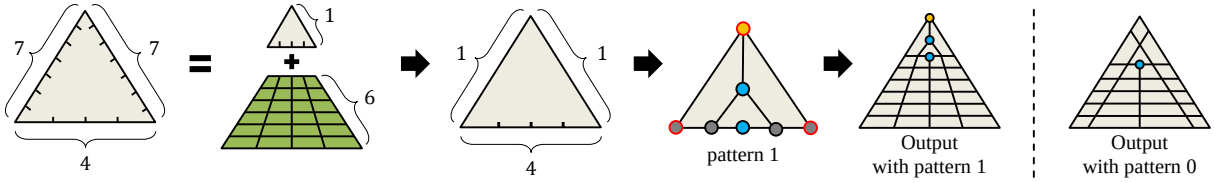


Figure 6: *An example of quadrangulating a 3-sided patch where maximally reducing the input leads to the selection of pattern 1 which is more complex than pattern 0 satisfying the same input.*

edge flow insertions x and y; see Figure 7 for some examples. This relation can be written in matrix form as

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{3}$$

where $\mathbf{A}$ is an $N \times M$ matrix (constant per pattern) with M being the number of the pattern's parameters, $\mathbf{x}$ an M-dimensional vector representing the pattern's parameters (unknown), and $\mathbf{b}$ an N-dimensional vector determined by the input $(l_0, \cdots, l_{N-1})$ and the pattern. An important requirement is that each parameter $x_i \in \mathbf{x}$ must be a non-negative integer. A pattern is then feasible if Eq. (3) has non-negative integer solutions. This can be formulated as an integer linear program (ILP):

$$\begin{aligned} \text{maximize} \quad & \mathbf{c}^\top \mathbf{x} && (4)\\ \text{subject to} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} && (5)\\ & \mathbf{x} \geq \mathbf{0}, \mathbf{x} \in \mathbb{Z}^M && (6) \end{aligned}$$

where $\mathbf{c}$ is an M-dimensional vector representing the objective we wish to maximize, which will be discussed later. The existence of solutions to this ILP means that the pattern is feasible for the input. Since the number of variables M is small (at most 10 in our case), it can be solved very quickly. To account for the permuted versions of the input (Section 2.1), we test if the pattern is feasible for all the permuted versions of the input. We further perform this test sequentially for each of the patterns in the ascending order of index (i.e., patterns with fewer singularities are tested first), and as soon as a feasible solution is found, we use the corresponding pattern, parameter, and permutation to generate the quadrangulation accordingly. Our pattern design (Section 2.2) ensures that *at least one* solution will be found during this process.

The ILP often has multiple solutions since $N \leq M$. This means that there are multiple parameters for the same pattern satisfying the input. In the next section, we show how to interactively explore the solution space.

2.4. Adjusting pattern parameters

As shown in Figure 8, some patterns accommodate additional edge flows other than those depicted in Figure 4. These are treated as additional variables in the ILP. Since the difference between the number of variables and the number of constraints $M - N$ is always less than 5, the space of feasible solutions is at most 4-dimensional.

The default parameters are computed by maximizing the sum of the paddings $\sum_{i=0}^{N-1} p_i$. This is achieved by setting $c_j = 1$ if the corresponding variable represents padding, and $c_j = 0$ elsewhere. To let the user freely and intuitively explore the space of feasible solutions, we propose a simple user interface where the user can adjust a certain parameter while keeping changes to other parameters minimal (Figure 9 top). Let us denote the current and new sought parameters as $\tilde{\mathbf{x}}$ and $\mathbf{x}$, respectively, and suppose we want to increase the k-th parameter x_k. First, we specify an additional constraint as

$$x_k \geq \tilde{x}_k + 1. \tag{7}$$

Then, we conceptually set the objective as

$$\text{minimize} \quad \sum_{j=0}^{M-1} |x_j - \tilde{x}_j|. \tag{8}$$

To handle this form of objective in ILP, we follow the conventional approach: we introduce auxiliary variables $\mathbf{y} = (y_0, \cdots, y_{M-1})$, set the objective as

$$\text{minimize} \quad \sum_{j=0}^{M-1} y_j, \tag{9}$$

and specify additional constraints as

$$\begin{aligned} y_j &\geq -(x_j - \tilde{x}_j) && (10)\\ y_j &\geq x_j - \tilde{x}_j && (11) \end{aligned}$$

for all $j \in \{0, \cdots, M-1\}$. This doubles the number of variables in the ILP, but it is still small and can be solved quickly. When the space of feasible solutions becomes empty due to the additional constraint (7), the system tells the user that the requested parameter adjustment is impossible.

Notice that the coefficient matrix for pattern 4 for 4-sided patches shown in Figure 7 contains two pairs of columns that are identical: the first and the third columns (corresponding to p_0 and p_2), and the second and the fourth columns (corresponding to p_1 and p_3). These parameters with the identical columns in the coefficient matrix affect the number of edge subdivisions at the patch boundary in the same manner. Adjusting these parameters while keeping their sum constant can be seen as topologically translating singularities (Figure 9 middle). By mapping the user's dragging of the mouse to the adjustment of these parameters, we provide an intuitive interface for translating singularities. The number of additional

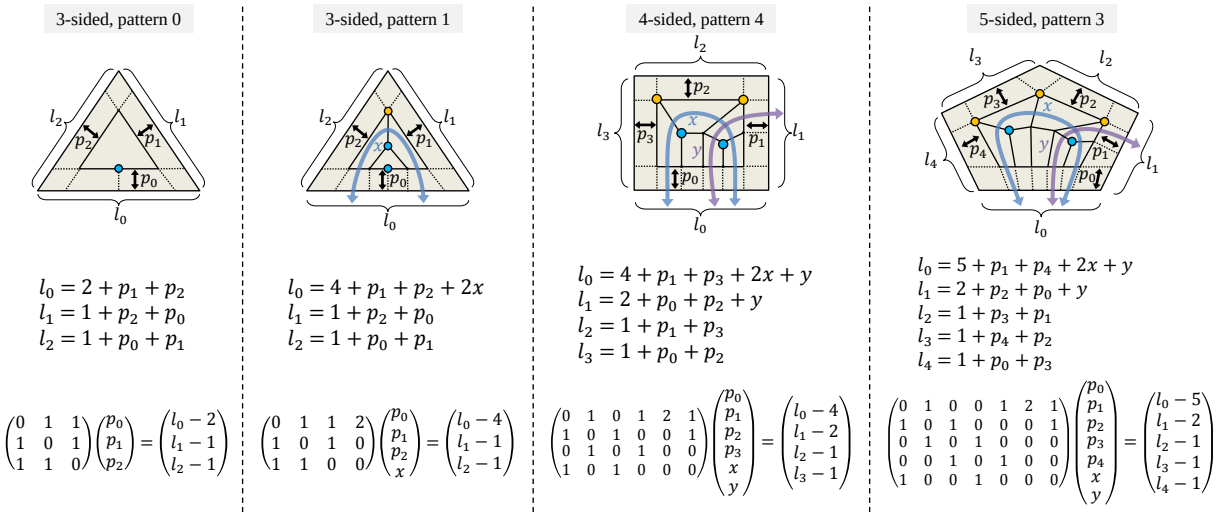


Figure 7: *Linear relations between the input and the parameters for four patterns.*

edge flows, q_i, introduced in Figure 8, is indexed in such a way that its corresponding column in the coefficient matrix is identical to that of the padding parameter p_i, enabling a similar operation of topologically translating singularities (Figure 9 bottom).

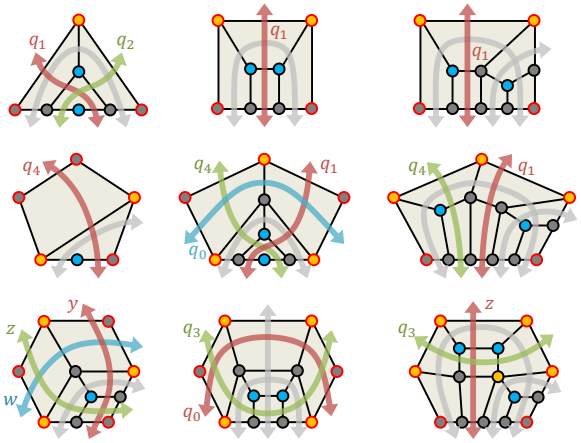


Figure 8: *Additional edge flows for some patterns which are treated as additional variables in ILP.*

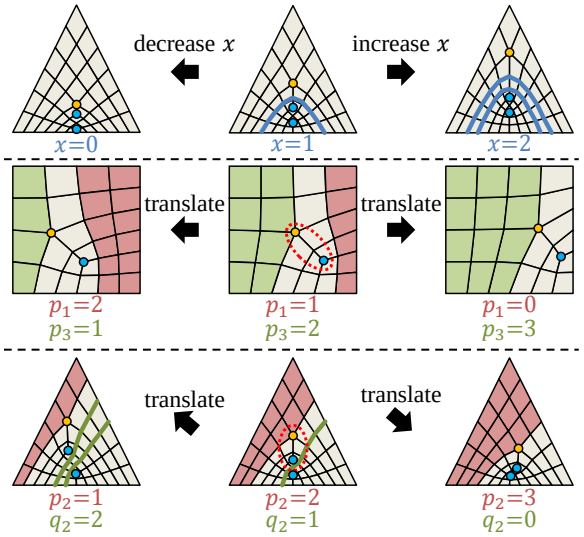


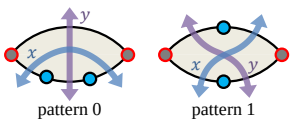
Figure 9: *A simple user interface for adjusting parameters. (Top) The user can increase or decrease a selected parameter. (Middle and bottom) Exchanging values between certain pairs of parameters amount to topologically translating singularities.*

2.5. Switching between patterns

As mentioned in Section 2.2, different patterns may satisfy the same input (Figure 10 left). Furthermore, the same pattern with different permutations may satisfy the same input (Figure 10 middle). We can offer the user further flexibility by allowing her to switch among these feasible patterns and permutations. In addition, we can add more patterns to the algorithm that are redundant for ensuring the algorithm's generality, but useful for providing the user with more options; for example, Figure 10 right shows pattern 2 for 3-sided patches (as proposed by Takayama et al. [TPSHSH13a]) which can be optionally chosen by the user.

2.6. 2-sided patches

The maximally reduced input for *doublets* (2-sided patches) can only take the



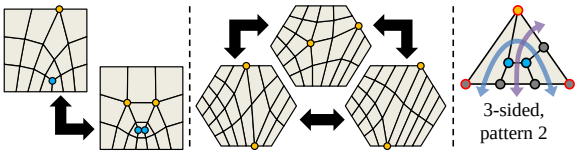


Figure 10: *Switching patterns and permutations. We can allow the user to choose (left) different patterns and (middle) different permutations within the same pattern. (Right) We can also incorporate additional patterns into the algorithm to provide the user with more options.*

form $(A,1)$. Due to our assumption of valid input, A must be odd, i.e., $A = 3+2x$, which is covered by pattern 0 (see inset). An exceptional singular case that cannot be handled by pattern 0 is the input of $(2,2)$, which is covered by pattern 1.

3. Results

We implemented our algorithm using OpenMesh [BSBK02] for handling the mesh connectivities and lp_solve [BDE*10] for solving the ILP. Note that the maximal reduction procedure described in Section 2.1 is only used to prove the generality of our algorithm. A reference C++ implementation of our algorithm is provided as a supplementary material.

Figure 1 (left) shows a few examples of quadrangulations generated by our algorithm. The computation time for generating a mesh connectivity of this level of density is at most around 30 ms on an Intel Core i7 2.67GHz CPU, where the steps of the ILP solve and the Laplacian energy minimization are equally dominant. Figure 1 (right) shows two results created using our algorithm integrated into Takayama et al.'s system [TPSHSH13b]. The modeling sessions took about 70 and 40 minutes for the *Bear* and the *Humanoid* models, respectively. While the design process is inherently artistic and exploratory, making quantitative comparisons difficult, existing manual retopology tools [Pil13, Pix13] tend to require roughly twice or more time to achieve similar results.

Comparisons. Figure 11 shows comparisons between Yasseen et al.'s results (top row), reported in Fig. 9 of their paper [YNB*13] and our results (bottom row). Our results have fewer singularities (left) or the same number of singularities with a different layout (middle). On the top right example, the sum of the input numbers of edge subdivisions is odd, and their algorithm necessarily introduces a triangular element to handle this invalid input. If we change the input slightly by incrementing the number of edge subdivisions at the top right side, our method produces a topologically similar result using pattern 3 for 5-sided patches (bottom right). Most importantly, our method is guaranteed to generate quadrangulations with known number of singularities (determined by the predefined patterns) for arbitrary input. To the best of our knowledge, this guarantee was never provided by any previous algorithm.

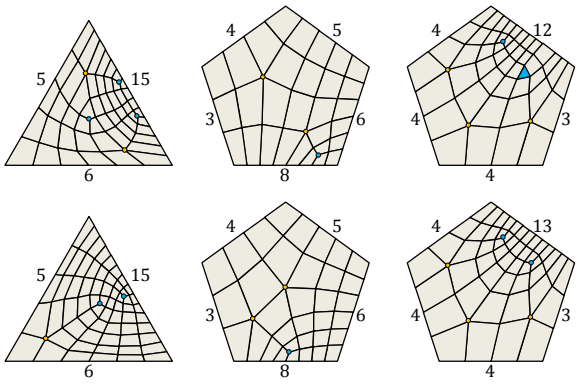
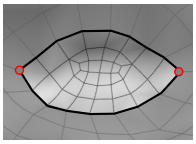


Figure 11: *Comparisons between Yasseen et al.'s results [YNB*13] (top row) and ours (bottom row).*

Our algorithm, as opposed to previous ones [YNB*13, PBJW14], allows singularities on the patch boundary. While this might be problematic for some applications, it is a key feature for our main target application of sketch-based quad meshing, since we want to always generate a quad mesh for every valid input specified by the user, including the cases with no subdivision on some sides of the polygon.

4. Limitations and future work



Our method generates only a subset of possible quadrangulations satisfying the same input because of the use of predefined patterns. If the user wants a certain quadrangulation which cannot be represented by any of these patterns (e.g., as in the inset showing a 2-sided case), she has to manually split the polygon into separate parts. Enlarging the solution space of our method by creating more complex patterns is left as future work.

Designing patterns for cases with $N \geq 7$ is another subject for future work. All possible cases of maximally reduced input for $N \in \{7,8\}$ are as follows:

N	Maximally-reduced input
7	$(\alpha,1,1,1,1,1,1),(\alpha,\beta,1,1,1,1,1),$ $(\alpha,1,1,\beta,1,1,1),(\alpha,\beta,1,1,\gamma,1,1)$
8	$(1,1,1,1,1,1,1,1),(\alpha,1,1,1,1,1,1,1),$ $(\alpha,\beta,1,1,1,1,1,1),(\alpha,1,1,\beta,1,1,1,1),$ $(\alpha,1,1,1,\beta,1,1,1),(\alpha,\beta,1,1,\gamma,1,1,1)$

Some of these cases involve three sides having more than one edge subdivisions (i.e., α, β, and γ) which may complicate the pattern design and drastically increase the number of patterns required. We have not yet attempted to manually design patterns for these cases, because the practical relevance did not seem worth the effort. An automatic algorithm for generating the minimal set of required patterns for arbitrary N is an interesting direction for future work.

Sophisticated geometric optimization could be utilized to further enhance the usability of our algorithm. For example, the uniform Laplacian smoothing used in our implementation causes flipped quadrilaterals when the shape of the patch is not convex. This issue would be solved by using one of the recent bijective mapping techniques [SHF13]. We could also take the geometric features of the 3D shape being retopologized into account, or consider the geometric qualities of the resulting quad mesh when choosing the best quadrangulation in the feasible set, similar to Peng et al.'s approach [PBJW14].

Lastly, generalization of our approach to hexahedral mesh generations is an interesting and challenging direction for future work. This is highly relevant especially in the context of finite element analysis. One possible approach would be to let the user decompose the model's volume into a set of volumetric patches of different types, such as cuboids, triangle prisms and triangular pyramids. The user would then be able to specify the number of edge subdivisions along each ridge line of the patches, based on which the system would hexahedralize the interior of each patch. Figure 12 shows a few topological patterns for such volumetric patches. It seems highly challenging to find a complete set of topological patterns that cover all possible configurations.

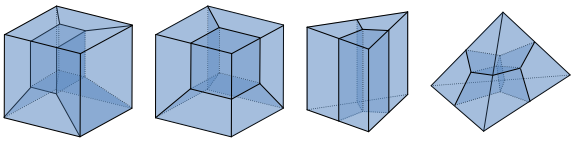


Figure 12: *An incomplete set of topological patterns for volumetric hexahedral mesh generation.*

Acknowledgements

The authors wish to thank Alexander Sorkine-Hornung and Maurizio Nitti for helpful discussion and feedback. The *Bear* model is courtesy of Takeo Igarashi. The *Humanoid* model is courtesy of the AIM@SHAPE Shape Repository. This work was supported in part by the ERC grant iModel (StG-2012-306877). Kenshi Takayama's stay at ETH Zurich until April 2014 was funded by JSPS Postdoctoral Fellowships for Research Abroad.

References

[BCE*13] BOMMES D., CAMPEN M., EBKE H.-C., ALLIEZ P., KOBBELT L.: Integer-grid maps for reliable quad meshing. *ACM Trans. Graph. 32*, 4 (2013). 2

[BDE*10] BERKELAAR M., DIRKS J., EIKLAND K., NOTEBAERT P., EBERT J.: lp_solve, 2010. Version 5.5.2.0, http://lpsolve.sourceforge.net/5.5/. 7

[BLK11] BOMMES D., LEMPFER T., KOBBELT L.: Global structure optimization of quadrilateral meshes. *Comput. Graph. Forum 30*, 2 (2011). 1

[BLP*13] BOMMES D., LEVY B., PIETRONI N., PUPPO E., SILVA C., TARINI M., ZORIN D.: Quad-mesh generation and processing: A survey. *Comput. Graph. Forum 32*, 6 (2013). 1

[BSBK02] BOTSCH M., STEINBERG S., BISCHOFF S., KOBBELT L.: OpenMesh: A generic and efficient polygon mesh data structure. In *OpenSG Symposium 2002* (2002). 7

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3 (2009). 1

[CBK12] CAMPEN M., BOMMES D., KOBBELT L.: Dual loops meshing: quality quad layouts on manifolds. *ACM Trans. Graph. 31*, 4 (2012). 2

[dOMM13] DE OLIVEIRA MIRANDA A., MARTHA L.: Quadrilateral mesh generation using hierarchical templates. In *Proc. IMR* (2013). 2

[ILS*11] II J. D., LIZIER M., SIQUEIRA M., SILVA C., NONATO L.: Template-based quadrilateral meshing. *Computers & Graphics 35*, 3 (2011). 1

[MPKZ10] MYLES A., PIETRONI N., KOVACS D., ZORIN D.: Feature-aligned t-meshes. *ACM Trans. Graph. 29*, 4 (2010). 1

[NSY09] NASRI A., SABIN M., YASSEEN Z.: Filling N-sided regions by quad meshes for subdivision surfaces. *Comput. Graph. Forum 28*, 6 (2009). 2

[PBJW14] PENG C.-H., BARTON M., JIANG C., WONKA P.: Exploring quadrangulations. *ACM Trans. Graph. 33*, 1 (2014). 2, 7, 8

[Pil13] PILGWAY: 3D-Coat, 2013. Version V3, http://3d-coat.com/. 2, 7

[Pix13] PIXOLOGIC: ZBrush, 2013. Version 4.4, http://www.pixologic.com/zbrush/. 2, 7

[PZKW11] PENG C.-H., ZHANG E., KOBAYASHI Y., WONKA P.: Connectivity editing for quadrilateral meshes. *ACM Trans. Graph. 30*, 6 (2011). 4

[SHF13] SCHNEIDER T., HORMANN K., FLOATER M.: Bijective composite mean value mappings. *Comput. Graph. Forum 32*, 5 (2013). 8

[SWZ04] SCHAEFER S., WARREN J., ZORIN D.: Lofting curve networks using subdivision surfaces. In *Proc. SGP* (2004). 2

[TPP*11] TARINI M., PUPPO E., PANOZZO D., PIETRONI N., CIGNONI P.: Simple quad domains for field aligned mesh parametrization. *ACM Trans. Graph. 30*, 6 (2011). 1

[TPSHSH13a] TAKAYAMA K., PANOZZO D., SORKINE-HORNUNG A., SORKINE-HORNUNG O.: *Robust and controllable quadrangulation of triangular and rectangular regions*. Tech. rep., ETH Zurich, 2013. 2, 6

[TPSHSH13b] TAKAYAMA K., PANOZZO D., SORKINE-HORNUNG A., SORKINE-HORNUNG O.: Sketch-based generation and editing of quad meshes. *ACM Trans. Graph. 32*, 4 (2013). 1, 2, 7

[YNB*13] YASSEEN Z., NASRI A., BOUKARAM W., VOLINO P., MAGNENAT-THALMANN N.: Sketch-based garment design with quad meshes. *Computer Aided Design 45*, 2 (2013). 2, 7

[ZHLB13] ZHANG M., HUANG J., LIU X., BAO H.: A divide-and-conquer approach to quad remeshing. *IEEE TVCG 19*, 6 (2013). 2