Eurographics Symposium on Geometry Processing 2021
K. Crane and J. Digne
(Guest Editors)

Volume 40 (2021), Number 5

Simpler Quad Layouts using Relaxed Singularities

M. Lyon[1], M. Campen[2], and L. Kobbelt[1]

[1]RWTH Aachen University, Germany
[2]Osnabrück University, Germany

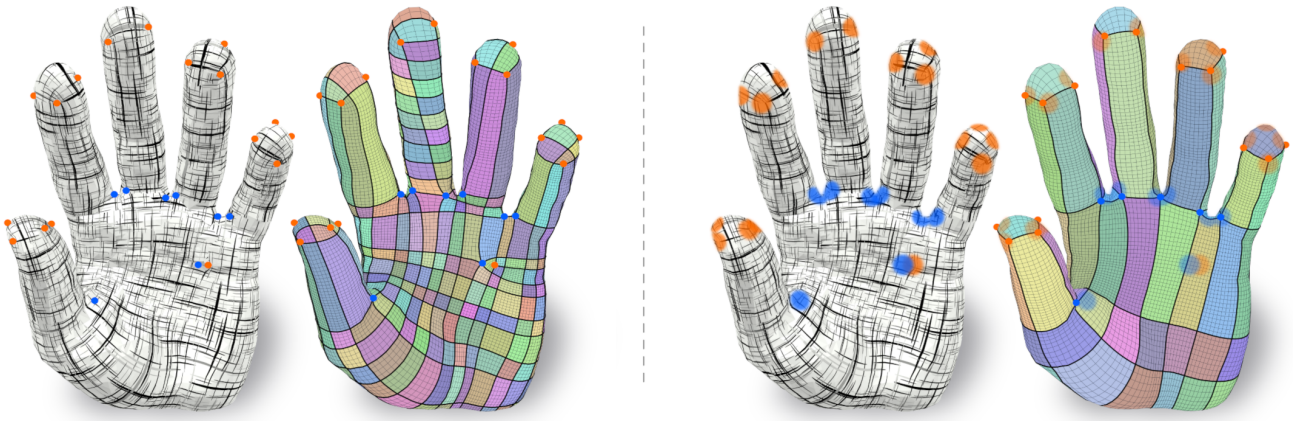


Figure 1: *Quad layouts on surfaces are commonly generated with a two-step strategy: first determine a plausible configuration of exceptional (irregular) layout vertices, then determine a suitable connectivity pattern. Left: Low layout quality due to suboptimal choices in the first step. Right: We consider the result of the first step a suggestion rather than a hard constraint, enabling the controlled relaxation (movement and merging) of the irregular vertex configuration as part of a novel connectivity determination step, yielding simpler layouts of high quality.*

Abstract

A common approach to automatic quad layout generation on surfaces is to, in a first stage, decide on the positioning of irregular layout vertices, followed by finding sensible layout edges connecting these vertices and partitioning the surface into quadrilateral patches in a second stage. While this two-step approach reduces the problem's complexity, this separation also limits the result quality. In the worst case, the set of layout vertices fixed in the first stage without consideration of the second may not even permit a valid quad layout. We propose an algorithm for the creation of quad layouts in which the initial layout vertices can be adjusted in the second stage. Whenever beneficial for layout quality or even validity, these vertices may be moved within a prescribed radius or even be removed. Our algorithm is based on a robust quantization strategy, turning a continuous T-mesh structure into a discrete layout. We show the effectiveness of our algorithm on a variety of inputs.

CCS Concepts

• ***Computing methodologies*** → ***Computer graphics; Mesh models; Mesh geometry models;*** *Shape modeling;*

1. Introduction

In a variety of industries from animation to engineering, quad meshes are the mesh representation of choice for a broad spectrum of tasks [BLP*13]. Ideally, the quad meshes do not only have individual elements of high quality (such as near-right angles) but the quad mesh as a whole has a high quality global structure in terms of patch layout and edge flow.

Algorithmically, the task of generating a quad layout for a given triangle mesh is often separated into two steps [Cam17, §6]. The first step decides on the number, positions, and valencies of the layout vertices. The second step is tasked with connecting these layout vertices by layout edges, so called separatrices, and usually has to find a trade-off between separatrices that are *well-aligned* with intended directions (given, e.g., as a guiding field) and separatrices that are short and form a *simple* layout with a small number of large patches.

Splitting the layout generation problem into two steps reduces complexity, but it also comes with a disadvantage. Since the first

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

step places layout vertices without considering their future interconnectedness, they are unlikely to be in ideal positions with regard to that goal. Not being able to change these vertices, the second step is bound to produce suboptimal results – either because layout edges are less aligned than they could be, or because short connections are avoided due to being deemed too badly aligned. In the worst case it may not even be possible to generate any valid layout with the given layout vertices due to topological restrictions [MPZ14].

Therefore, we propose an algorithm for the generation of simple quad layouts with a relaxed setting in the second step. Concretely, we enable the layout vertices generated in the first step to be moved as well as merged over short distances – whenever this is beneficial to the objective of alignment and simplicity. Our algorithm offers two simple parameters, α and r. The former controls the amount of acceptable deviation of the layout edges from the intended directions, the latter controls the amount of acceptable movement of the layout vertices proposed by the first step.

This relaxed view on the layout vertex configuration opens opportunities to generate better layouts in two ways. For one, vertices may be moved into the acceptable directional deviation range specified by α, enabling shorter layout edges. Secondly, vertices may be merged, possibly even leading to cancellation, i.e., the formation of regular vertices instead.

Similar to [CBK15, LCK21], our layout construction algorithm is based on the technique of quantizing a T-mesh, turning an (easy-to-construct) non-conforming rectangular partitioning of the input surface into a conforming partition (without T-junctions). We setup linear constraints that ensure high layout quality and solve an integer linear program to find a quantization that adheres to these constraints globally. The T-mesh can be generated using existing methods, e.g., by tracing a motorcycle graph in a seamless parametrization or, more directly, in a cross field.

2. Related Work

Quad Layouts Quad layout generation algorithms can be grouped into two categories, methods that rely on the simplification of an initially complex layout and methods that compute layouts for a given surface "from scratch".

Bommes et al. [BLK11] take as input quad meshes and iteratively remove certain helical structures. Similarly, Tarini et al. [TPP*11] use iterative improvements but modify the quad layout more directly by reconnecting individual layout edges. Viertel et al. [VOS19] improve non-conforming layouts by collapsing quad strips in order to reduce the number of patches and T-junctions in the layout. Such iterative methods have in common that layout adjustments are typically applied in a greedy fashion meaning they may get stuck in non-optimal local minima.

A number of algorithms have been proposed for the simplification of quad meshes rather than quad layouts [DSSC08, DSC09, TPC*10, PZKW11]. Conceptually, these are also applicable to quad layouts which could simply be considered very coarse quad meshes. However, these methods only optimize for coarseness and do not consider any other layout quality aspect such as its alignment to prescribed directions.

Razafindrazaka et al. [RRP15] generate layout edges for a given set of layout vertices by first finding a large set of candidates by tracing within a seamless parametrization and then selecting a consistent subset by solving a binary program. Pietroni et al. [PPM*16] propose a similar algorithm, but find candidates directly in a cross field. Their formulation is not always able to find a valid connection for every layout vertex which leads to T-junctions in the final layout and may require the insertion of additional singularities. Instead of explicitly enumerating candidates, Lyon et al. [LCK21] encode possible separatrices via a T-mesh that has been traced in a seamless parametrization. A quad layout is found by solving an integer linear program to find a coarse quantization of the T-mesh while taking care not to generate separatrices that exceed a user provided directional deviation bound. In contrast to our method, these three methods generate layouts that contain (at least) all input singularities which limits the achievable layout simplicity.

Input Complexity Reduction Since layout generation algorithms are often tasked with connecting layout vertices given as input, one way of producing less complex layouts is generating fewer layout vertices in the first place. Commonly, the layout vertices stem from a cross field [VCD*16] which not only defines the layout vertices via its singularities, but also specifies the desired layout edge alignment over the surface. Automatic methods for the generation of such cross fields [BZK09, KCPS13, DVPSH14] commonly use a field smoothness objective, essentially promoting the geodesic straightness of the field's integral curves. The number and positions of singularities that arise is mainly mandated by the surface's Gaussian curvature distribution. Ray et al. [RVAL09] describe an approach based on the smoothing of the Gaussian curvature field beforehand, leading to a smaller number of singularities. In a different approach, Ebke et al. [ECBK14] compute cross fields based on a smoothed normal field. Subsequently, the crosses are projected back onto the original surface, which, unfortunately, may flip the cross. Surface parametrizations that adhere to these directions will contain flipped or degenerate triangles which is problematic for quad mesh [CBK15, LCBK19] and quad layout generation algorithms [RP17, LCK21] that require flip free parametrizations as input.

Motorcycle Graph Tracing In the context of quad meshing, motorcycle graphs [EGKT08] are used to create rectangular partitions of surfaces. In the simplest case such a partition is created by discretely following edges of a given quad mesh [RP17]. Here, tracing is trivial as one just needs to follow the edges of the quad mesh and intersections between traces are easily identifiable. However, suitable quad meshes are relatively difficult to obtain.

Alternatively, motorcycles can be traced in a seamless parametrization [CBK15, LCK21] which is still relatively straight forward if care has been taken to properly sanitize the parametrization [MC19].

Finally, tracing motorcycles within a cross field is attractive as the input is relatively easy to obtain. However, tracing then requires the most sophisticated algorithms [BJB*11, MPZ14, RS14] as well as additional considerations when working with the resulting graph in the downstream application (cf. Section 4.4).

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

3. Quad Layout via T-mesh Quantization

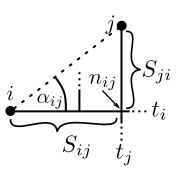
In our algorithm we adopt the setting of Lyon et al. [LCK21] and construct a quad layout via the quantization of a T-mesh, i.e., the constrained assignment of non-negative integer values to elements of a non-conforming surface partition in a globally consistent manner. In this section we concisely recap this setting and discuss how the quantization affects the final quad layout. For an in-depth explanation of the background we refer to the above paper.

3.1. T-mesh

A T-mesh $\mathcal{T} = (\mathcal{N}, \mathcal{A}, \mathcal{P})$ consists of nodes $\mathcal{N}$, arcs $\mathcal{A}$, and patches $\mathcal{P}$. In our context, T-meshes are, for instance, generated by tracing a so called motorcycle graph, either in a cross field [RS14, MPZ14] or in a seamless parametrization [CBK15, LCBK19,LCK21]. Conceptually, particles (called motorcycles) are spawned from every singularity (of the field or the parametrization) into all field or isoline directions; they stop when they intersect the trace of another motorcycle (or their own). When the last motorcycle stops, the union of traces form a T-mesh whose nodes consist of the singularities and intersection points, and whose arcs are formed by the trace segments between nodes. In the case of tracing in a cross field, small additional measures can be necessary to ensure unconditional termination [MPZ14]. It can be shown that all patches (bounded by T-mesh arcs) are logically rectangular [EGKT08].

In slight deviation from this, we adopt the setting of Lyon et al. [LCK21]: motorcycles do not stop at the first intersection, but continue through further intersections until a certain condition is met (cf. Section 3.3). This yields an extended motorcycle graph, containing more intersection nodes, exploited in the following to set up quantization constraints targeting layout quality. In Figure 6b we show one such T-mesh. Note that while the T-mesh is depicted on the surface in 3D, the only relevant information required for our method is its topology as well as a length l_i for every arc $a_i \in \mathcal{A}$. If the T-mesh is traced in a parametrization, l_i is simply the length of the arc in the parametric domain. For a T-mesh traced in a (possibly non-uniform) cross field, specified by vectors v_1, v_2 per face, the length l_i is measured piecewise w.r.t. the local per-face coordinate systems formed by the cross field vectors, i.e., using the metric tensor $v_1 v_1^T + v_2 v_2^T$.

At each intersection point the two intersecting traces suggest a candidate separatrix connecting the two singularities that spawned the traces. Geometrically, this separatrix can be depicted as the hypotenuse of a right triangle formed by the two traces. To make this precise: Traces t_i and t_j starting in singularities i and j intersect in node n_{ij}. With $S_{ij} \subset \mathcal{A}$ we denote the set of arcs between i and n_{ij}. With the combined length l_{ij} of arcs S_{ij}, the angular deviation α_{ij} between trace t_i and the potential separatrix connecting i and j, can be computed as $\tan\alpha_{ij} = l_{ji}/l_{ij}$. Note that this angle is only an estimate of the actual pointwise angular deviation in case the T-mesh was traced in a locally non-integrable cross field. For the smooth cross fields commonly used in practice, however, this can be expected not to be an issue of relevance.



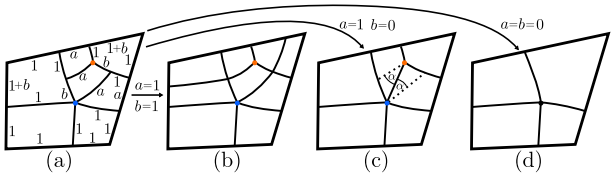


Figure 2: *Effect of different quantizations for T-mesh in (a). If both a and b are set to 1, the two inner singularities are not connected (b). If b (or a) is set to 0, the corresponding arcs can be collapsed and the two singularities are connected by a separatrix that deviates α from the original trace directions (c). If a and b are both set to 0, the singularities are merged forming a regular inner vertex (d).*

3.2. Quantization

A quantization is an assignment of non-negative integer lengths to the arcs of a T-mesh $\mathcal{T}$. We denote with $q_i \in \mathbb{Z}^{\geq 0}$ the quantization of arc $a_i \in \mathcal{A}$ and use $q_{ij} := \sum_{a_k \in S_{ij}} q_k$ for the total quantization of the arcs in S_{ij}. A quantization is *consistent* if the patches of $\mathcal{T}$ remain rectangular, i.e., if the total quantization of the arcs on opposite sides of the patches is equal.

The purpose of these assigned integer values is to define how the non-conforming T-mesh shall be turned into a conforming quad layout (without T-joints), see Figure 2 for an illustration. Effectively, the quantization specifies (A) how arcs are to be conceptually extended across T-joints, eventually connecting them with opposite arcs in a regular manner, and (B) which arcs may be and are to be collapsed ($q_i = 0$) to yield a simple layout. In particular, it also determines which separatrices are formed, i.e., which of the above mentioned candidate separatrices actually materialize: If all arcs forming one leg (S_{ij} or S_{ji}) of the right triangle are quantized to 0, singularities i and j end up on the same layout edge or edge sequence, as in the $b = 0$ case in Figure 2(c).

3.3. Integer Linear Program

The following constraints have been employed to restrict to quantizations that actually yield valid and desirable layouts:

1. *Consistency constraints*: For a patch of $\mathcal{T}$, the total quantization of the arcs on opposite sides of the patch is required to be equal. This ensures that the patch remains rectangular [CBK15].
2. *Separation constraints*: Given two traces t_i and t_j that intersect at n_{ij} with distances $l_{ij} > l_{ji}$, requiring $q_{ij} \geq 1$ prevents singularities i, j from being forced to collapse into each other [LCK21].
3. *Layout constraints*: If in addition α_{ij} exceeds a given direction deviation bound, also $q_{ji} \geq 1$ is required, preventing the separatrix from i to j from materializing [LCK21].

As an example, separation constraints prevent Figure 2(d), leaving (b) and (c) as options. Layout constraints may further rule out (c), leaving only option (b).

Given a T-mesh (cf. Section 3.1) the above constraints are constructed for every patch and every intersection node n_{ij}. The purpose of using the *extended* motorcycle graph, as mentioned above, is to have the intersection n_{ij} explicitly present in the T-mesh for each pair i, j of singularities whose separatrix connection shall potentially be prevented by a layout constraint.

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

A quantization can then be computed by means of an integer linear program (ILP) whose variables are the arc quantization values q_i. To yield a simple layout, the employed objective is the minimization of the sum of all q_i, possibly weighted [LCK21]:

$$E = \sum_{a_i \in \mathcal{A}} l_i^{\perp} \cdot q_i, \tag{1}$$

where $l_i^{\perp}$ denotes half the length of the patches incident to arc a_i.

4. Relaxed Quantization

For the purpose of generating simpler quad layouts we propose a relaxed formulation for the quantization of a T-mesh. It differs from the above in three key ways:

- The positions of the given irregular layout vertices are not considered to be fixed. Instead, we enable them to move by some distance in case this is of benefit for the layout's connectivity.
- We enable nearby irregular vertices to merge in case this is of benefit. This includes the possibility of mutual cancellation of two or more irregular vertices.
- We generalize the formulation to support T-meshes traced directly in a cross field rather than a seamless parametrization.

The latter point is non-trivial because the existence of limit cycles in this more general setting may imply infeasibility of the above consistency constraints [MPZ14]. Our relaxed formulation gracefully handles such cases, effectively by adjusting the input irregular vertices as necessary to facilitate a valid layout. In contrast to other methods which insert additional singularities in such cases [MPZ14, PPM*16] our method rather simplifies the layout by merging existing singularities.

Parameters. Our algorithm is controlled via two parameters:

- Angular parameter α controls the acceptable amount of deviation of the layout separatrices from the intended directions, as given by a cross field or parametrization isolines.
- Distance parameter r controls the acceptable amount of movement of irregular layout vertices from their original position.

Both are soft parameters in our method, i.e., they do not define hard bounds, but still enable the overall trade-off of priorities (cf. Section 6). This is a sensible approach, given that it is common to follow up with a final global geometric optimization of the resulting quad layout's or quad mesh's embedding in the surface [LCBK19, CK14, TPP*11], which will adjust positions and alignment wherever beneficial for global quality objectives, cf. Figure 3.

Relaxed Constraints. We employ an integer linear program to compute a quantization, but use a different set of constraints than previous work. Given a T-mesh (cf. Section 3.1), we set up *relaxed separation constraints*; they prevent collapses of pairs of layout vertices only if they are out of reach, considering their allowed movement by r (Section 4.1). Additional index constraints are required on top, because for reasons of topological validity, depending on their degrees of irregularity, not all pairs (or larger sets) of irregular vertices form groups that may be merged (Section 4.2). Using *relaxed layout constraints*, we prevent badly aligned separatrices (controlled by parameter α), again taking the vertices' allowed movement by distance r into account (Section 4.3). Finally, a modification of the above constraints addresses the problem of infeasibility due to limit cycles, which may be present if the T-mesh was traced in a generic cross field rather than a parametrization (Section 4.4).

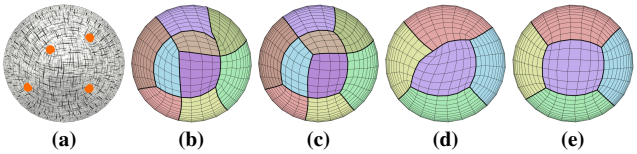


Figure 3: *Badly aligned singularities (a) may lead to more complex layout connectivity to avoid distorted elements (b). A simple post-process moves vertices to improve element quality (c). Our algorithm anticipates small movements and allows short, badly aligned layout edges (d) whose alignment is improved during the smoothing post-process (e).*

The generation of quad meshes, exhibiting the desired layout structure, from the quantized T-mesh is explained in Section 5.

4.1. Relaxed Separation Constraints

Similar to [LCK21] we want to keep irregular layout vertices separate – unless their distance is smaller than $2r$, because for pairs of layout vertices closer than $2r$ we intend that they may move towards each other, each by r, and may be merged if beneficial. Thus, the separation constraints are only added for a pair of singularities i, j with a trace intersection n_{ij} if their distance exceeds $2r$. The separation is enforced in the direction of larger distance.

$$q_{ij} \geq 1 \quad \text{if} \quad l_{ij} > l_{ji} \text{ and } l_{ij} > 2r \tag{2}$$

Note that this means that the movement (limited by r) is effectively measured in the L^{∞}-norm with respect to the cross field or the seamless parametrization that the T-mesh is aligned to. In other words, r controls how far a singularity may move per (parametric or field) direction. This simplifies the condition under which the above constraints are added, relative to the use of the L^2-norm. Note also that by measuring the distance with respect to the cross field or seamless parametrization it naturally adapts to situations where the field or parametrization employs local (anisotropic) sizing to better capture the surface geometry.

4.2. Index Constraints

Each irregular layout vertex has a prescribed valence. When these vertices are derived from the singularities of a cross field, this valence v_i of a layout vertex i is related to the corresponding singularity's index I_i [RVLL08] by $v_i = 4(1 - I_i)$. When merging a set of singularities, the resulting singularity will have an index that is the *sum* of the individual indices. Analogously, merging a set of irregular layout vertices $\{k\}$, the resulting vertex i will have a valence

$$v_i = 4 - \sum_k (4 - v_k).$$

As layout vertex valences below or above a certain threshold may be undesirable or even unacceptable depending on the target use case, with the goal of preventing them we add index constraints.

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

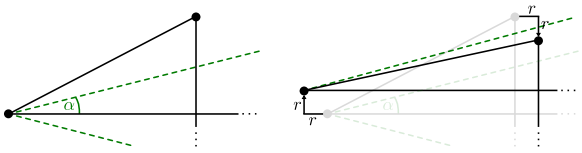


Figure 4: *Left: with fixed vertices some separatrices cannot be used in the layout due to exceeding the maximum deviation. Right: if vertices can move by r in each direction the deviation may be reduced into the acceptable range.*

Let N_i be the set of layout vertices reachable from vertex i with a distance of at most $2r$. For every such neighboring vertex $j \in N_i$ we add a binary indicator variable $c_{ij} \in \mathbb{B}$ that will take the value 1 if vertices i and j collapse. This is achieved as follows: Let $P_{ij} \subset \mathcal{A}$ be the shortest arc path from i to j. If all arcs $a_k \in P_{ij}$ are quantized to zero, vertices i and j collapse onto each other. We therefore set up the following linear constraint that ensures the indicator c_{ij} is 1 if and only if the complete path P_{ij} is quantized to zero:

$$1 - \sum_{a_j \in P_{ij}} q_j \leq Q c_{ij} \leq Q - \sum_{a_j \in P_{ij}} q_j \tag{3}$$

where Q is a large number. The actual value of Q is not important as long as it is ensured that the right term never becomes negative. Due to the objective in Equation (1), quantization values q_j tend to be 0 unless otherwise constrained. Separation and layout constraints will only enforce values of 1. Consistency constraints, however, may enforce larger values if the quantization of a single arc needs to be equal to the sum of quantizations of multiple arcs on the opposite side of a patch. Thus, a single q_j is bounded by the number of arcs $|\mathcal{A}|$ and we can conservatively set $Q = |\mathcal{A}| \cdot \max_{i,j \in \mathcal{N}}(|P_{ij}|)$.

The resulting index I_i^* of vertex i can then be estimated as

$$I_i^* = I_i + \sum_{j \in N_i} c_{ij} I_j \tag{4}$$

and constrained to be within the user specified desirable range $[I_{\min}, I_{\max}]$ with two simple linear constraints:

$$I_{\min} \leq I_i^* \leq I_{\max}. \tag{5}$$

Note that it is conceptually possible for two vertices to collapse not along the shortest arc-path P_{ij} but another one – which would be overlooked by the above indicator construction. Due to the effect of the consistency constraints (Section 3.3), however, this is only possible for non-shortest paths in a different path homotopy class (i.e., winding around handles or further singularities). This could be covered by the addition of further constraints, along shortest arc paths per homotopy class up to the movement distance limit. In our experiments we have not encountered a case where this would have been relevant though.

4.3. Relaxed Layout Constraints

Layout constraints ensure that the resulting quantization does not induce separatrices that deviate too much, i.e., more than α, from their intended direction. With a strict preservation of input vertex positions ($r = 0$), this means that for a pair of vertices i, j with intersection n_{ij} with $l_{ij} > l_{ji}$ (which are already separated in one direction due to separation constraints, cf. Section 4.1) additional separation in the orthogonal direction is to be enforced via a constraint $q_{ji} \geq 1$ if $l_{ji}/l_{ij} > \tan\alpha$. This way layout vertices i and j cannot end up on the same line in the parametric domain defined by the quantization, hence no separatrix is formed.

If layout vertex positions are assumed to be flexible within a distance of r, the directional deviation of a potential separatrix between two vertices can be reduced. Figure 4 illustrates the maximum reduction that can be achieved, by conceptually moving both vertices in an opposite manner in both parametric directions. This reduces the length of one leg of the right triangle by $2r$ while increasing the other by the same amount. This leads to the addition of a layout constraint under an accordingly relaxed condition:

$$q_{ji} \geq 1 \quad \text{if} \quad \frac{l_{ji} - 2r}{l_{ij} + 2r} > \tan\alpha. \tag{6}$$

4.4. Infeasibility

Due to the fact that cross fields are typically not integrable they may contain limit cycles. Tracing a motorcycle graph in such a cross field may thus lead to T-meshes containing arcs for which no positive length can be assigned without violating consistency [MPZ14]. Two examples where this happens are shown in Figure 5. In the sea shell model on the left the two arcs marked with q_1 must be quantized to zero because for the inner most patch to be rectangular it must hold that $q_1 + q_2 = q_2$ which can only be fulfilled with $q_1 = 0$. A similar problem occurs in the cube model on the right, with a valence 3-5 pair of singularities on one of its sides and a partially specified fixed layout along the 12 cube edges. Due to the singularities at least one of the sides of the top face needs to have a

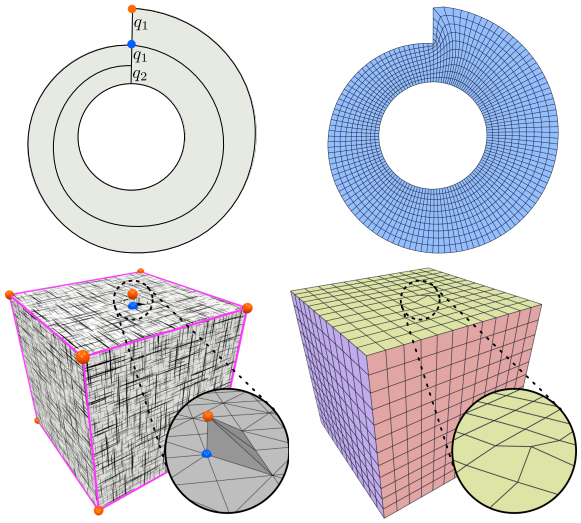


Figure 5: *Two examples of input singularities that do not permit a valid quad mesh. Top, an annulus with a valence 2 and a valence 4 singularity on its border. Bottom, a cube with partially fixed layout along the 12 edges and a valence 3-5 pair on one of its sides. In both cases our formulation merges singularities as necessary, turning them into regular vertices.*

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

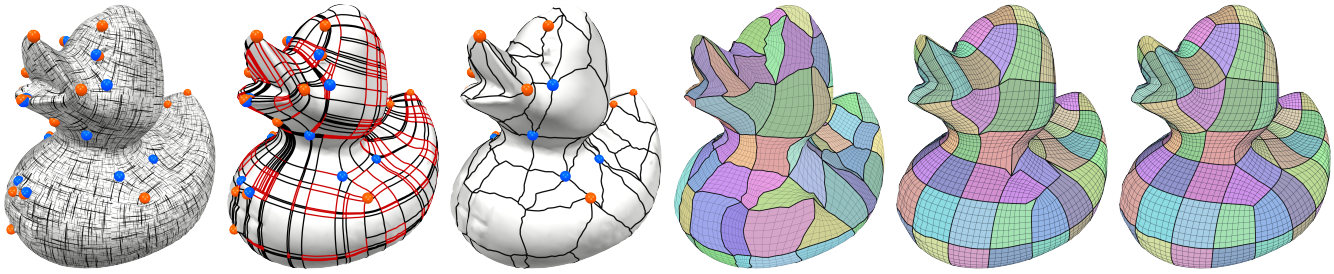


Figure 6: *Overview of our pipeline. (a) Cross field. (b) Traced T-mesh. Arcs marked in red are quantized to zero and will be collapsed. (c) Collapsed T-mesh. (d) Initial patch-wise parametrization. (e) Quad mesh extracted from optimized parametrization. Note how the parametrization optimization cannot relocate singularities. (f) Geometric relaxation moves singularities to better locations.*

different quantization value (thus number of quads) than its opposite side. This, however, is not possible because all other faces of the cube need to have equal numbers of quads on opposing sides.

The presence of such arcs which must be quantized to zero can make our constraints infeasible if two layout vertices exist that are connected via arcs which are all forced to be zero by consistency constraints, yet the vertices lie so far apart that they are forced to be separated by separation constraints.

To enable the handling of such cases – which may only occur if the T-mesh stems from a generic cross field rather than a seamless parametrization – instead of using the hard constraints discussed in the previous sections we can effectively soften them by modifying separation and layout constraints of the form $q_{ij} \geq 1$ to

$$q_{ij} + h_{ij} \geq 1, \tag{7}$$

where h_{ij} is a binary slack variable which is heavily penalized. This enables the merging even of vertices that are more distant than $2r$, but only if this is inevitable. To this end we choose the penalty so high that increasing any value h_{ij} above 0 is as costly as setting all values q_i to 1 in Equation (1).

Similarly, Equation (5) can be implemented in a soft way:

$$I_{\min} - h_i \leq I_i^* \leq I_{\max} + h_i \tag{8}$$

where h_i again is a highly penalized integer variable.

Finally, we augment the objective in Equation (1) with the penalty terms:

$$E = \sum_{a_i \in \mathcal{A}} l_i^{\perp} \cdot q_i \quad + \left(\sum h_i + \sum h_{ij}\right) \cdot \sum_{a_i \in \mathcal{A}} l_i^{\perp} \tag{9}$$

Note that if a pair of singularities exists that cannot be separated even though it should be, there are two ways of dealing with the issue. Additional irregular vertices can be inserted so as to enable the separation [MPZ14]. This increases the layout's complexity. We are interested in simple layouts and with the above strategy we allow the merging of vertices even though this requires a singularity movement larger than desired. For the two examples in Figure 5 our algorithm simplifies the layout by removing singularities. Which of the two strategies is beneficial ultimately is a question of the specific application scenario.

4.5. Relaxed Integer Linear Program

In summary, we find a quantization by solving the following ILP:

$$\begin{array}{ll} \text{minimize} & \text{Energy } \mathbf{(9)} \\ \text{subject to} & \mathbf{(2)}^*, \mathbf{(3)}, \mathbf{(6)}^*, \mathbf{(8)} \end{array} \tag{10}$$

where Equations (2) and (6) are implemented in a soft manner according to Equation (7).

As in [LCK21], not every single arc needs its own quantization variable; they can be shared within sets of arcs that trivially need to have identical quantization values due to being solitary arcs on opposite sides of simple patches. Also the number of (relaxed) separation and layout constraints can be reduced since many of the sets of arcs required to be quantized to ≥ 1 are supersets of another one and therefore are implied. Hence, the number of quantization variables, separation and layout constraints are bounded by the number of traces, rather than the number of arcs or patches, which depends on the number of input singularities but is independent of the choice of α and r [LCK21].

The possibility to collapse layout vertices increases the complexity of the integer linear program. While there are only two index constraints (Equation (8)) per layout vertex, the required collapse indicator variables c_{ij} and corresponding constraints (Equation (3)) depend on r: for each layout vertex i one indicator variable and two constraints are added for each other layout vertex j which i may collapse into, i.e., with $l_{ij} \leq 2r$ (cf. Equation (6)). In the worst case, for extreme r, every vertex can collapse into any other one. In this case the number of additional variables and constraints would be quadratic in the number of layout vertices.

5. Quad Mesh Extraction

Lyon et al. [LCBK19] propose a robust algorithm for the quad mesh generation, using Tutte-embeddings of the T-mesh's individual patches. This initial parametrization can then be optimized globally while taking care not to introduce any invalid elements [RPPSH17]. This strategy works well due to the theoretical guarantees of the Tutte-embedding. However, for the Tutte-embedding itself to yield valid results the prescribed boundary needs to be non-degenerate. Therefore, patches that are quantized to zero area are removed by iteratively collapsing arcs [MPZ14, LCBK19].

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

Our method builds on the observation that, in contrast to previous methods that use a quantized T-mesh to prescribe positional constraints on all original singularities, with this collapse based approach strict singularity separation is not technically required. As Figure 2 illustrates, singularities can be collapsed onto each other in the T-mesh, forming a single node. We can therefore use the zero-arc collapse algorithm as suggested in [LCBK19] with one added rule: when an arc is collapsed we always collapse towards the singularity if one is present. This way, singularities remain on their original position unless two or more are merged in which case the collapse may be performed in either direction. Note that when merging two singularities the position of the remaining T-mesh node is not important since the collapse distance is small (below $2r$) and the final positions will be determined by a geometric optimization step in the end.

To globally improve the initial patch-wise Tutte-embedding we optimize the alignment-promoting energy proposed in [BZK09], for which we compute a new cross field that fits the new singularity configuration. We use [EBCK13] to extract a quad mesh from the resulting parametrization.

Note that so far all singularities, apart from those that were merged, still remained on their original position as the parametrization optimization is unable to move them. This means that some separatrices may have relatively bad alignment as the movement of up to r assumed during solving of the integer linear program has not occurred yet. Thus, as a final step, we optimize the layout embedding to move singularities to better locations on the surface. For this purpose it would be ideal to use an algorithm which is explicitly designed to optimize layout embeddings such as [CK14], however, we found that in practice also a simple Laplacian smoothing of the quad mesh, with vertices constrained to remain on the input surface, works well. Our complete pipeline is summarized in Figure 6.

6. Results

As input for our method we created cross fields and seamless parametrizations with [BZK09] using a target edge length of 1% of the bounding box diagonal. We use Gurobi [GO21] to solve our integer linear program.

We show the results of our method on a variety of inputs in Figure 7 and sumarize the statistical data in Table 1. We used parameters $\alpha = 15°$, $r = 2.0$, $I_{\min} = -4$ (valence 8), and $I_{\min} = 1$ (valence 3) by default if not stated otherwise.

Model	#Faces	$\#S_{in}$	$\#S_{out}$	#P	MSJ_{avg}	time
ARMADILLO	43160	185	145	391	0.96	2.4 s
WRENCH	50000	24	16	20	0.98	0.12 s
DANCER	18292	88	49	226	0.94	1.0 s
LION	100002	274	240	805	0.97	10.8 s
DANCER2	49996	82	62	207	0.97	0.5 s
SANTA	151558	80	74	228	0.98	0.8 s
ROLLSTAGE	100000	50	48	77	0.99	0.3 s
ELEPHANT	49918	132	97	246	0.98	1.7 s

Table 1: *Statistical data for our results. From left to right: Model name, number of triangles, singularities in the input and output, number of patches in the resulting layout, average minimum scaled Jacobian of quads, time to compute quantization.*



Figure 7: *Our method's result layouts and meshes on various models with cross field derived input singularities.*

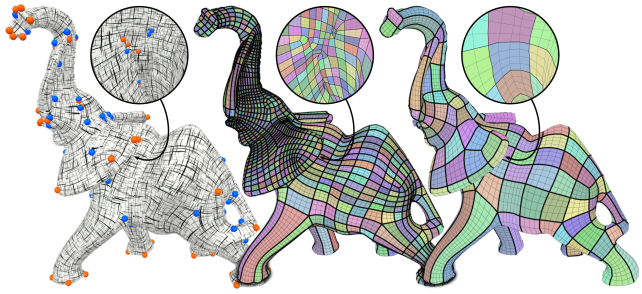


Figure 8: *Left: input cross field. Inset shows noisy singularities behind the ear. Middle: keeping all singularities leads to dense layout. Right: allowing singularities to move small distances and merge with others yields a simpler layout.*

Figure 8 shows the benefit of singularity merges. When singularities are not allowed to move they are reproduced in the final layout which is necessarily relatively dense. Allowing nearby singularities to collapse removes many pairs of valence 3-5 singularities, especially those behind the ear created by noise, but also on the trunk or around the eye region, leading to a much simpler layout.

In Figure 9 we show the effect of the two parameters that control

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

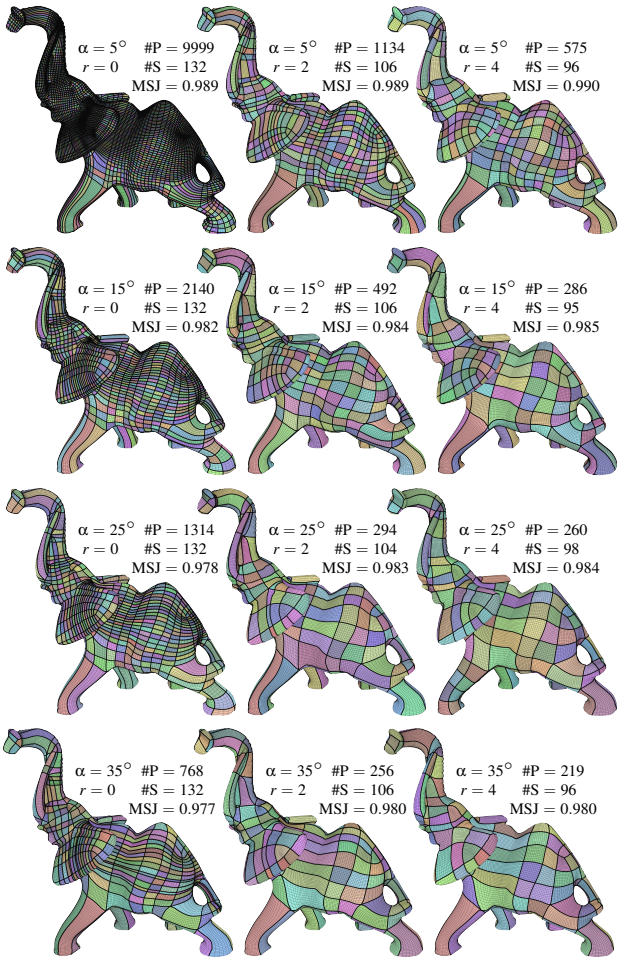


Figure 9: *Effect of the parameters on the layout. From left to right r is 0, 2, and 4; from top to bottom* α *is 5°, 15°, 25° and 35°.*

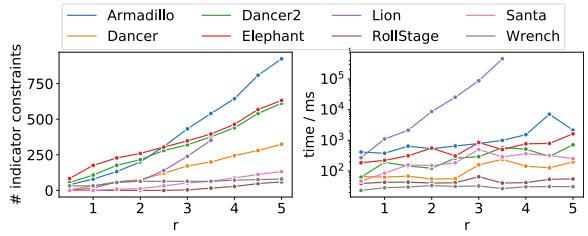


Figure 10: *For the models in Figure 7 we show how increasing r influences the number of indicator constraints (Equation (3), used for the index constraints) and time for solving the ILP.*

our algorithm. We vary α, the parameter that controls the maximal separatrix deviation, between 5° and 35°, and r between 0 and 4. Increasing either parameter produces coarser layouts. Note that increasing r allows to simplify the layout in two ways: the relaxed constraints (Equation (6)) permit separatrices which would be outside the angular bound otherwise, and by merging nearby singularities fewer vertices necessarily remain in the layout, requiring fewer separatrices and leading to layouts with fewer patches.

To evaluate the effect of r on the number of layout vertices another layout vertex i can collapse into, and thus on the number of constraints and the run-time of the solver, we compute quantizations for the models in Figure 7 for varying r. The graphs in Figure 10 show that an increased number of constraints generally lead to a longer run-time. The exact correlation is hard to predict as the used general purpose solver package employs a variety of opaque heuristics and strategies to solve the NP-hard problem.

6.1. Comparison

We compare our method with the methods of Pietroni et al. [PPM*16] and Lyon et al. [LCK21] on models also used in these works and show the results in Figures 11 and 12. For the models from [PPM*16] (Figure 11 (top) and Figure 12) we use the frame-fields provided by the authors as input to our method. For the others we generate cross fields using [BZK09]. For [PPM*16] we compare to their published results, for [LCK21] we generate results using our method with the setting $r = 0$.

Pietroni et al. produce quad layouts by searching separatrix candidates in a cross field and solving a binary program to select a subset which forms the final layout. Their method may choose to not connect some of the layout vertices leading to separatrices that end in T-junctions. While this is not necessary for the simpler models in Figure 11, for the more complex models in Figure 12 up to 450 T-junctions are inserted. Quantizing these layouts, to turn them into conforming quad meshes without T-junctions, requires either inserting additional singularities, or extending arcs beyond T-junctions until another singularity is hit. This may lead to very dense layouts, with 42k, 55k, 284k, 40k, and 124k patches for the models DRAGON, GARGOYLE, REDBOX, NEPTUNE, and LUCY, respectively (see Figure 12 center column).

Lyon et al., similar to the proposed method, solve a linear integer program to compute a quantization of a T-mesh. They strictly preserve all singularities of the input without adding any new ones and generate conforming layouts without any T-junctions. For the simpler models in Figure 11 this method achieves coarse layouts of high quality. But their strict adherence to the original layout vertex positions leads to layouts that are denser than the non-conforming ones of [PPM*16] for the more complex models in Figure 12. Coarser layouts can be achieved by increasing α at the cost of more directional deviation and quad elements with higher distortion.

On the simpler models, shown in Figure 11, our relaxed approach is able to outperform both previous methods in terms of layout simplicity while at the same time improving or preserving the geometric quality of quad elements, measured by the minimum scaled Jacobian. Only the parameter α was varied for these experiments; r was kept at 2.0 in all 12 cases. However, notice that since singularities are relatively sparse in these simpler models, the layout complexity reduction is moderate. In contrast, on the more complex models in Figure 12 we achieve numbers of patches that are only fractions of those of [LCK21] and are lower than even the non-conforming ones of [PPM*16]. Element quality, see Table 2, lies in the same range, with MSJ_{avg} commonly between these previous methods, and MSJ_{min} not rarely better than both.

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

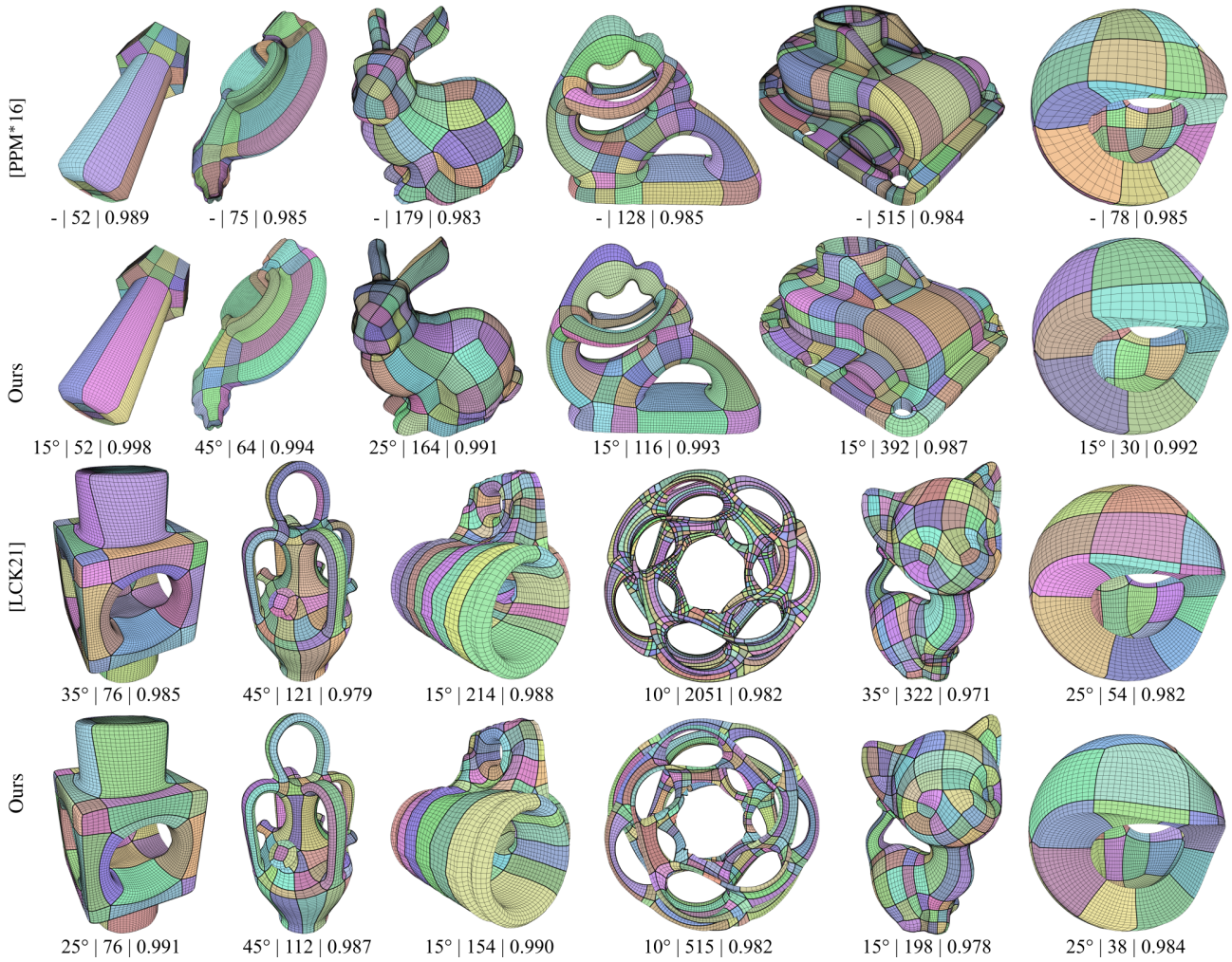


Figure 11: *Comparison on models of Figure 3 in [PPM*16] and Figure 5 in [LCK21]. Listed numbers are* α | *number of patches* | MSJ_{avg}.

7. Limitations & Future Work

We presented a method for the generation of coarse quad layouts which utilizes layout vertex movements to achieve better results, either by moving them to improve the layout edge alignment, or by completely merging them to reduce the total number of layout vertices. Our experiments show that the resulting layouts are of high quality and coarser than the results of other state-of-the-art layout generation algorithms.

Typically, our algorithm takes roughly twice the amount of time than the similar algorithm of [LCK21] that strictly preserves all vertices. However, on some models with high singularity density solving our linear integer program may take an unexpectedly long time (cf. LION in Figure 10 and REDBOX in Table 2). This is probably due to the index constraints which for each singularity adds a constraint for each other vertex reachable within r. Here it could be interesting to investigate whether a specialized solver can reduce the expected run-time. Another option could be to solve the problem iteratively and to add constraints lazily only for those singularities that collapsed into a single vertex with undesired singularity index.

Currently, our formulation considers vertex movement only locally for each pair of vertices which could potentially be connected by a layout edge and optimistically assumes that the vertices can actually be moved the full distance in case the layout edge is formed. If for two or more layout edges a movement is assumed in conflicting directions the resulting separatrices may deviate more than expected from the intended directions as the vertex could not be moved in both directions. To remedy this it would be interesting to include the actual vertex movement into the linear program formulation to prevent conflicting vertex movement assumptions.

Acknowledgements

The authors thank Jan Möbius for creating and maintaining the geometry processing framework OpenFlipper [MK12]. Models were provided by [MPZ14, PPM*16]. This work was supported by the Gottfried-Wilhelm-Leibniz Programme of the Deutsche Forschungsgemeinschaft (DFG) and in part funded by the Deutsche Forschungsgemeinschaft (DFG) - 427469366.

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

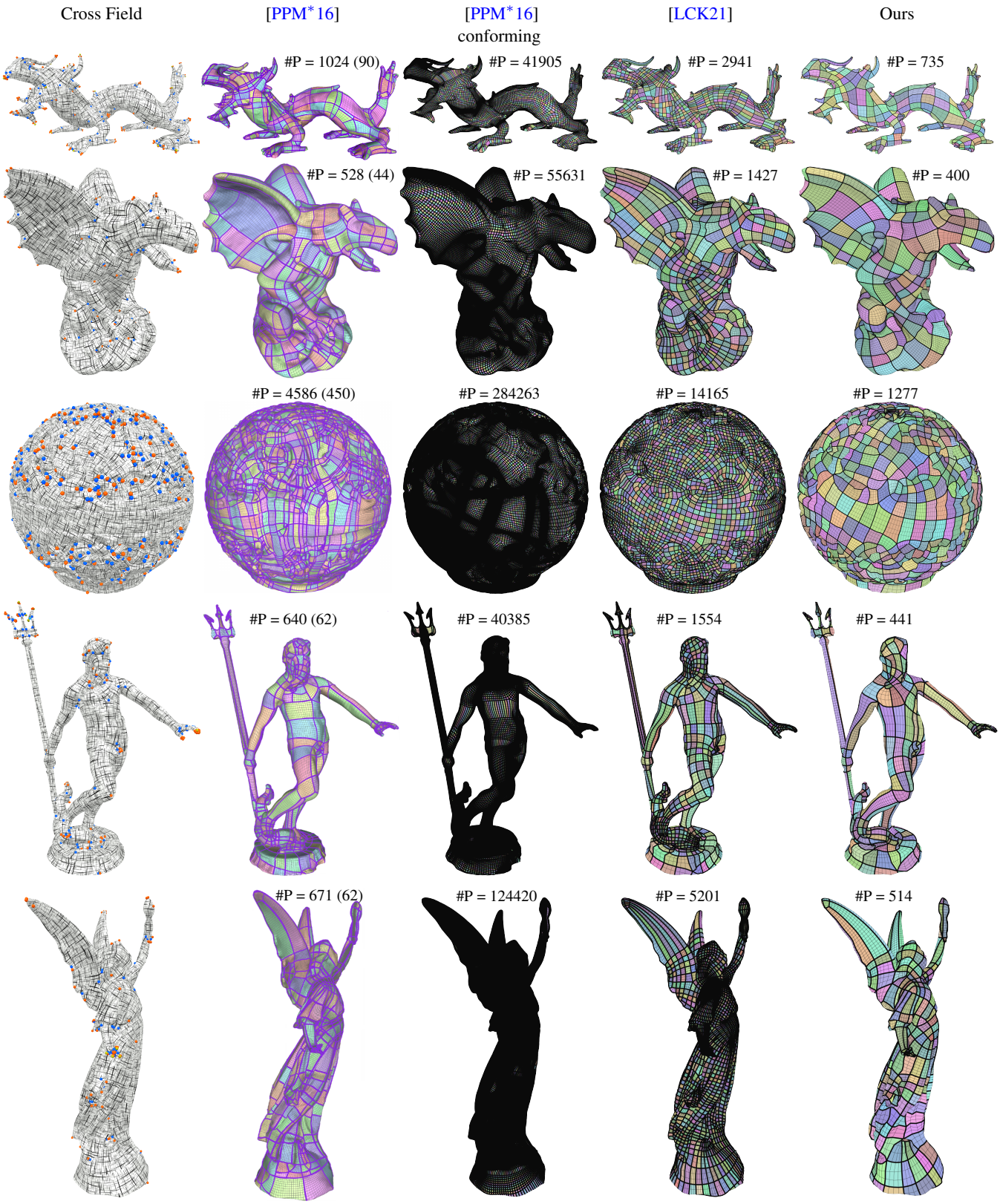


Figure 12: *Comparison with [PPM*16] and [LCK21]. The number indicates the number of patches in the layout. For [PPM*16] the number in brackets is the number of T-junctions. For the conforming version of [PPM*16] we show the complete base complex (without any T-junctions) of the quad meshes published with the corresponding paper. [LCK21] and ours do not produce any T-junctions. Also see Table 2.*

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.

Model	Method	α	r	#P	#T	$\#S_{out}$	MSJ_{min}	MSJ_{avg}	time
DRAGON	[PPM*16]			1024	90	450	0.015	0.928	64.0 s
	[LCK21]	25°		2941	0	343	0.023	0.977	2.1 s
	ours	25°	1.0	735	0	276	0.164	0.959	4.5 s
GARGOYLE	[PPM*16]			528	44	178	0.118	0.973	18.2 s
	[LCK21]	25°		1427	0	141	0.100	0.981	0.5 s
	ours	25°	2.0	400	0	106	0.223	0.978	1.5 s
REDBOX	[PPM*16]			4586	450	1214	0.007	0.961	209.0 s
	[LCK21]	25°		14165	0	1170	0.317	0.981	4.3 s
	ours	25°	2.0	1277	0	422	0.292	0.966	>1000.0 s
NEPTUNE	[PPM*16]			640	62	344	0.069	0.958	32.9 s
	[LCK21]	25°		1554	0	226	0.202	0.977	1.1 s
	ours	25°	1.0	441	0	167	0.403	0.963	2.2 s
LUCY	[PPM*16]			671	62	243	0.046	0.965	101.0 s
	[LCK21]	15°		5201	0	173	0.219	0.986	1.4 s
	ours	15°	1.0	514	0	115	0.251	0.969	3.0 s

Table 2: *Statistics accompanying Figure 12. #P, #T, $\#S_{out}$, MSJ_{min}, MSJ_{avg} are number of layout patches, T-junctions, resulting singularities, minimum and average minimal scaled Jacobian of the quads, respectively. For* REDBOX *we stopped the ILP solver after 1000 s and show the best solution it had found so far (with a remaining optimality gap of 10%) – a first solution (1404 patches, 16% gap) was found after 325s.*

References

[BJB*11] BHATIA H., JADHAV S., BREMER P.-T., CHEN G., LEVINE J. A., NONATO L. G., PASCUCCI V.: Edge maps: Representing flow with bounded error. In *Proc. IEEE Pacific Visualization Symposium* (2011), p. 75–82. 2

[BLK11] BOMMES D., LEMPFER T., KOBBELT L.: Global structure optimization of quadrilateral meshes. *Comp. Graph. Forum 30*, 2 (2011), 375–384. 2

[BLP*13] BOMMES D., LÉVY B., PIETRONI N., PUPPO E., SILVA C., TARINI M., ZORIN D.: Quad-mesh generation and processing: A survey. *Comp. Graph. Forum 32*, 6 (2013), 51–76. 1

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3 (2009), 77:1–77:10. 2, 7, 8

[Cam17] CAMPEN M.: Partitioning surfaces into quadrilateral patches: A survey. *Computer Graphics Forum 36*, 8 (2017), 567–588. 1

[CBK15] CAMPEN M., BOMMES D., KOBBELT L.: Quantized global parametrization. *ACM Trans. Graph. 34*, 6 (2015). 2, 3

[CK14] CAMPEN M., KOBBELT L.: Quad layout embedding via aligned parameterization. *Comp. Graph. Forum 33*, 8 (2014), 69–81. 4, 7

[DSC09] DANIELS J., SILVA C. T., COHEN E.: Localized quadrilateral coarsening. In *Comp. Graph, Forum* (2009), vol. 28, pp. 1437–1444. 2

[DSSC08] DANIELS J., SILVA C. T., SHEPHERD J., COHEN E.: Quadrilateral mesh simplification. *ACM Trans. Graph. 27*, 5 (2008), 148. 2

[DVPSH14] DIAMANTI O., VAXMAN A., PANOZZO D., SORKINE-HORNUNG O.: Designing *n*-PolyVector fields with complex polynomials. *Computer Graphics Forum 33*, 5 (2014). 2

[EBCK13] EBKE H.-C., BOMMES D., CAMPEN M., KOBBELT L.: QEx: Robust quad mesh extraction. 168:1–168:10. 7

[ECBK14] EBKE H.-C., CAMPEN M., BOMMES D., KOBBELT L.: Level-of-detail quad meshing. *ACM Trans. Graph. 33*, 6 (2014). 2

[EGKT08] EPPSTEIN D., GOODRICH M. T., KIM E., TAMSTORF R.: Motorcycle Graphs: Canonical Quad Mesh Partitioning. *Comp. Graph. Forum 27*, 5 (2008), 1477–1486. 2, 3

[GO21] GUROBI OPTIMIZATION L.: Gurobi optimizer reference manual, 2021. URL: http://www.gurobi.com. 7

[KCPS13] KNÖPPEL F., CRANE K., PINKALL U., SCHRÖDER P.: Globally optimal direction fields. *ACM Trans. Graph. 32*, 4 (2013), 59. 2

[LCBK19] LYON M., CAMPEN M., BOMMES D., KOBBELT L.: Parametrization quantization with free boundaries for trimmed quad meshing. *ACM Trans. Graph. 38*, 4 (2019). 2, 3, 4, 6, 7

[LCK21] LYON M., CAMPEN M., KOBBELT L.: Quad layouts via constrained t-mesh quantization. *Computer Graphics Forum 40*, 2 (2021). 2, 3, 4, 6, 8, 9, 10, 11

[MC19] MANDAD M., CAMPEN M.: Exact constraint satisfaction for truly seamless parametrization. *Comp. Graph. Forum 38*, 2 (2019). 2

[MK12] MÖBIUS J., KOBBELT L.: OpenFlipper: An open source geometry processing and rendering framework. In *Curves and Surfaces*, vol. 6920 of *Lecture Notes in Computer Science*. 2012. 9

[MPZ14] MYLES A., PIETRONI N., ZORIN D.: Robust field-aligned global parametrization. *ACM Trans. Graph. 33*, 4 (2014). 2, 3, 4, 5, 6, 9

[PPM*16] PIETRONI N., PUPPO E., MARCIAS G., SCOPIGNO R., CIGNONI P.: Tracing field-coherent quad layouts. *Computer Graphics Forum 35*, 7 (2016). 2, 4, 8, 9, 10, 11

[PZKW11] PENG C.-H., ZHANG E., KOBAYASHI Y., WONKA P.: Connectivity editing for quadrilateral meshes. *ACM Trans. Graph. 30*, 6 (2011), 141. 2

[RP17] RAZAFINDRAZAKA F. H., POLTHIER K.: Optimal base complexes for quadrilateral meshes. *Computer Aided Geometric Design 100*, 52-53 (2017), 63–74. 2

[RPPSH17] RABINOVICH M., PORANNE R., PANOZZO D., SORKINE-HORNUNG O.: Scalable locally injective mappings. *ACM Trans. Graph. 36*, 4 (2017). 6

[RRP15] RAZAFINDRAZAKA F. H., REITEBUCH U., POLTHIER K.: Perfect matching quad layouts for manifold meshes. *Comp. Graph. Forum 34*, 5 (2015), 219–228. 2

[RS14] RAY N., SOKOLOV D.: Robust polylines tracing for N-symmetry direction field on triangulated surfaces. *ACM Trans. Graph. 33*, 3 (2014). 2, 3

[RVAL09] RAY N., VALLET B., ALONSO L., LÉVY B.: Geometry-aware direction field processing. *ACM Trans. Graph. 29*, 1 (2009). 2

[RVLL08] RAY N., VALLET B., LI W. C., LÉVY B.: N-symmetry direction field design. *ACM Trans. Graph. 27*, 2 (2008), 10:1–10:13. 4

[TPC*10] TARINI M., PIETRONI N., CIGNONI P., PANOZZO D., PUPPO E.: Practical quad mesh simplification. *Computer Graphics Forum 29*, 2 (2010), 407–418. 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). 2, 4

[VCD*16] VAXMAN A., CAMPEN M., DIAMANTI O., PANOZZO D., BOMMES D., HILDEBRANDT K., BEN-CHEN M.: Directional field synthesis, design, and processing. *Comp. Graph. Forum 35*, 2 (2016). 2

[VOS19] VIERTEL R., OSTING B., STATEN M.: Coarse quad layouts through robust simplification of cross field separatrix partitions. *Proc. 28th International Meshing Roundtable* (2019). 2

© 2021 The Author(s)
Computer Graphics Forum © 2021 The Eurographics Association and John Wiley & Sons Ltd.