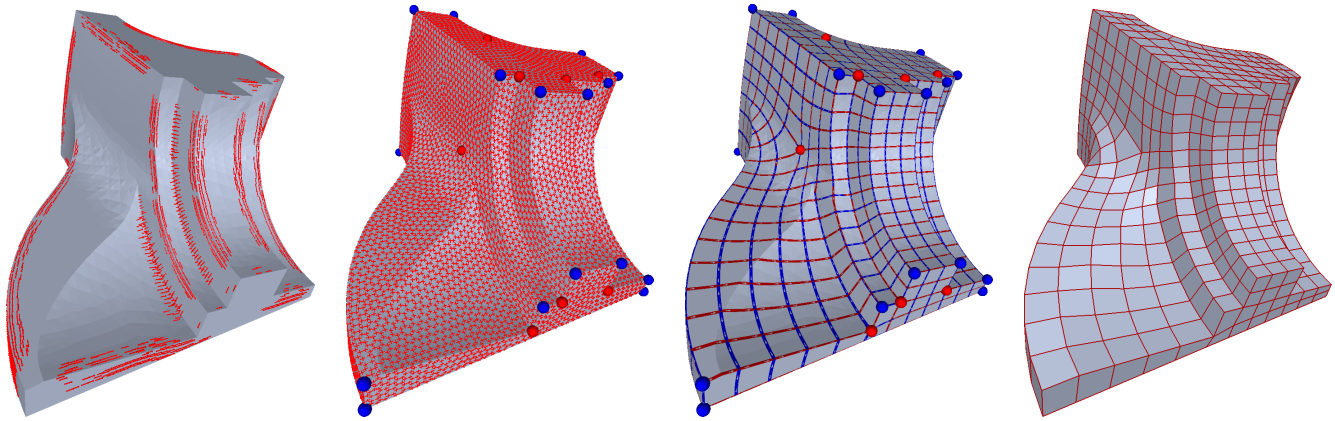
Mixed-Integer Quadrangulation

David Bommes Henrik Zimmer Leif Kobbelt

RWTH Aachen University



(a) (b) (c) (d)

Figure 1: *Quadrangulation example: (a) A sparse set of conservatively estimated orientation and/or alignment constraints is selected on the input mesh by some simple heuristic or by the user. (b) In a global optimization procedure a cross field is generated on the mesh which interpolates the given constraints and is as smooth as possible elsewhere. The optimization includes the automatic generation and placement of singularities. (c) A globally smooth parametrization is computed on the surface whose iso-parameter lines follow the cross field directions and singularities lie at integer locations. (d) Finally, a consistent, feature aligned quadmesh can be extracted.*

Abstract

We present a novel method for quadrangulating a given triangle mesh. After constructing an as smooth as possible symmetric cross field satisfying a sparse set of directional constraints (to capture the geometric structure of the surface), the mesh is cut open in order to enable a low distortion unfolding. Then a seamless globally smooth parametrization is computed whose iso-parameter lines follow the cross field directions. In contrast to previous methods, sparsely distributed directional constraints are sufficient to automatically determine the appropriate number, type and position of singularities in the quadrangulation. Both steps of the algorithm (cross field and parametrization) can be formulated as a mixed-integer problem which we solve very efficiently by an adaptive greedy solver. We show several complex examples where high quality quad meshes are generated in a fully automatic manner.

CR Categories: I.3.5 [Computational Geometry and Object Modeling]: Hierarchy and geometric transformations

Keywords: remeshing, quadrangulation, parametrization, direction field, singularities, mixed-integer

ACM Reference Format
Bommes, D., Zimmer, H., Kobbelt, L. 2009. Mixed-Integer Quadrangulation. *ACM Trans. Graph. 28*, 3, Article 77 (August 2009), 10 pages. DOI = 10.1145/1531326.1531383 http://doi.acm.org/10.1145/1531326.1531383.

1 Introduction

The problem of generating high quality quad meshes from unstructured triangle meshes has received a lot of attention recently. The reason for this interest is that quad meshing converts raw geometric data into a higher representation which effectively supports sophisticated operations like texturing and shape modification. The difficulties in quad meshing arise from the fact that the quality criteria are diverse and their optimization often requires the consideration of global dependencies. The most common quality aspects are:

1. **Individual Element Quality:** Each quad should be close to a rectangle or square, i.e. the four corner points should be coplanar, opposite edges should have equal length and the four interior angles should be 90 degrees.
2. **Orientation:** Away from flat or umbilic points on the surface, mesh edges should be orthogonal to the principal curvature directions such that the dihedral angle across edges captures these curvatures in a natural way.
3. **Alignment:** Sharp features of the surface should be explicitly represented by a sequence of mesh edges in order to minimize the Hausdorff-distance between triangulation and quadrangulation and to prevent normal noise.
4. **Global Structure:** Singularities, i.e. vertices with valence $\neq 4$, are necessary to compensate for the Gaussian curvature. Their number and position must be chosen carefully to capture the global geometric structure since otherwise the element quality and orientation is heavily affected.
5. **Semantics:** In some applications additional requirements emerge from the intended usage of the 3D model and cannot be derived from the geometry alone. In finite element simulation of deformation processes, e.g., the optimal mesh depends

on the rest geometry as well as on the external forces and constraints.

A quadrangulation algorithm should optimize the output simultaneously with respect to all of these criteria. However, while element quality, orientation and alignment are rather simple, the global structure is much more difficult to handle. Consequently, we focus on automatically finding good singularity positions which optimize the global structure of the quadrangulation. Although the overall method is designed to run fully automatic, the user can still manually override some decisions, e.g. by manually shifting a singularity wherever it is necessary to take semantical side-conditions into account (see criterion 5).

Many recent methods use smoothed (discrete) principal curvature directions to guide the quad meshing. The problem with these approaches is that the final singularity positions are effectively determined by the local smoothing operator applied to the initial curvature estimates. Especially in flat or umbilic regions where the initial directions have a random orientation, clusters of singularities may occur. Another problem is oversmoothing which may destroy the original orientation information in feature regions.

To overcome these problems we propose to select only the most relevant and dominant directions as depicted in Figure 1 (a), which can be detected, e.g., by conservative thresholding of some anisotropy measure or by manual selection. Starting with these sparsely distributed direction constraints we then search for the smoothest interpolating cross field. The singularities in this interpolating cross field are mostly due to the surface metric deviating from a planar configuration and not caused by incompatible constraints.

In the second phase of our algorithm the smooth cross field is used as input for a global parametrization method. We cut the mesh open such that we create a surface patch with a disk-like topology where all cross field singularities lie at the boundary. Subsequently, we can compute two piecewise linear scalar fields u and v whose gradients follow the given cross field. Finally, a consistent quadrangulation can be extracted since by construction the parametrization is compatible at the cuts and all singularities are mapped to integer positions along the boundary of the parameter domain.

In both steps of the algorithm the task can be formulated in terms of a mixed-integer problem. These are linear problems where a subset of the variables is continuous ($\in \mathbb{R}$) and the others are discrete ($\in \mathbb{Z}$). In Section 2 we therefore present a greedy solver for this class of problem.

1.1 Related Work

A lot of effort has been spent in the last years to compute high quality quadrangulations. Since there are several nice surveys [Alliez et al. 2005; Hormann et al. 2007], we will discuss here only the most related works. Generally, there are two classes of approaches, namely explicit quadrangulations and parametrization based techniques. Examples of explicit approaches are [Alliez et al. 2003; Marinov and Kobbelt 2004] which trace curves along the principal curvature directions or [Lai et al. 2008] which iteratively transforms a triangular mesh into a quad-dominant mesh. For all such methods it is difficult to obtain coarse meshes consisting of quads only. Most structure aligned parametrization techniques are guided by vector or cross fields usually arising from estimated principal curvature directions [Cohen-Steiner and Morvan 2003] or a manual design process [Zhang et al. 2006; Fisher et al. 2007]. Especially cross fields are promising since they can capture singularities of fractional index which naturally arise in quadrangulations.

Cross fields can be seen as four coupled vector fields. Consequently, smoothing algorithms must be able to handle the discretely switching vector assignments, which can be achieved by a non-linear angle formulation like in [Hertzmann and Zorin 2000]. However, such methods often get stuck in local minima and the result strongly depends on the initial solution.

Recently Ray et al. proposed a formalism to handle N-symmetry direction fields [Ray et al. 2008b] in a linear manner, enabling the computation of globally smooth solutions. However, all singularity positions must be prescribed by the user. In contrast we search for singularity positions which enable the smoothest cross field for a set of sparse directional constraints. Besides the automatic singularity placement, another contribution of our approach is the smooth handling of multiple directional constraints, not possible in [Ray et al. 2008b]. Their user-given hard angle constraints already fix the smoothness between multiple constraints, and do not exploit that the orientation of cross fields is invariant w.r.t. rotations by multiples of 90 degrees and that there might be rotations leading to a smoother cross field.

In the case of highly detailed geometry a smooth cross field naturally requires lots of singularities which can be prevented by the smoothing algorithm of Ray et al. [Ray et al. 2008a]. In contrast to their approach, we interpret the cross field as an infinitely fine quadrangulation and compute all necessary singularities. The merging of singularities is later controlled by the parametrization which can perform singularity cancellations w.r.t. the given target edge length.

Structure aware Parametrization techniques can be divided into two classes, namely high-level methods where the quad orientation is controlled by a rough patch layout and low-level methods where a desired orientation is given per triangle. Dong et al. used the Morse-Smale complex of Laplacian eigenfunctions [Dong et al. 2006] to derive high-level patch layouts. Their method was extended in [Huang et al. 2008] to enable the control over singularity positions, size, orientation and feature alignment. However, computing coarse high-quality results is still involved and requires an experienced user. Tong et al. used high-level user-designed singularity graphs to enrich the space of harmonic one-forms and compute globally smooth parametrizations [Tong et al. 2006]. Once a suitable singularity graph is provided by the user, these harmonic parametrizations produce nice quadrangulations. By allowing affine transition functions and optimizing the charts, [Bommes et al. 2009] improved the distortion of the parametrization.

Ray et. al proposed a fully automatic non-linear parametrization technique which is guided by a low-level vector field and assumes a single chart for each triangle [Ray et al. 2006]. Kälberer et al. developed a linear algorithm by mapping a cross field to a single vector field on a branched covering [Kälberer et al. 2007]. As in our parametrization, a triangle based energy is optimized and the singularity positions are completely defined by the input cross field. Their intermediate parametrization, i.e. the integral of the hodge decomposed vector field which is incompatible at the cuts, is exactly the continuous solution of our mixed-integer formulation. However, ensuring compatibility at the cuts is done in a different way. Instead of rounding the coefficients of the transition functions at once, we apply our proposed greedy strategy which improves the resulting quality. A more detailed comparison will be given later in Section 6.

Automatic cone singularities In the setting of conformal parametrizations, cone singularities as introduced by Kharevych et al. [Kharevych et al. 2006] can be placed in a greedy manner at the local extrema of discrete conformal scaling factors [Ben-Chen et al. 2008]. These positions can be further improved by a non-

linear Gauss-Seidel solver [Springborn et al. 2008]. Both methods are designed to compute conformal parametrizations with a small number of cone singularities and lower distortion. Unfortunately, even when restricting to cross field cone singularities, the resulting positions are often not sufficient for structure aligned parametrizations where singularities are additionally induced by the desired orientations, e.g. the leftmost red singularity lying on the flat part of the fandisk in Figure 1. Furthermore, supporting orientational constraints isn't straight forward in this formulation.

1.2 Contributions

We propose an adaptive greedy solver for mixed integer problems which increases the computation time compared to a continuous linear system solver only moderately. This is achieved by iterative rounding combined with local Gauss-Seidel updates in order to reduce the local residui.

We formulate the quadrangulation problem as a two-step process, cross field generation and global parametrization, which both reduce to a mixed-integer problem.

Our cross field generator is able to take sparsely scattered as well as densely distributed orientation constraints into account. By smoothly interpolating between the constraints the system can automatically place singularities at geometrically meaningful locations.

Our new globally smooth parametrization technique allows us to generate seamless quad meshes while satisfying various constraints like orientation, alignment, and integer singularity locations. An optional anisotropic stretch metric allows us to trade squareness of the quads for improved feature alignment.

2 A Greedy Mixed-Integer Solver

The minimization of a quadratic energy $E(x_1, \dots, x_n)$ is called an *integer problem* if $\mathbf{x} \in \mathbb{Z}^n$. In this paper we encounter more general problems where some of the unknown variables $x_1 \dots x_k$ are integers and the others $x_{k+1} \dots x_n$ are real numbers. Integer and mixed-integer problems are usually very hard to solve exactly, see [Floudas 1995; Gorry et al. 1970] for more details. Hence, a common way to find an approximate solution is to first compute the continuous minimizer, which simply requires the solution of the linear system $\{\partial E/\partial x_i = 0 \mid i = 1 \dots n\}$. Then the first k variables of the solution vector are rounded to the nearest integer and a new minimizer is computed by assuming these rounded values $x_1 \dots x_k$ to be constant.

While this *direct rounding* is a common practice, we observed that, depending on the number of integer variables and on their mutual dependencies, the obtained solution can deviate significantly from the true solution. Hence, we propose an alternative approach which we call *greedy rounding*. The idea is to round the integer variables one at a time, followed by an immediate update of the continuous part of the solution.

Let $\mathbf{x}^0$ be the continuous solution to the linear system and let x_i, $i \leq k$ be the variable which causes the smallest absolute error if we round it to the nearest integer. Then we can set x_i to this integer value and update the linear system by assuming x_i as constant. We solve again for the remaining variables $\mathbf{x}^1$ and continue to eliminate in each step that variable which causes the least round-off error until all variables $x_1 \dots x_k$ have an integer value.

The motivation for this approach is based on the assumption that small round-off errors will only have little impact on the final solution and that by recomputing the free variables after every rounding step we will compensate these errors.

The obvious drawback of greedy rounding is that we have to solve k (= number of integer variables) $+ 1$ linear systems, which increases the computation time prohibitively. However, as we said above, small round-off errors only have a small impact on the solution which can be exploited to design an efficient adaptive solver.

2.1 Adaptive Mixed-Integer Solver

The first idea is to use an iterative solver like the Conjugate Gradient or BiCG (for non-symmetric matrices) which after each rounding reuses the previous solution as the initial value. But we can do even better. We observed that for sparse matrices, typically arising in the case of triangle meshes, it is often sufficient to update a small local set of variables. In this context local means that we have a short path in the dependency graph of all variables. Therefore, we start with a local Gauss-Seidel iteration. That means after variable x_i is rounded we push all variables whose Gauss-Seidel update depends on x_i into a queue. These are exactly the nonzero elements of the row A_i. Now in each iteration step we fetch the first element from the queue, say x_k, and recompute the local residuum

$$r_k = b_k - \sum_{j=1}^{n} A_{kj} x_j$$

If $|r_k|$ is larger than a prescribed tolerance, e.g. 10^{-6}, we update the variable $x_k \rightarrow x_k - r_k/A_{kk}$ and push all variables which depend on x_k onto the queue. The iteration terminates if the queue is empty, i.e. all local residui are within the prescribed tolerance, or a maximum number of iterations is reached.

Algorithm 1 Local Gauss-Seidel

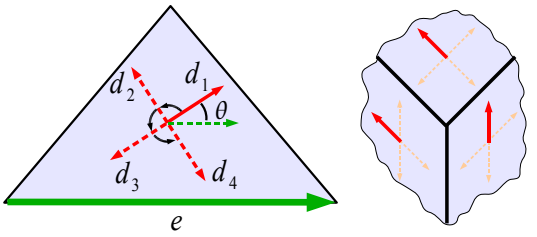
```
1:  x_i = round(x_i)
2:  push nonzero(A_i) into queue
3:  iter = 0
4:  while ( (not queue empty) and (iter < maxiter) ) do
5:      iter = iter + 1
6:      x_k = pop( queue )
7:      r_k = b_k − Σ_{j=1}^{n} A_kj x_j
8:      if (|r_k| > tolerance) then
9:          x_k = x_k − r_k/A_kk
10:         push nonzero(A_k) into queue
11:     end if
12: end while
```

If the local Gauss-Seidel solver does not converge ($iter \geq maxiter$), we first switch to a more global Conjugate Gradient solver and finally, if it is necessary, to a sparse Cholesky solver [Chen et al. 2006]. Such time consuming Sparse-Cholesky computations are only necessary when a variable with large impact is rounded. Our experiments showed that this adaptive solver is more efficient than restricting to pure iterative solvers. More detailed statistics about the solver are given in Section 6.

In the future we plan to publish our implementation of the presented solver which can be applied to arbitrary quadratic mixed-integer problems. For maximum flexibility, we added the handling of linear constraints, by internally eliminating a variable for each independent constraint.

3 Salient Curvature Directions

In the vicinity of flat or umbilic points, the principal curvature directions are ill defined. Consequently, using the principal curvature directions as a dense guiding field for quadrangulation leads to suboptimal results. Typical artifacts are noisy directions with badly



(a) (b)

Figure 2: *(a) The four cross field directions in a triangle are parametrized by the angle θ w.r.t. a local reference edge e. (b) Depicts a smooth cross field in the vicinity of a cube corner, where the red arrows reflect the corresponding period jumps.*

placed singularities or even clusters of unnecessary singularities. Generally, these artifacts cannot be removed by cross field smoothing algorithms, since the configurations often form local minima. Therefore, in contrast to other methods, we aim at finding the smoothest cross field, interpolating only sparse directional constraints that can be found in a reliable manner.

The directions we want to identify are in the spirit of feature lines, as computed in [Hildebrandt et al. 2005]. However in our case a simple heuristic which robustly identifies parabolic regions is sufficient. Since parabolic regions are equipped with a well-defined orientation they are the best candidates to guide a quadrangulation. Parabolic regions can be identified by measuring the relative anisotropy of the principal curvatures

$$\tau = \frac{||\kappa_{max}| - |\kappa_{min}||}{|\kappa_{max}|} \in [0,1]$$

which is defined to be zero, if κ_{max} is zero.

Computing meaningful curvatures on discrete triangle meshes is involved. A common technique is evaluating the shape operator [Cohen-Steiner and Morvan 2003] of a geodesic disk near a point $\mathbf{p}$. But depending on the radius r we will get different estimates. To achieve a more stable result we compute for each point a set of shape operators S_r with different geodesic radii $r \in [r_0, r_1]$ and select the most promising one with a simple heuristic. A shape operator S_r is said to be valid if all shape operators in the interval $[r-w, r+w]$ have a relative anisotropy larger than a prescribed threshold τ_{min} and a mean curvature larger than K to exclude almost flat regions. For all points which provide a valid shape operator, we add a directional constraint. If there are multiple valid candidates for a single point we choose the one with the most stable direction, i.e. the one with the minimal angle deviation within its interval.

Fortunately all necessary coefficients of this heuristic have an intuitive meaning. Appropriate directions should be stable within a range depending on the target edge length h. Following this observation we choose $w = h/4$. Furthermore in our experiments we chose r_0 to be the average length of all triangle edges, $r_1 = h$, $\tau_{min} = 0.8$ and $K = 0.1/b_s$, where b_s is the radius of a bounding sphere. In general the quadrangulation result is not very sensitive w.r.t. these parameters, since similar cross fields can be generated with a large range of different sparse constraints, generated with slightly different parameters.

4 Smooth cross fields

In this section we will use the elegant formalism for N-Symmetry direction fields [Ray et al. 2008b] where a cross field ($N = 4$) on a triangle mesh $M = (V, E, F)$ is defined by an angle-field $\theta : F \mapsto R$ assigning a real number to each face and a period-jump field $p : E \mapsto \mathbb{Z}$ assigning an integer to each edge. The main idea is to use the angles θ to determine a single unit length vector-field which then extends to a symmetric cross field by applying three rotations of $\frac{\pi}{2}$ as shown in Figure 2 (a). Because a cross consists of four vectors between neighboring triangles it is necessary to identify which vector of the first cross is associated with which vector of the second cross. All these topological issues are handled by the period-jumps, as illustrated in Figure 2 (b) for a smooth cross field near the corner of a cube. In this section we will summarize only the discrete results about cross fields that we need in this paper. For more details see [Ray et al. 2008b].

4.1 Measuring cross field smoothness

After fixing the topology, measuring the smoothness of a cross field reduces to measuring the smoothness of one of the four rotation symmetric vector-fields.

The smoothness of a unit vector-field can be measured as the integrated squared curvature of the direction field. Following [Ray et al. 2008b], on a discrete triangle mesh it turns out to be simply the sum of all squared angle differences between neighboring triangles:

$$E_{smooth} = \sum_{e_{ij} \in E} (\theta_i - \theta_j)^2$$

where θ_i is the angle of triangle i and neighboring angles are represented in a common coordinate frame, which is always possible by flattening both triangles along their common edge. However, for a surface with non-zero Gaussian curvature it is not possible to find a global coordinate frame. Therefore, a local coordinate frame is used for each triangle, where the x axis is identical to the first edge $\mathbf{e}$ of the triangle (Figure 2(a)). Thus, by incorporating the coordinate transformations between neighbors we can express the smoothness energy of a cross field:

$$E_{smooth} = \sum_{e_{ij} \in E} (\underbrace{\theta_i + \kappa_{ij} + \frac{\pi}{2} p_{ij}}_{\theta_i \text{ w.r.t. frame } j} - \theta_j)^2 \qquad (1)$$

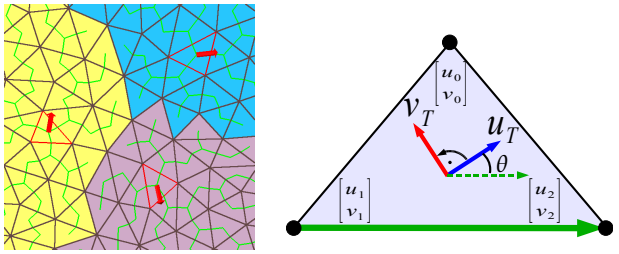
where $\kappa_{ij} \in (-\pi, \pi]$ is the angle between both local frames and p_{ij} is the integer valued period jump across edge e_{ij}. The cross field index of a vertex can be computed as

$$I(v_i) = I_0(v_i) + \sum_{e_{ij} \in N(v_i)} \frac{p_{ij}}{4}$$

with the constant integer valued base index

$$I_0(v_i) = \frac{1}{2\pi} \left(A_d(v_i) + \sum_{e_{ij} \in N(v_i)} \kappa_{ij} \right)$$

and $A_d(v_i)$ is the angle defect of vertex v_i. Only singularities of the cross field have a nonzero index which is always a multiple of $\frac{1}{4}$ [Ray et al. 2008b], e.g. $\frac{1}{4}$ and $-\frac{1}{4}$ for quadrangulation configurations corresponding to valence 3 and 5 respectively.



(a) (b)

Figure 3: *(a) The three constrained faces (red) are the roots of dual spanning trees (green) covering the respective Voronoi cells. Each cell contains only one constraint and along all branches of the tree zero period jumps can be propagated without changing the total smoothness energy. (b) With the angle θ w.r.t. the local reference direction (green) the cross field directions $\mathbf{u}_T$, $\mathbf{v}_T$ can be extracted and used for the parametrization. In the computation two linear scalar functions (u, v) are sought whose gradients are oriented consistently with the cross field directions.*

4.2 Finding a smooth, interpolating cross field

Equipped with these basic definitions we are ready to formulate the optimization problem. Given a mesh M and a subset of faces $F_c \subset F$ with constrained directions $\theta_i = \hat{\theta}_i$, we search for the smoothest interpolating cross field, i.e. we want to minimize (1). Accordingly we have to find an integer p_{ij} per edge and a real valued angle θ_i per face.

Reducing the Search Space: Up to here there is a whole space of equivalent minimizers to the energy (1). To understand this, assume we have already computed a minimizer which for one triangle provides the angle θ_0 and the three period jumps p_{01}, p_{02} and p_{03}. If we now rotate the vector by a multiple of $\frac{\pi}{2}$, i.e. set $\tilde{\theta}_0 = \theta_0 + k \cdot \frac{\pi}{2}$ and compensate this change by updating the affected period jumps to $\tilde{p}_{0i} = p_{0i} - k$, the smoothness energy is unchanged. We can repeat this procedure for all free triangles $f \in F \setminus F_c$. Consequently the solution can be made unique by fixing one period jump per free triangle to an arbitrary value, e.g. zero, without changing the energy of the minimizer. Care should be taken not to fix edges whose dual path connects two constrained faces, as done in [Ray et al. 2008b], or closes loops because in these cases the cross field curvature along this path would be fixed to an arbitrary value and is not the intended result of the minimizer.

A valid set of edges, whose period jumps are allowed to be set to zero, can be found by constructing a forest of Dijkstra trees of the dual mesh as shown in Figure 3. Each constrained face in F_c is the root of a separate tree such that no tree connects constrained faces. The number of fixed edges is exactly $|F \setminus F_c|$ since starting from the constrained faces each other face of the mesh is conquered by adding a single edge. Notice that no dual loop can be closed by a tree structure, such that we end up with a valid set of edges which can be fixed to zero period jumps without changing the energy of the minimizer.

Obviously there are many other valid sets of edges which could be fixed. The reason why we use trees living in the discrete Voronoi cells of the corresponding constrained faces is that this choice minimizes the length of a path to its corresponding constraint and so improves the accuracy of the greedy mixed-integer solver.

Additionally to the period jumps on tree edges each period jump between two adjacent constrained faces f_i and f_j can be fixed to

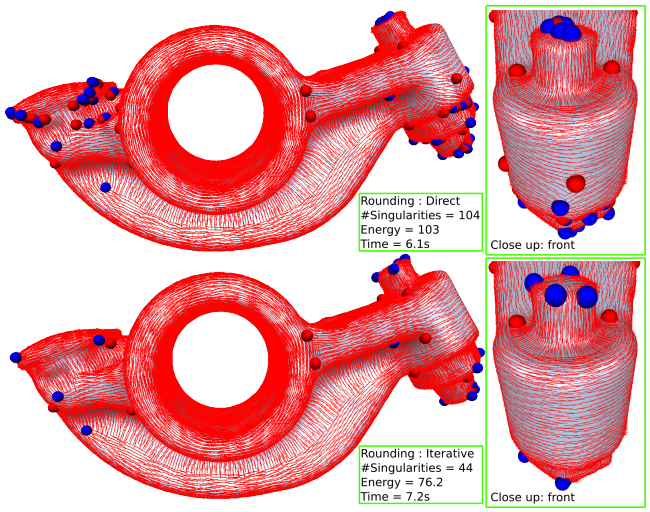


Figure 4: *Greedy rounding yields a smaller smoothness energy and fewer singularities (bottom), whereas the direct rounding produces unnecessary singularities and a higher energy (top). Note that these are the singularities and the field as they emerge from the solver, no singularity optimization has been carried out.*

$p_{ij} = round(2/\pi(\hat{\theta}_j - \hat{\theta}_i - \kappa_{ij}))$, since p_{ij} is only part of a single quadratic term in (1), which is independent from other variables.

In summary we end up with a mixed-integer problem consisting of $|F \setminus F_c| \approx 2|V|$ real valued variables θ_i and $|E| - |F \setminus F_c| \approx |V|$ integer valued variables p_{ij}.

Mixed-Integer Formulation: To apply the greedy mixed-integer solver from Section 2 it is sufficient to assemble the system of linear equations by setting the gradient of the energy (1) to zero:

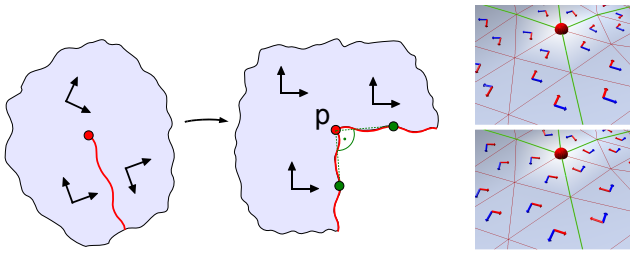
$$\frac{\partial E_{smooth}}{\partial \theta_k} = \sum_{e_{kj} \in N(f_i)} 2(\theta_k + \kappa_{kj} + \frac{\pi}{2} p_{kj} - \theta_j) \stackrel{!}{=} 0 \quad (2)$$

$$\frac{\partial E_{smooth}}{\partial p_{ij}} = \pi(\theta_i + \kappa_{ij} + \frac{\pi}{2} p_{ij} - \theta_j) \stackrel{!}{=} 0 \quad (3)$$

Notice that the values on edges are antisymmetric, i.e. $p_{ij} = -p_{ji}$ and $\kappa_{ij} = -\kappa_{ji}$, which can lead to sign changes in equations (2) and (3). For all variables which are not fixed, we set up a row and assemble all of them into a single matrix. After applying our greedy mixed-integer solver, the result is a smooth cross field where the integer valued period jumps define type and position of all singularities. Figure 4 compares the result of our greedy solver with that of a direct rounding, where red and blue spheres represent singularities with negative and positive index respectively.

In practice we observed that some singularity positions, especially those in flat regions, can sometimes be improved by a local search algorithm, as described in the next section.

Local Search Singularity Optimization: In a postprocess we optionally check for each singularity, if the energy can be decreased by moving it to a neighboring vertex. Moving a singularity along an edge e_{ij} means changing the corresponding period jump p_{ij}. Notice, that by this operation only the right-hand-side of the linear system is changed. Consequently we can precalculate the sparse Cholesky factorization of this matrix once and then compute solutions for different right-hand-sides efficiently [Botsch et al. 2005].



(a) (b)

Figure 5: *(a) By placing a cut to a cone singularity* $\mathbf{p}$ *(here of index* $\frac{1}{4}$*) a distortion free unfolding of the patch is possible. (b) The upper image shows two directions of the cross field. In the lower image the mesh is cut into disk topology along the green edges, such that these directions can be consistently oriented on each side of the cut.*

5 Global Parametrization

We now compute a global parametrization, i.e., a map from the given mesh $\mathcal{M}$ to some disk-shaped parameter domain $\Omega \in \mathbb{R}^2$. Since the parametrization should be piecewise linear, it is sufficient to assign a (u, v) parameter value to each vertex — more precisely to each triangle corner — in the mesh.

The parametrization should be locally oriented according to the optimized cross field from Section 4 which implies that the gradients of the piecewise linear scalar fields u and v defined on the mesh $\mathcal{M}$ should minimize the local orientation energy

$$E_T \;=\; \|h\nabla u - \mathbf{u}_T\|^2 + \|h\nabla v - \mathbf{v}_T\|^2$$

for each triangle T. Here h is a global scaling parameter which controls the edge length of the resulting quad mesh. The vectors $\mathbf{u}_T$ and $\mathbf{v}_T$ are two orthogonal vectors in T corresponding to the cross field directions θ and $\theta + \pi/2$. Since the cross field is defined only up to rotations by $\pi/2$ we will have to specify which of the four possibilities we are picking in each triangle such that the proper compatibility conditions are satisfied across each edge in the mesh.

The global orientation energy is then defined as the integral of E_T over the entire mesh $\mathcal{M}$

$$E_{orient} \;=\; \int_{\mathcal{M}} E_T \, dA \;=\; \sum_{T\in\mathcal{M}} E_T \, area(T). \qquad (4)$$

The minimizer of this quadratic functional is obtained by solving the sparse linear system which sets all the partial derivatives of E_{orient} to zero.

Cutting the mesh: In order to be able to compute a proper parametrization minimizing E_{orient} we have to cut open the mesh $\mathcal{M}$, such that we obtain a patch that is topologically equivalent to a disk. An additional requirement is that all singular vertices must lie on the cut, i.e. at the boundary of the parameter domain. The reason is that the angle defect of a singularity cannot be represented by an inner vertex of the parametrization as depicted in Figure 5. We compute an appropriate cut graph in two steps.

First we start from a random triangle and grow a topological disk by constructing a dual spanning tree. Thus the primal of all non spanning tree edges is already a cut graph which transforms $\mathcal{M}$ into disk topology. The size of this cut graph can be significantly reduced by iteratively removing all open paths.

In the second step paths connecting each singularity to the cut graph are added. This can be done by successively applying Dijkstra's shortest path algorithm.

At the end of the two cutting steps we have a triangle mesh patch where all the singularities are located at the boundary. If a singularity is not a leaf node of the cut graph then it appears several times along the boundary. In order to compute a parametrization we have to find a planar embedding of this boundary polygon as well as all the interior vertices. The location of the mesh vertices in the parameter domain is computed by minimizing E_{orient}, however, there are a number of consistency constraints that have to be taken into account.

Integer location of singularities: By allowing a singularity to be in general position, it would cause an n-sided face instead of a valence-n vertex. Therefore to guarantee a pure quadrangulation, we have to snap all singularities to integer locations in the parameter domain. This means that the overall parametrization task is now a mixed-integer problem which we solve by our mixed-integer greedy solver from Section 2.

Cross boundary compatibility: In order to avoid visible seams across the cut paths on the surface we have to make sure that the quad structure on both sides of a cut edge is compatible. This is guaranteed by allowing only a grid automorphism as a transition function. This requires that the (u, v) parameter values on both sides of a cut edge are related by

$$(u', v') \;=\; Rot^i_{90}\,(u, v) + (j, k)$$

with integer coefficients (i, j, k).

The rotation coefficient in the transition functions can easily be computed by propagating a globally consistent orientation in the cross field, as illustrated in Figure 5 . Since after the cutting, all interior vertices of the mesh are regular, we can start at a random face and propagate its orientation in a breadth first manner to all the neighboring faces. This will establish a zero-rotation across all inner edges. The rotations Rot^i_{90} across the cut edges can be found by simply comparing the orientations in neighboring faces.

After fixing the rotations, the cross boundary compatibility conditions can be incorporated into the optimization scheme as linear constraints. Therefore for each cut edge $\mathbf{e} = \overline{\mathbf{p}\,\mathbf{q}}$ we introduce two integer variables $j_\mathbf{e}, k_\mathbf{e}$ to formulate the four compatibility conditions:

$$\begin{aligned}(u'_p, v'_p) &= Rot^{i_\mathbf{e}}_{90}\,(u_p, v_p) + (j_\mathbf{e}, k_\mathbf{e})\\ (u'_q, v'_q) &= Rot^{i_\mathbf{e}}_{90}\,(u_q, v_q) + (j_\mathbf{e}, k_\mathbf{e})\end{aligned}$$

Hence, in total we add two integer variables and eliminate four continuous variables per cut edge.

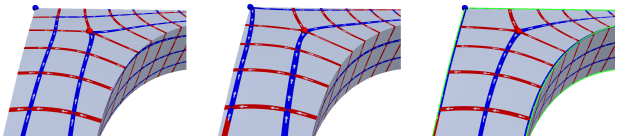
Applying our mixed-integer greedy solver to this parametrization task can be understood in an intuitive way. After computing an all-continuous solution, which corresponds to the unconstrained parametrization, we iteratively snap the singularities to integer locations.

5.1 Anisotropic Norm

In practice exact orientation is often more important than exact edge length. The reason is that changing the orientation along a highly curved feature line, the quadrangulation quality will drop off dramatically due to normal noise. The orientation can be improved by less penalizing stretch which is in the direction of the desired iso-lines. This can be achieved by an anisotropic norm

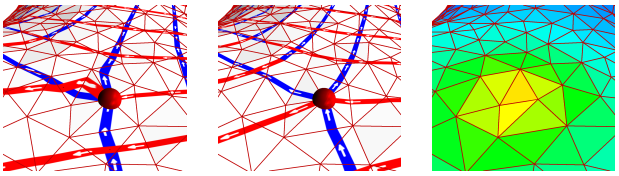
$$\|(u, v)\|^2_{(\alpha,\beta)} = \alpha u^2 + \beta v^2$$

which penalizes the deviation along the major directions with different weights. Notice, such a diagonal metric is sufficient since we



(a) (b) (c)

Figure 6: *The parametrization in (a) is not aligned to the sharp edges of the object. Using the anisotropic norm the quads are allowed to stretch in order to better align with a given input field as shown in (b). In (c) alignment constraints have been imposed, leading to perfectly preserved features.*



(a) (b) (c)

Figure 7: *In (a) the minimized orientation energy produces flipped triangles, which can be removed by local stiffening (b). The chosen weighting is shown in (c) and decreases from orange to blue.*

use $(\mathbf{u}_T, \mathbf{v}_T)$ as the local coordinate frame in each triangle.

$$E_T \;=\; \|h\nabla u - \mathbf{u}_T\|^2_{(\gamma,1)} + \|h\nabla v - \mathbf{v}_T\|^2_{(1,\gamma)}$$

with $\gamma \leq 1$. Figure 6 (b) shows an example, where the orientation of the parametrization is improved by using the anisotropic norm.

5.2 Feature Line Alignment

Sharp feature lines of the input mesh should be preserved in the quadrangulation. Given a subset $S \subset E$ of triangle mesh edges, the necessary alignment conditions can be incorporated in a straightforward way. First of all, alignment requires correct orientation. Therefore, while computing the cross field, all edges in S are used as orientational constraints in both adjacent triangles. Additionally to the correct orientation for alignment, a constant integer coordinate along the edge is necessary, which guarantees that this edge is preserved in the quadrangulation. Each alignment condition for an edge $\overline{\mathbf{p}\,\mathbf{q}}$ can be formulated independently. If the cross field direction $\mathbf{u}_T$ is already oriented along the alignment edge, we end up with a simple condition for the v parameter values

$$v_p \;=\; v_q \;\in \mathbb{Z}$$

which ensures that $\overline{\mathbf{p}\,\mathbf{q}}$ is mapped to an integer valued iso-line. The $u = const$ case is handled analogously. Consequently, for each alignment edge a single variable can be eliminated and the remaining integer variables can be handled by the greedy mixed-integer solver. Figure 6 (c) shows an example, where all feature edges are aligned.

Notice that for meshes with boundaries we can exploit the presented alignment functionality to guarantee that the boundaries are preserved in the quadrangulation and thus prevent jagged boundary lines (see Figure 9).

5.3 Singularity Relocation

By computing a parameterization with the given target edge length h , new requirements have to be taken into account which cannot be anticipated by the cross field computation, since it is independent from h. Examples are singularities which are too close to each other, a boundary or a given alignment edge. Other aspects are symmetries which are irrelevant for a smooth cross field, but important for a quadrangulation. Therefore, to achieve maximal quality it can be necessary to relocate the singularties w.r.t. the requirements of the parameterization. This can be done with a local search algorithm similar to Section 4.2. Depending on how much time is available we can restrict the search to the best local candidate, i.e. the closest neighbor in the parametrization, or evaluate the quality of all neighbors. In each step it is necessary to recompute the smooth cross field w.r.t. the relocated singularity as well as the parametrization. In the cross field computation the cross field indices are now prescribed by linear constraints. Movements are performed if the overall quality improves, i.e. the energy (4) decreases.

The obvious drawback of this singularity relocation is its heavy computational cost. Fortunately in all of our examples the initial singularity positions were already sufficient. However, coarsely quadrangulating meshes with fine details will require singularity relocation.

5.4 Local Stiffening

The parametrization is the result of a quadratic energy minimization. Thus despite the global optimum, for a few triangles it might happen that the metric distortion gets very high or even worse, that the orientation of a mapped triangle flips. Figure 7 shows an example where such a problem occurs in the vicinity of a singularity. The idea of local stiffening is to add an adapted triangle weighting $w(T)$ into the energy formulation to penalize high local distortions, yielding:

$$E_{orient} \;=\; \sum_{T\in\mathcal{M}} w(T)\, E_T\, area(T)$$

This weighting, which is initialized to one, can be updated iteratively, as described in the following, until the quality of the parametrization is sufficient.

The metric distortion is characterized by the singular values σ_1 and σ_2 of the Jacobi matrix as described in [Hormann et al. 2007]. Furthermore to penalize flips we evaluate the orientation of a triangle

$$\tau = sign(det \begin{bmatrix} u_1 - u_0,\, u_2 - u_0 \\ v_1 - v_0,\, v_2 - v_0 \end{bmatrix})$$

where (u_i, v_i) are the vertex parameter coordinates in counterclockwise ordering. We measure the local distortion of each triangle by

$$\lambda = |\tau\frac{\sigma_1}{h} - 1| + |\tau\frac{\sigma_2}{h} - 1|$$

which respects the edge length h. Finally, we update the weight of a triangle by evaluating a uniform Laplacian defined on the dual mesh

$$w(T) \;\leftarrow\; w(T) + min\{c \cdot |\triangle\lambda(T)|,\, d\}$$

with the proportionality constant c and a maximal allowed update of d, which we chose as $c = 1$ and $d = 5$ in all our examples. Notice that directly using the distortion instead of the Laplacian wouldn't be a good idea. The reason is that the weighting would reflect the *global* stretch distribution, which is necessary for a globally consistent quadrangulation, instead of the desired *local* distortions. Subsequently, we increase the smoothness of the weighting field $w(T)$ by a few uniform smoothing steps, which in general leads to nicer quadrangulations.

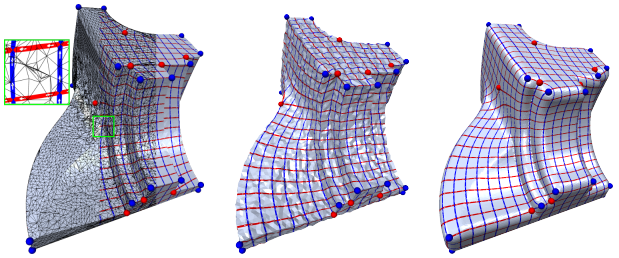


Figure 8: *The presented algorithm is robust w.r.t. bad triangles (a) and can produce meaningful singularities in the presence of noise (b) and on smooth offset geometries (c).*

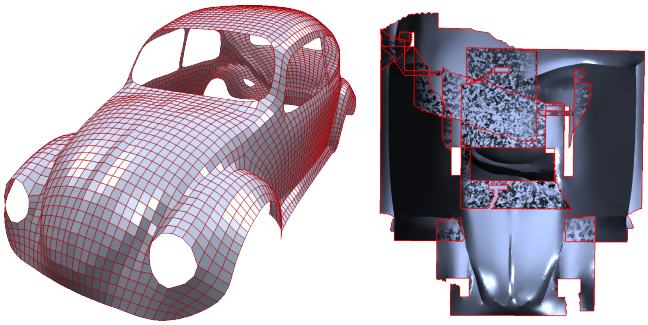


Figure 9: *Quadrangulation of the* BEETLE *model having 11 boundaries. On the right the parametrization is shown. Naturally, due to the occurrence of* $-\frac{1}{4}$ *singularities, parts of the flattening are overlapping.*

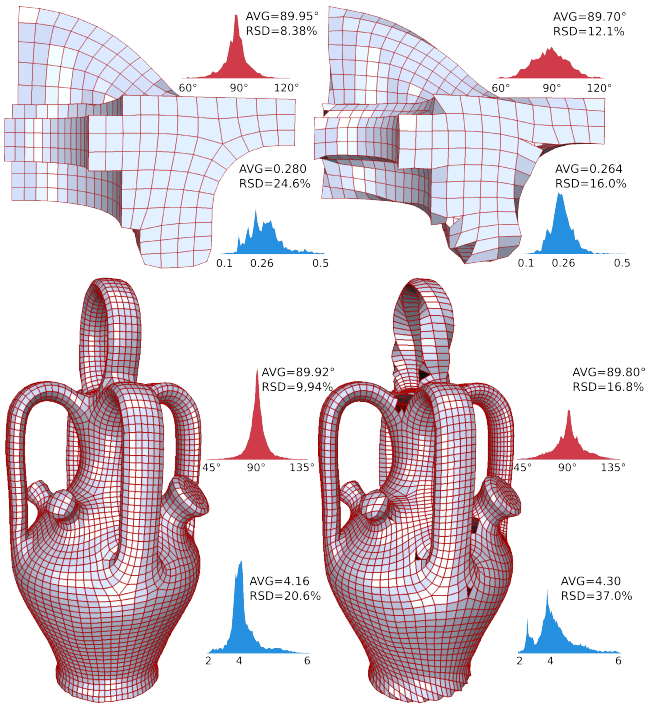


Figure 10: *A comparison between the technique described in this paper (left) and the QuadCover approach by [Kälberer et al. 2007] (right) for a sharp object (*FANDISK*) and a smooth object (*BOTIJO*). In both comparisons the same target edge length and the same cross field generated by our mixed-integer formulation were used.*

6 Comparisons and Results

The backbone of our approach is the mixed-integer solver introduced in Section 2, which is used for the computation of both the smooth cross field and the parametrization. Although it is often necessary to round tens of thousands of variables for the cross field computation, the timings in Table 1 show that this can be done efficiently using the proposed solver.

The example in Figure 4 shows that the greedy rounding leads to a significantly smoother cross field with less singularities compared to the direct rounding approach. All our experiments confirmed this behavior.

A comparison between our approach and QuadCover [Kälberer et al. 2007] is carried out in Figure 10. In both examples the same input cross field and target edge length have been used. The FANDISK comparison clearly shows the benefit of alignment on models with sharp feature edges, while the limitations of direct rounding are especially noticeable on the BOTIJO. On complex objects having many singularities or when remeshing with very coarse target edge length the direct rounding generates many ”twists” and non-injectivities in the parametrization, such that the extraction of a hole-free quad mesh is not always possible. However, the combination of greedy rounding and local stiffening allow us to automatically generate consistent, hole-free quadrangulations at almost any resolution and with significantly less ”twists”.

The spectral approach [Huang et al. 2008] also produces oriented and aligned parametrizations with few singularities, however the Morse-Smale Complex sometimes fails to capture the detailed structure of the surface. This can lead to an unfavorable stretch

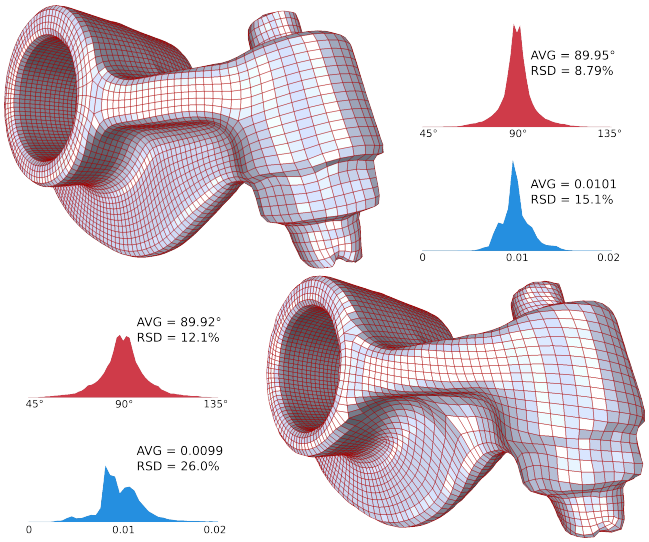


Figure 11: ROCKERARM *comparison between the technique described in this paper (top) and the spectral quadrangulation approach by [Huang et al. 2008] (bottom). The upper mesh has 9413 faces and 36 singularities, the lower one has 9400 faces and 26 singularities.*

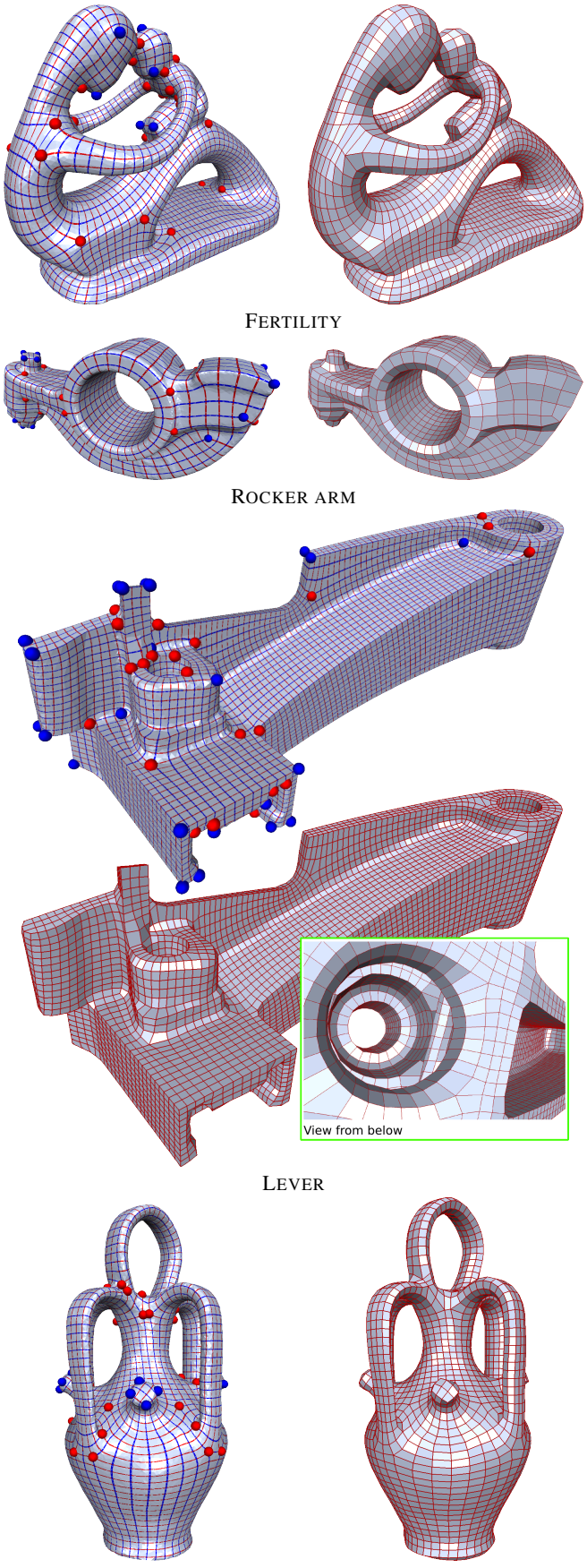


Figure 12: *Results of our Mixed-Integer Quadrangulation approach*

	Solver Statistics								Quadmesh statistics	
	Cross Field					Parametrization				
Model	Dim	#Int	#IS	#DS	Time	Dim	#Int	Time	#Sing	#Quad
FERTILITY	29342	10621	762	11	6.6s	27954	108	25.7s	48	3357
ROCKERARM	32843	12064	385	7	7.6s	41552	72	25.5s	36	1127
LEVER	21113	8254	62	2	3.3s	19331	148	19.9s	83	7880
BOTIJO	25821	9611	669	7	5.4s	29994	164	44.0s	74	8395
FANDISK	17820	6447	209	6	2.2s	13776	19	2.4s	30	764
BEETLE	46425	15827	562	14	13.3s	34705	82	10.3s	6	3778

Table 1: *Statistics of the Greedy Mixed-Integer Solver used for computing the cross field (Section 4) and the parametrization (Section 5).* Dim *refers to the initial dimension of the linear system,* #Int *is the number of integer variables,* #IS *and* #DS *is the number of calls to iterative and direct solvers respectively.* Time *is the total time for the solution. Due to the global nature of the parametrization, the local and iterative search seldom lead to a gain of efficiency and therefore* Time *refers solely to the direct solver.*

of the quads affecting the angle as well as the edge length distribution. A comparison between [Huang et al. 2008] and our approach can be found in Figure 11.

Quadrangulations computed by our technique typically have angle distributions with a sharp peak around 90° and an edge length distribution centered around the target edge length. However, for aligned meshes, like the FANSIDK in Figure 10, further peaks, which reflect the unavoidable stretch may occur in the edge length histogram.

The geometrically complex examples shown in Figure 12 underline the ability of our method to compute coarse, oriented quadrangulations with naturally placed singularities.

All examples were computed on a 3.0GHz standard PC, the statistics are shown Table 1. Interestingly the cross field computation is less demanding to compute than the parametrization, even though it requires practically two orders of magnitude more roundings. This effect is due to the locality of the cross field energy (Equation (1)). Rounding a period jump mainly affects a local neighborhood on the mesh and the solution can be efficiently updated by local Gauss-Seidel iterations. Whereas, rounding a corner point in the parametrization domain usually has global impact. Motivated by this observation and the typically low number of integer variables for the parametrization, we restricted the greedy solver to sparse Cholesky updates.

Finally Figure 8 demonstrates the robustness of the mixed-integer quadrangulation approach w.r.t. different (degenerate) representations of a given object. The mesh in Figure 8 (a) contains almost 1000 triangles with vanishing area (the close-up shows a part of the mesh where about 8 triangles are nearly colinear), the model in Figure 8 (b) has been subjected to normal noise with a magnitude of 0.3% of the bounding box diagonal and the right most model (Figure 8 (c)) was offset, yielding a mesh without sharp corners. These fandisks and most of the other triangle meshes used in this work (along with the extracted quad meshes) can be found in the supplementary material of this paper.

7 Conclusion and Future Work

We have presented a complete quadrangulation method which starts with a pre-process that finds reliable orientation constraints. Based on these, possibly sparsely distributed, constraints we compute a smooth cross field on the surface. The global optimization produces a set of singularities that are automatically placed at geometrically meaningful locations. The cross field is used as input for a global

parametrization technique which cuts the surface open into a disk-like patch and then computes a planar embedding which takes orientation, alignment, as well as boundary compatibility constraints into account.

In the future we would like to integrate both parts of the algorithm into a single optimization scheme. Instead of making the parametrization smooth by least squares approximation of a smooth cross field it would be more natural to smooth the parametrization directly. However this would most probably lead to a non-linear optimization.

One limitation of our method is that for coarse quadrangulations of highly complex models with many cross field singularities, the local singularity relocation in the parametrization step is dominating the overall computation time. Here we would like to develop a more global search strategy.

Acknowledgements

This work has been supported by the UMIC Research Centre, RWTH Aachen University. We would like to thank Felix Kälberer and Matthias Nieser for their helpful support, Tamal Dey, Muyang Zhang, *AIM@SHAPE* and Carlos Hernández (*www.tsi.enst.fr/3dmodels*) for providing us with datasets, Jan Möbius for the geometry processing framework *OpenFlipper.org* and the reviewers for their competent and helpful comments.

References

ALLIEZ, P., COHEN-STEINER, D., DEVILLERS, O., LEVY, B., AND DESBRUN, M. 2003. Anisotropic polygonal remeshing. *ACM Trans. Graph. 22*, 3, 485–493.

ALLIEZ, P., UCELLI, G., GOTSMAN, C., AND ATTENE, M. 2005. Recent advances in remeshing of surfaces. Research report, AIM@SHAPE Network of Excellence.

BEN-CHEN, MIRELA, GOTSMAN, CRAIG, BUNIN, AND GUY. 2008. Conformal flattening by curvature prescription and metric scaling. *Computer Graphics Forum 27*, 2 (April), 449–458.

BOMMES, D., VOSSEMER, T., AND KOBBELT, L. 2009. Quadrangular parameterization for reverse engineering. Lecture Notes in Computer Science, to appear.

BOTSCH, M., BOMMES, D., AND KOBBELT, L. 2005. Efficient linear system solvers for mesh processing. In *IMA Conference on the Mathematics of Surfaces*, Springer, R. R. Martin, H. E. Bez, and M. A. Sabin, Eds., vol. 3604 of *Lecture Notes in Computer Science*, 62–83.

CHEN, Y., DAVIS, T. A., HAGER, W. W., AND RAJAMANICKAM, S. 2006. Algorithm 8xx: Cholmod, supernodal sparse cholesky factorization and update/downdate. Technical Report TR-2006-005, University of Florida.

COHEN-STEINER, D., AND MORVAN, J.-M. 2003. Restricted delaunay triangulations and normal cycle. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, 312–321.

DONG, S., BREMER, P.-T., GARLAND, M., PASCUCCI, V., AND HART, J. C. 2006. Spectral surface quadrangulation. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, 1057–1066.

FISHER, M., SCHRÖDER, P., DESBRUN, M., AND HOPPE, H. 2007. Design of tangent vector fields. *ACM TOG 26*, 3, 56.

FLOUDAS, C. A. 1995. *Nonlinear and Mixed-Integer Optimization Fundamentals and Applications*. Hardback.

GORRY, G., SHAPIRO, J., AND WOLSEY, L. 1970. *Relaxation methods for pure and mixed integer programming problems*. Cambridge, M.I.T., Cambridge.

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating smooth surfaces. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 517–526.

HILDEBRANDT, K., POLTHIER, K., AND WARDETZKY, M. 2005. Smooth feature lines on surface meshes. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 85.

HORMANN, K., LÉVY, B., AND SHEFFER, A. 2007. Mesh parameterization: theory and practice. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, 1.

HUANG, J., ZHANG, M., MA, J., LIU, X., KOBBELT, L., AND BAO, H. 2008. Spectral quadrangulation with orientation and alignment control. *ACM Trans. Graph. 27*, 5, 1–9.

KÄLBERER, F., NIESER, M., AND POLTHIER, K. 2007. Quadcover - surface parameterization using branched coverings. *Computer Graphics Forum 26*, 3 (Sept.), 375–384.

KHAREVYCH, L., SPRINGBORN, B., AND SCHRÖDER, P. 2006. Discrete conformal mappings via circle patterns. *ACM Trans. Graph. 25*, 2, 412–438.

LAI, Y.-K., KOBBELT, L., AND HU, S.-M. 2008. An incremental approach to feature aligned quad dominant remeshing. In *SPM '08: Proceedings of the 2008 ACM symposium on Solid and physical modeling*, 137–145.

MARINOV, M., AND KOBBELT, L. 2004. Direct anisotropic quad-dominant remeshing. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, IEEE Computer Society, Washington, DC, USA, 207–216.

RAY, N., LI, W. C., LÉVY, B., SHEFFER, A., AND ALLIEZ, P. 2006. Periodic global parameterization. *ACM Trans. Graph. 25*, 4, 1460–1485.

RAY, N., VALLET, B., ALONSO, L., AND LÉVY, B. 2008. Geometry aware direction field design. Tech. rep., INRIA - ALICE Project Team. Accepted pending revisions.

RAY, N., VALLET, B., LI, W. C., AND LÉVY, B. 2008. N-symmetry direction field design. *ACM Trans. Graph. 27*, 2, 1–13.

SPRINGBORN, B., SCHRÖDER, P., AND PINKALL, U. 2008. Conformal equivalence of triangle meshes. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, 1–11.

TONG, Y., ALLIEZ, P., COHEN-STEINER, D., AND DESBRUN, M. 2006. Designing quadrangulations with discrete harmonic forms. In *Proc. SGP*, Eurographics Association, 201–210.

ZHANG, E., MISCHAIKOW, K., AND TURK, G. 2006. Vector field design on surfaces. *ACM Trans. Graph. 25*, 4, 1294–1326.