Simple Quad Domains for Field Aligned Mesh Parametrization

Marco Tarini[1,2] Enrico Puppo[3] Daniele Panozzo[3] Nico Pietroni[1] Paolo Cignoni[1]
[1]ISTI-CNR, Italy [2]Università dell'Insubria, Italy [3]Università di Genova, Italy

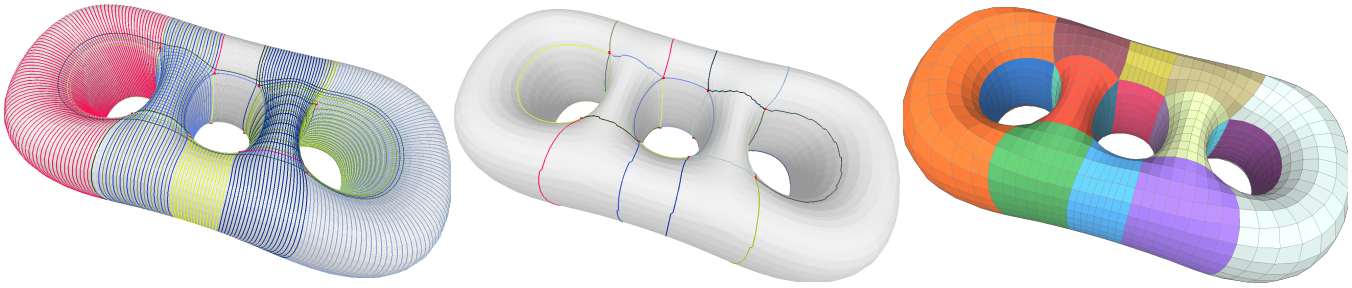

Figure 1: *(Left) An input mesh of quads induces a cross field with an entangled graph of separatrices defining almost eight thousand domains; (center) the graph is disentangled with small distortion from the input field to obtain just twenty parametrization domains; (right) parametrization is smoothed to make it conformal; an example of remeshing from the parametrization.*

Abstract

We present a method for the global parametrization of meshes that preserves alignment to a cross field in input while obtaining a parametric domain made of few coarse axis-aligned rectangular patches, which form an abstract base complex without T-junctions. The method is based on the topological simplification of the cross field in input, followed by global smoothing.

CR Categories: I.3.5 [Computer graphics]: Computational geometry and object modeling—Curve, surface, solid and object repres.

Keywords: quad mesh, mesh parametrization

Links: DL PDF WEB

1 Introduction

Finding a high-quality parameterization $f : D \rightarrow M$ for a given 3D polygonal mesh M is a prerequisite in a number of applications, such as quad-based semiregular remeshing, texture mapping, compression, fitting high order surfaces, physical simulations, tangent space geometry processing, and even tasks outside computer graphics, like physical modeling with metal sheets.

The definition of quality for f depends on the application, but usually encompasses criteria like injectivity, isometry (implying angle preservation and area preservation), smoothness (continuity of gradient vectors), and alignment of gradient vectors with geometric features of M. This problem has been addressed with a wide arsenal of tools [Hormann et al. 2008] and good automatic results are becoming increasingly common.

A recent trend is to first define a cross field $\mathcal{C}$ over M, and then to find f such that its gradient vectors match $\mathcal{C}$ as much as possible. Interestingly, each of the criteria above can be redefined in terms of desired properties of $\mathcal{C}$. Thus the task is shifted from the definition of a good parameterization to the definition of a good cross field $\mathcal{C}$ for a given M. High quality parametrization can be obtained following this approach [Ray et al. 2006; Bommes et al. 2009]. It is now appearent that the definition of a good cross field $\mathcal{C}$ implies, among other things, the good placing of a few irregular points (a.k.a. cone singularities). Irregular points tend to be needed, for example, in places where M exhibits high Gaussian curvature.

In this paper, we focus on an important *additional* criterion for the quality of f, which we refer to as the *simplicity* of domain D (see below for an informal definition). Simplicity determines how much a parametrization f will be effectively useful in most applications, just as much as the other criteria listed above. As we will show, a cross field $\mathcal{C}$ designed to satisfy all the above conventional criteria, but not simplicity, will usually fail producing an acceptably simple domain. Still, it is often the case that a slightly modified cross field $\mathcal{C}'$ exists, which is able to generate a parametrization f with a dramatically simpler domain, while preserving to a large extent the other qualities of $\mathcal{C}$. This work presents a way to obtain the cross field $\mathcal{C}'$, given $\mathcal{C}$.

1.1 Objective: Domain Simplicity

For topologies of M other than the disk, the domain D must necessarily include discontinuities (a.k.a. cuts, or seams): two infinitesimally close points m_0 and m_1 of M lying of different sides of the cut may be mapped by f^{-1} to arbitrarily distant positions d_0 and d_1 of D. The values d_0 and d_1 are often constrained to be reciprocally associated with a "transition function" associated to that cut.

Simplicity of domain D is a concept encapsulating: how many discontinuities are needed (the fewer, the simpler); how simple the discontinuity lines are in D (e.g., straight axis-aligned lines are simpler than curved or jagged lines); and also how constrained and straightforward the transition functions are (the more constrained, the simpler). For example, a domain D consisting of a single flat unit square, with no seams, would exhibit the maximal possible domain simplicity (possible only for disk-like M). On the other

extremum, a 2D packing of all single faces of M, each one laid separately, would be so complex to make parametrization useless for any practical purpose. When, like in our case, D is defined as a collection of patches separated by cuts (also known as an atlas), domain simplicity means to have fewer and more regularly shaped patches, properly aligned and connected by simple transitions.

In our work, we consider the use of a domain D consisting of a collection of n integer sized, axis-aligned rectangular patches $D_0, D_1, \cdots, D_n$ (see Sec. 4). Rectangles have a side-to-side adjacency relation, mapping the *entire side* of a rectangle D_i to the *entire side* of another rectangle D_j, (i.e., there are no "T-junctions"), thus making the transition between them straightforward. For small n, this type of domains can be considered extremely simple, allowing, for example, straightforward applications to tasks like regular remeshing at arbitrary resolutions, texture mapping, GPU based computations in parametric space, and so on.

For a parametrization f strictly following a cross field $\mathcal{C}$, the simplest domain of this kind is determined by cutting M along the separatrices connecting the singularities of $\mathcal{C}$. Such separatrices define a graph G embedded on M, which may contain crossing nodes. Even when $\mathcal{C}$ has relatively few singularities and separatrices connecting them, graph G can contain many crossing nodes and, thus, it can induce a very high number of patches. This happens because separatrices will often make long tours, crossing other separatrices (or even themselves) a very large number of times. See the left side image of Fig. 1 for an example.

1.2 Main contributions

The main contributions of this paper are:

1. A new general algorithm for simplification of the graph of separatrices G generated by a cross field $\mathcal{C}$. A modified cross-field $\mathcal{C}'$ is trivially induced from the simplified graph of separatrices. Notwithstanding the dramatically simpler graph that it generates, $\mathcal{C}'$ is similar to $\mathcal{C}$ (see Sec 3).
2. A practical way to implement the above algorithm on a semi-regular quad-mesh Q, taken in input as a way to describe $\mathcal{C}$: edges of Q are aligned with directions of $\mathcal{C}$, and irregular points of Q correspond to singularities of $\mathcal{C}$ (see Sec 3.8).
3. A new kind of parametrization domain D, consisting of a collection of axis-aligned 2D rectangles with predefined side-to-side manifold connectivity. This type of domain arises naturally from a graph G induced by $\mathcal{C}$, and it exhibits an high degree of simplicity by construction. Global smoothing of an existing parametrization defined over D, and quad remeshing (see Sec. 4).

Contributions 1 and 3 could be adopted independently in different scenarios. In the following, we integrate all three contributions in a pipeline, ultimately aimed at parameterizing a semi-regular quad mesh over a simple domain.

1.3 Overview

Our pipeline consists of the following phases:

1. Input: an initial semi-regular quad-based domain Q;
2. A graph of separatrices G is trivially extracted from Q (Fig. 1, left);
3. Graph simplification: a new "disentangled" graph of separatrices G', defined over Q, is constructed from G (Fig. 1, center);
4. A "simple" abstract domain D is constructed from G', as well as an initial parametrization $f : D \rightarrow Q$ (see Sec. 4.4);
5. Global smoothing: parameterization f is globally smoothed (see Sec. 4.4).

A new semi-regular remeshing (or embedded domain) Q' can be found by applying f to a regular sampling of D. Mesh Q' is naturally partitioned into few rectangular regions corresponding to the various faces of D. See the image on the right side of Fig. 1.

2 Related work

Mesh parametrization has been studied thoroughly in the literature (e.g. see [Hormann et al. 2008] for a survey). Here we consider just those global parametrization works addressing simplicity of domain.

Geometry Images [Gu et al. 2002] map a whole triangular mesh onto a squared parametric domain. However, this mapping is possible just for surfaces of genus zero, and mapping can be affected by a large amount of distortion. Distortion can be reduced with Multichart Geometry images [Sander et al. 2003] where the domain is decomposed into a set of irregularly shaped flat patches. Because of their irregular borders, multi-chart geometry images have a complicate handling of discontinuities. Polycube-maps [Tarini et al. 2004] produce a seamless parametrization by projecting the geometry onto the faces of a polycube embedded in 3D space. Polycube mapping provides a very compact yet simple representation, and interpolating parametric positions involves just a simple 3D reprojection operation. Although techniques for the automatic generation of Polycube-maps have been proposed [Lin et al. 2008], generation of high quality embeddings still remains a manual task.

Methods based on mesh simplification [Lee et al. 1998; Khodakovsky et al. 2003; Pietroni et al. 2010] automatically produce a parametrization domain composed of a set of equilateral triangles, producing very coarse domains. However they do not take into account alignment of parametrization to shape features or to a given cross field. Moreover, they cannot be used to produce a quadrilateral remeshing. Other recent methods [Dong et al. 2006; Daniels et al. 2009] can produce simple parametric domains composed of a set of adjacent quads. These techniques do not allow control over the quad alignment. Spectral surface quadrangulation has been extended in [Huang et al. 2008] to take into account alignment to geometric features. However, methods based on Morse-Smale complexes force the size of quad patches to be uniformly determined by the underlying field. For this reason, as noted in [Huang et al. 2008], it is difficult to deal with cases with close feature lines.

Most recent approaches use a precomputed quadrilateral feature aligned quadrilateral mesh, either directly modeled by a human, or computed with methods such as Quadcover [Klberer et al. 2007], PGP [Ray et al. 2006], Mixed Integer [Bommes et al. 2009], or Standing Wave [Zhang et al. 2010], as a base to gather a simplified parametric domain. Meshes generated automatically have a high quality, but they usually contain far too many quads to be used as parametrization domains for practical applications. Such meshes may in fact provide an input for our method.

Motorcycle Graphs [Eppstein et al. 2008] partition a quadrilateral mesh into a set of quadrilateral patches by allowing T-junctions. A similar idea has been further exploited in [Myles et al. 2010] to improve simplicity of the domain. This method exhibits a great degree of adaptivity, i.e patches can vary noticeably their size over the surface to conform to details at different scale. However the presence of T-junctions complicates the structure of the parametric domain

and may hinder its use for several applications (see additional materials for a deeper discussion and comparisons).

Very recently, a method has been presented in [Bommes et al. 2011], to simplify the structure of a quadrilateral mesh while preserving its alignment. The optimization method is based on an extension of the polycord collapse operations [Daniels et al. 2008] and consists of a greedy application of grid preserving operators directly on the quad mesh, aimed at removing helical configurations. Similarly to what we propose here, this method has also the effect of simplifying the graph of separatrices induced by the quad mesh. While the method of [Bommes et al. 2011] manipulates the underlying quad mesh, our method is defined in general for cross fields, and it works directly on the graph itself, without changing the mesh (see additional materials for further details).

Finally, the topology of vector and tensor fields has been studied in the context of flow visualization techniques [Delmarcelle and Hesselink 1994]. In order to avoid visualization cluttering, many approaches for simplifying the topology of fields have been proposed, like for example [Tricoche et al. 2001; Chen et al. 2008]. Our approach faces this task from a totally different (and complementary) perspective: instead of targeting removal of critical points, we focus on the problem of simplifying the graph of separatrix lines.

3 Cross-field topology simplification

In this section, we describe an algorithm to "disentangle" the graph G of separatrices induced by an input cross field $\mathcal{C}$, producing a simpler but still consistent graph G'. This implicitly produces a modified cross field $\mathcal{C}'$ that has G' as separatrix graph. We strive to keep the differences between $\mathcal{C}'$ and $\mathcal{C}$ small, even if G' is dramatically simpler than G. Specifically, $\mathcal{C}'$ and $\mathcal{C}$ share identical irregular points (with the same indices). In terms of parametrization, as discussed, this means that we trade some alignment for a simpler domain topology.

3.1 Preliminaries

Let $\mathcal{C}$ be a smooth cross field defined on a 2-manifold M. For all general definitions and properties about cross fields, we refer to [Ray et al. 2008]. We assume that $\mathcal{C}$ has only a finite set S of isolated *singularities*. Each point of M, which is not a singularity of S, will be said to be *regular*.

A *streamline* of $\mathcal{C}$ is a line on M that is tangent/orthogonal to the directions defined by $\mathcal{C}$ at each point. A streamline with endpoints at singularities is called a *separatrix*. Field $\mathcal{C}$ is regular within each patch, i.e., patches are actually "gridded" by the streamlines of $\mathcal{C}$.

For a singularity of index $\frac{k}{4}$, there are exactly $k-4$ incident separatrices. The network of separatrices is a graph embedded on M, describing the topology of $\mathcal{C}$. Separatrices cross at a finite set X of regular points of M, called the *crossing nodes*. Only two (possibly not distinct) separatrices cross at each node. Crossing nodes subdivide separatrices into *arcs* of a set E. The planar graph $G = (V, E)$, where $V = S \cup X$ subdivides M into quadrangular patches.

For each singularity $v \in S$, let $s_1, \ldots, s_k$ be the separatrices incident at v, and let $\mathbf{t}_{v,s_i}$ be the unit tangent vector of separatrix s_i at v, pointing outwards v in the direction of s. Each vector $\mathbf{t}_{v,s_i}$ is called a *port* of v.

By design, the simplified graph $G' = (S \cup X', E')$ will have the same set of irregular points, ports, and separatrices of the original graph $G = (S \cup X, E)$. The objective of the simplification is the reduction of the number of crossings $|X'|$ and, thus, of edges $|E'|$.

3.2 Graph energy

Given the graph G of separatrices, we aim at obtaining another graph $G' = (V', E')$ such that $V' = S \cup X'$, with $|X'| < |X|$ and having the same features of G: nodes of S maintain the same ports; nodes of X' are regular; patches induced on M are quadrangular; separatrices are smooth lines defined by chains of arcs.

We introduce a measure of *"drift"* and *"extension"* for a line l on M. Drift $\delta(l)$ measures the misalignment of l with respect to $\mathcal{C}$. Extension $\eta(l)$ measures the length covered by l following $\mathcal{C}$.

Assume l is parametrized by arc-length: $l : (0; \lambda(l)) \to M$, where $\lambda(l)$ is its total length. Let $\mu(t)$, $t \in (0; \lambda(l))$, be the unit vector of $\mathcal{C}(l(t))$ - one out of four - which is closest to $\nabla l(t)$, and $\theta(t)$ be the angle between vectors $\mu(t)$ and $\nabla l(t)$. Then $\delta(l)$ and $\eta(l)$ are defined as:

$$\delta(l) = \int_l |\sin(\theta(t))| dt, \quad \eta(l) = \int_l \cos(\theta(t)) dt,$$

We define the energy of a graph as the sum of the energies associated to all its separatrices. We define the energy of a separatrix s as weighted sum of its extension and its drift:

$$k\delta(s) + \eta(s)$$

with a parameter k setting the relative importance of the two terms. The first term forces the graph to follow the original field $\mathcal{C}$. The second term is minimized when shorter separatrices are used, which implies a simpler graph with less crossings. We could penalize directly the (discrete) number of crossings, instead of the separatrix estensions. However, the two terms of energy as defined above use the same unit of measure (metric length), thus making k independent of rescaling. Parameter k is the *only* one in our framework, and it balances the need to preserve the initial field $\mathcal{C}$ with the amount of simplification. Empirically, we found a good value of k to be 5, which was used in all our experiments.

3.3 Graph reduction algorithm

In the initial graph G, which follows $\mathcal{C}$ exactly, there is no drift, and the total energy is given by the sum of the extensions of the separatrices, which for each separatrix amounts to its (geodesic) length. Our algorithm follows a greedy strategy, trying to reduce the energy of the graph by substituting some of its separatrices with different lines, while maintaining its topological structure consistent. The algorithm performs a sequence of simplification *cycles*, each consisting in a sequence of *moves*: an *opening* move; a sequence of (zero or more) *continuation* moves; and a *closing* move. Each move is composed of at most two sub-operations: an opening move consists of a deletion sub-operation; a continuation move consists of a deletion sub-operation, immediately followed by a creation sub-operation; a closing move consists of a creation sub-operation.

The **deletion sub-operation** is simply the removal of a separatrix. One *open port* is created at each end, i.e., two ports remain with no associated separatrix. A graph is consistent only if it has no open ports.

The **creation sub-operation** connects two given open ports p_0 and p_1 with a new separatrix, thus closing both. The new separatrix is plotted over the surface: it starts from from p_0, it may cross other separatrices (but never any singularity), and it ends at p_1. At both ends, the new separatrix matches the tangent directions of the open ports it connects to.

Some constraints must be fulfilled when the new separatrix l is traced, to ensure consistency. We impose the drift to be monotonic,

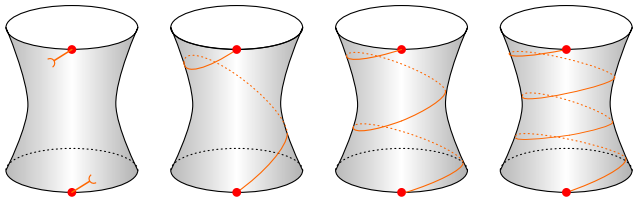

Figure 2: *Two open ports (left) can be connected with several different separatrices traveling in the same corridor: the separatrix minimizing the energy is selected (depending on the field, not necessarily the shortest one).*

i.e., such that $\theta(t)$ is either positive (right drift) or negative (left drift) along the whole line. We also impose $|\theta(t)| < \pi/4$, posing a limit on the accepted pointwise drift of a separatrix. This means that $\mu(t)$ can only change with continuity over l (in other words, l is not allowed to switch the direction of the field it is aligned to). Fianlly, two separatices can cross only if their two vectors μ at the crossing point are orthogonal.

If a separatrix fulfilling these requirements cannot be drawn to connect the two open ports, then the move is rejected. There can be several valid ways to connect, even very different from each other (e.g. see Fig. 2): the one with the least associated energy is selected.

3.4 Selecting moves

A cycle starts with a consistent graph without open ports. The opening move creates two open ports; each continuation move creates two more open ports (separatrix deletion), but immediately closes other two (separatrix creation), so that a total of two ports remain open during the entire cycle; the closing move closes the two open ports, bringing the graph back to a consistent state.

A deletion sub-operation necessarily decreases the total energy of the graph (removing the contribution of the deleted separatrix), whereas a creation sub-operation necessarily increases it (adding the contribution of the new separatrix). Therefore an opening move always decreases energy, and a closing move always increases energy (but it is necessary to close the cycle). A continuation move changes the energy of the difference between the deleted and the created separatrix energies, which can be positive or negative.

At every step, we list and evaluate all potential moves.

- When there is no open port (i.e. at the beginning of each cycle), potential moves are opening moves: there is one for each separatrix (deleting it).
- When two open ports are present, there can be several potential continuations moves, and sometimes a potential closing move too. They are all detected, as explained in Sec. 3.5.

In any case, the effect of each potential move on energy is evaluated. A potential move is considered valid only if, after performing it, the *total* change of energy of the current cycle remains strictly negative. This guarantees that overall energy is decreased by the cycle.

If a valid closing move is available, it is always preferred over any other move (the closing move leaves the graph consistent, so that a new cycle can be started by selecting an opening move again). If no valid closing move is available, we choose the move, among all possible valid ones, that decreases the energy the most (which sometimes can be an energy increasing move). If no valid move is available, a backtracking mechanism is triggered (see Sec. 3.6).

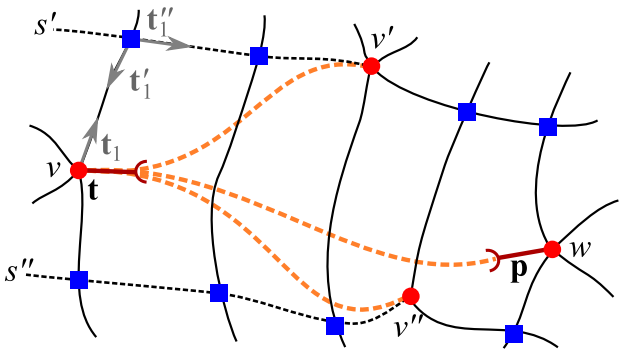

Figure 3: *The corridor stemming from an open port $\mathbf{t}$ at v: $\mathbf{t}$ can be connected with singular nodes v' (or v'') on the walls, with a continuation move, which would first delete separatrix s' (or s''), shown dotted in black, then connect $\mathbf{t}$ with v' (or v''), with the top (or bottom) orange dashed line. In case the other open port $\mathbf{p}$ at w is reachable along the corridor, then a closing move is possible, which connects $\mathbf{t}$ with w (with the middle orange dashed line). Crossings are shown as blue squares.*

3.5 Enumerating continuation and closing moves

Let $\mathbf{t}$ be an open port (see Fig. 3). In order to maintain a consistent structure of graph G', any new separatix starting at $\mathbf{t}$ must necessarily be contained in a *corridor* bounded by two chains of arches, which we term the *walls*. The left wall of the corridor is defined as follows (refer to Fig. 3). Let $\mathbf{t}_1$ be the port next to $\mathbf{t}$ by rotating counterclockwise about its node, let $\mathbf{t}_1'$ be the port opposite to $\mathbf{t}_1$ on the same arc, and let $\mathbf{t}_1''$ be the port next to $\mathbf{t}_1'$ by rotating counterclockwise about its node. The left wall starts at $\mathbf{t}_1''$; it continues at each next node by skipping one port in counterclockwise order, and taking the next port; and it stops when reaching either the node of $\mathbf{t}$ or the node of $\mathbf{t}_1''$. The right wall of the corridor is defined analogously (rotating clockwise).

Ports skipped at intermediate nodes along the walls connect opposite walls through transversal arcs. If the other open port $\mathbf{p}$ belongs to a vertex w lying along one of these arcs, and $\mathbf{p}$ is directed towards the beginning of the corridor, then a potential closing move which connects $\mathbf{t}$ to $\mathbf{p}$ is reported as possible. In this case, the end wall of the corridor is made of the two transverse arcs emanating from $\mathbf{p}$. Otherwise, the corridor is circular and it ends at one of the two transverse arcs emanating from $\mathbf{t}$ itself.

For each port $\mathbf{p}$ stemming from singular nodes that lie on walls and point towards the beginning of the corridor, there is a possible continuation move that connects $\mathbf{t}$ to $\mathbf{p}$. This move consists in the deletion of the separatrix currently starting from $\mathbf{p}$, followed by the creation of a separatrix connecting port $\mathbf{t}$ to $\mathbf{p}$.

That procedure is repeated starting from each of the two open ports, finding the set of all potential continuation (and possibly closing) moves.

Recall the delta of energy associated to a potential move is computed as the difference between the energy associated the separatrix to be plotted (except opening moves), minus the energy associated to the separatrix to be deleted (except closing moves). To evaluate the former, we need to compute drift and the extension of the separator s' which would be plotted. The latter is quickly evaluated recording the drift and extension of each separatrix.

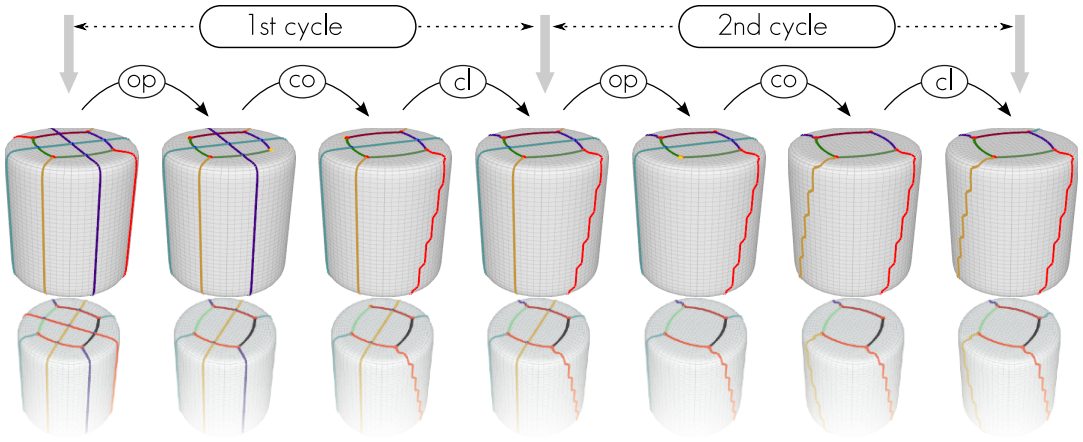

Figure 4: *A simple example of the graph simplification algorithm in action. A floor reflecting the surface is shown to make the lower part visible. The algorithm starts from a cross-field with eight singularities of index $1/4$, for a total of 24 ports, connected by 12 separatrices, which cross 10 times and divide the surface into 16 rectangular patches. Valid configurations are pointed by gray arrows. The algorithm performs two cycles: each cycle starts with a opening move ("op"), a single (in this case) continuation move ("co"), and finishes with a closing move ("cl"). In this specific case, the algorithm removes all the crossings, producing a graph where only 6 rectangular patches are needed to cover the object. A third cycle (not shown) is then attempted, in which a total of 54 moves are performed and eventually rolled back as they do not lead to a configuration with lower energy. The algorithm then returns the rightmost configuration.*

3.6 Exploring the space of solutions

If no valid move is available, i.e. when not even the best possible move would result in a strictly negative overall energy change for the current cycle, then the last performed move is rolled back. The cycle continues by picking the *next* best potential move at that configuration, if there are any valid ones left. Else, another rollback is performed, and so on.

Iterating this simple strategy is equivalent to a depth-first visit of the tree of possible states reachable by valid moves: the root is the consistent graph at the beginning of the cycle, intermediate nodes are inconsistent graphs, leaves are consistent graphs with less associated energy, and links are valid moves.

This search for a closing move for the current cycle can be either successful or not. If a valid closing move is found, then it is performed and the current cycle is over. The valid graph reached in this way is necessarily different, and it has a strictly smaller energy, than before. After that, a new cycle can be started.

Conversely, if the search fails (this happens when there are no opening moves left at the root of the tree) the algorithm is over, and the current graph (which is consistent) is returned as the final result.

The non-negative energy reduction constraint serves as a heuristic pruning of the tree during the search. In theory, the number of reachable nodes of the tree is still gigantic, but a dynamic programming approach makes the search feasible. A graph with n ports can connect them, pairwise, in $\frac{n!}{(n/2)!2^{n/2}}$ different ways (the vast majority of which is not consistent or not reachable). We hash each such configuration, and we reject any move that would produce a configuration already seen, during the simplification process, with a lower or equal associated energy.

The algorithm is greedy and it does not give any guarantee of returning the best configuration, but in practice it dramatically improves over input configurations, unless they are optimal.

3.7 Crease preservation

For surfaces representing mechanical objects, it can be important that separatrices pass *exactly* along sharp creases. In our algorithm, it is easy to prevent losing separatrices which are aligned to creases, simply by tagging them and disallowing any opening or continuation move which would remove them. This is a conservative option, but it reduces the degrees of freedom of the algorithm, resulting in a less simplified graph (see for example Fig. 5, second image).

Another viable strategy is to let crease separatrices be lost in this phase, and then force nearby separatrices to snap into their place during the smoothing phase (see rest of Fig. 5, and Sec. 4.5).

3.8 Implementation on semi-regular quad meshes

We show a practical implementation of the graph reduction algorithm which takes as input a semi-regular quad mesh Q, which implicitly provides a discretized cross field $\mathcal{C}$. Edges of Q represent directions of $\mathcal{C}$. Irregular vertices of Q represent singularities of $\mathcal{C}$, and edges stemming from them are their ports. For simplicity, we further assume that, as it commonly results from remeshing algorithms, all edges of Q have approximately the same length, which we will use as the unit length (this is not an intrinsic limitation: implementation for a more general setting would only require summing the lengths of traversed edges, rather than just counting their number). In this setting, the algorithm described above can be implemented in a simple and efficient manner.

Each separatrix s is composed of a sequence of edges of Q. The initial graph of separatrices G can be extracted trivially from Q by tracing all chains of edges stemming from irregular vertices. Any drifting separatrix generated during graph reduction is stored as a jagged sequence of edges (see Fig. 6). This choice can at first look problematic, as separatrices will not be smooth lines in general. However, it will be easy to remove these artifacts in a subsequent smoothng phase (see Sec. 4), taking advantage of the simplicity of global parametrization domain.

Length and drift of a separatrix can be computed by a simple count of the edges agreeing and disagreeing with the reference direction

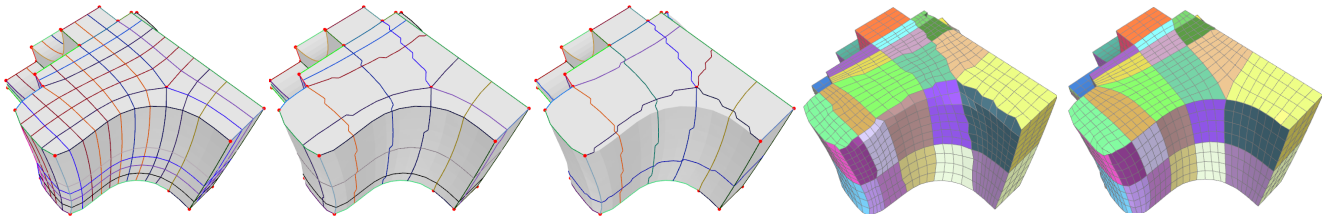

Figure 5: *Preservation of creases. From the left: initial graph; graph simplified by hard constraints on creases; graph can be simplified more without constraints, but some creases are lost; related domains just after simplification; final domain of parametrization: creases lost during unconstrained simplification can be recovered through snapping during the smoothing phase.*

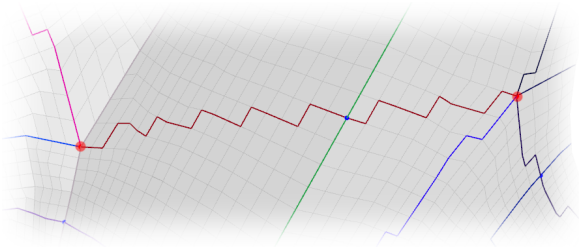

Figure 6: *On a quad mesh, a drifting separatrix s' of G' consists of a chain of either field-aligned or trasverse edges of Q. The one depicted here has a value of extension of 24 units, and a value of drift of 8 units.*

of the starting port. The constraint on pointwise drift of a separatrix (see Sec. 3.3) translates into a prohibition of having two consecutive disagreeing edges.

The operations described in Sec 3.5 to find candidate ports to be connected with open ports can be easily implemented by navigating over Q along edges. The length and drift, and therefore the energy, of the corresponding potential new separatrix can be computed during the same process.

4 Parameterizing over the abstract quad-mesh domain

A graph of separatrices G over mesh Q partitions the surface of Q into rectangular patches. Each patch can be easily parameterized over a flat, axis aligned rectangular domain D_i. Let D be the collection of all $D_1, D_2, \cdots D_n$. D will serve as the domain of our final parametrization $f : D \rightarrow Q$.

This section discusses the properties of D and f, and shows the operations that can be performed over them, including global smoothing. These operations have the following aims: removing jagged artifacts introduced by the algorithm in 3.8; computing a globally smooth parametrization over Q (whereas tracing separatrices only defines it along these lines); optimizing the placement of singularities and cross points (to comply with the new graph); optionally, recovering lost feature lines.

In this section it is easier to consider the triangle mesh M obtained from Q by diagonal splits and a corresponding parametrization $f : D \rightarrow M$. Since the connectivity of Q is unaffected by all operations, its original connectivity can be recovered at any time simply by removing the extra edges of M. If Q was in turn computed as a quad re-meshing of an initial mesh M_o, then in this smoothing phase the original mesh M_o can be used instead.

4.1 Properties of the parameterization and its domain

A position $p \in D$ consists of a tuple $p = (k, \alpha, \beta)$, k being the index of a rectangle D_k in D, and α β being the Cartesian coordinates of p inside $[0, w_k] \times [0, h_k]$, where w_k and h_k are the width and height of D_k, respectively.

Rectangles $D_0, D_1, ..., D_n$ in D are logically connected edge-to-edge: each of the four edges of each D_i is shared with exactly one other edge of some D_j (i and j being not necessarily distinct), and the two edges are constrained to have the same length: this means that the frame of D_j can be rotated (by a multiple of $\pi/2$) and translated to become the continuation of the frame of D_i. Likewise, each corner of each D_i is considered to be logically coinciding with a certain number of corners of other rectangles. We call the domain D abstract in the sense that, even if it has the logical connectivity of a manifold quad mesh, it is not embedded in a 3D Euclidean space.

As commonplace, parametrization $f : D \rightarrow M$ is defined by discretely sampling its inverse ϕ, e.g., by explicitly assigning a parametric position $p_i = \phi(v_i) \in D$ to each vertex v_i of M. This per-vertex assignment can be propagated, by interpolation, over the entire mesh M. In fact, as we will show (Sec. 4.2), D allows for interpolations of positions, even among different rectangles. The only necessary assumption is that, for a triangle t in M, its three vertices are mapped into rectangles of D with at least one common coinciding corner (i.e., t does not span an excessively large area in parametric domain). Note that we do not require any position p_i to be on the boundaries or at the corners of rectangles in D, meaning also that we do not require, in the final parameterization, that patch boundaries coincide with edges of M.

4.2 Interpolation domains

Positions in D lying in the same rectangle D_i can be interpolated linearly, as usual. For the purpose of defining interpolations among positions in D lying in different rectangles, two or more adjacent rectangles of D can be temporarily assembled into larger "interpolation domains". This approach follows the spirit of [Pietroni et al. 2010], where a similar concept is introduced for triangle-based domains.

We term "interpolation domain" E_i a 2D region where a set of k logically contiguous rectangles $D_{a_1}, D_{a_2}, \cdots D_{a_k}$ are mapped by a corresponding invertible function $g_{E_i} : \cup_{j \in (1..k)} D_{a_j} \rightarrow E_i$. These functions and their inverses will be defined in closed form and easy to evaluate. We use three kinds of interpolation domains.

An *"Edge"* interpolation domain E_i^e unifies two rectangles D_i and D_j of D, logically sharing an edge, into a larger rectangle. The associated function g_i is the identity for D_i, while, for D_j, it is the roto-translation which moves D_j so that the corresponding sides of the two rectangles coincide (see Fig. 8, middle left).

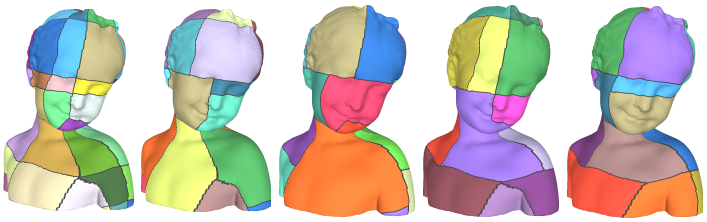

Figure 7: *Leftmost image: mesh M is partitioned into $f(D_1), (D_2), \cdots, (D_n)$ (each vertex v is coded according to domain $\phi(v)$, but triangles connecting fixed vertices are darkened). Other images: the same is repeated using different set of domains $E_1, E_2 \cdots, (E_m)$, partitioning of D. Note that each vertex of M is not a fixed vertices in at least one of the partitions.*

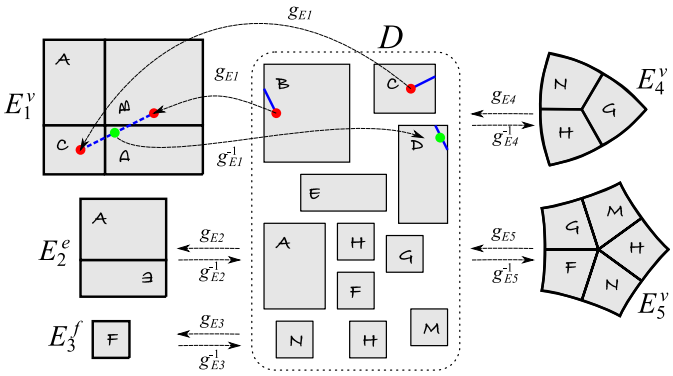

Figure 8: *Middle: a domain D composed of patches $D_1 \cdots D_n$, marked by calligraphic uppercase letters. Left, from top: an example of a Vertex, Edge and Face interpolation domain, and associated functions g. Right, from top: other examples of Vertex domain for vertices of D with valency 3 and 5 respectively. Top: an interpolation domain used to define the interpolation between the two red dots $\in D$. The result of the interpolation is the green dot in D.*

A *"Vertex"* interpolation domain E_i^v unifies all n patches sharing a given corner. In the regular cases, i.e., when $n = 4$, E_i^v is a rectangle (see Fig. 8 and the associated function g_i is simply made of appropriate rigid roto-translations of the various rectangles involved. For irregular vertices ($n \neq 4$), roto-translation is combined with an exponential map (with exponent $k = 4/n$) which is known to be conformal, and the interpolation domain is a star shaped region (see Fig. 8 right). Centering the origin in the shared corner and expressing positions p_i in polar coordinates (ρ, α), the exponential map is the function $(\rho, \alpha) \mapsto (\rho^k, \alpha \cdot k)$.

It will ease method description to also define a trivial *"Face"* interpolation domain E_i^f which is identical to a single patch D_i (see Fig. 8, bottom left); the associated function g_i is the identity.

There is exactly one Edge domain for each shared edge of D, one Face domain for each rectangle of D, and one Vertex domain for each shared corner of D. Any position $p \in D$ belongs to one Face domain, four Edge domains, and four Vertex domains.

An interpolation domain allows to interpolate, in parameter space, between any pair of positions $p_0, p_1 \in D$, as long as they belong to two domains D_0 and D_1 which share at least one vertex. First, one interpolation domain E_a (either an Edge, Face, or Vertex one) is selected such that both D_0 and D_1 are included in it. The interpolation can be then computed as $g_a^{-1}(I(g_a(p_0), g_a(p_1)))$, where I is the common linear interpolation operator (see Fig. 8 top for a graphical representation of this formula).

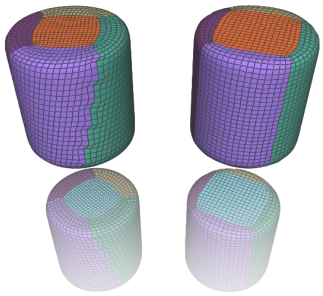

Figure 9: *Two semi-regular meshes obtained by resampling the parametric domain Left: parametric domain resulting from the simplification of Fig. 4: patches are separated by visibly jagged lines. Right: after global smoothing (see Sec. 4.4), the jagged lines are gone, and irregular points moved to aligned, optimized positions.*

We employ interpolation domains also for the task of globally smoothing an existing parameterization, as explained in Sec. 4.4.

4.3 Construction of an initial parameterization

After graph simplification (Sec. 3), we define the parameter domain D and an initial rough parameterization of M over it, which will then be globally smoothed to obtain the final parameterization (Sec. 4.4).

Recall that defining a parameterization over M means to assign to each vertex v_i of M a unique parametric position p_i in D, $p_i = (k_i, \alpha_i, \beta_i)$.

Initially, i.e. after graph simplification (Sec. 3), separatrices are defined as collection of edges of M, which partitions the faces of M into disjoint groups, each one surrounded by four arcs of the graph G. We create a rectangle D_i of D for each group of faces, and all the vertices of these faces are assigned to such a rectangle (vertices of M shared by faces belonging to different groups are arbitrarily assigned to any of these groups). This amounts to assign an initial k_i to each vertex of M. The connectivity among rectangles in D is likewise extracted from the graph of separatrices.

For any vertex of a face of M touched by a separatrix of G, we also assign an initial parametric position α_i, β_i close to the appropriate border of the rectangle D_{k_i}. The exact positioning is not crucial as the final values of α_i, β_i will be determined during the subsequent smoothing phase (Sec. 4.4).

For all other vertices of M, initial positions α_i, β_i are determined by applying a single-patch energy-minimization parametrization technique inside every patch. We use [Hormann and Greiner 2000], but many other methods could be used in its place. This method tends to produce conformal mapping and, due to the shape and size similarities between the D_i and $\phi(D_i)$, it also delivers a certain degree of isometry.

The dimensions of D_i in parametric space are determined using a separate procedure, described in Sec. 4.6.

4.4 Global smoothing of the parametrization

Once an initial parameterization is found, we perform a global smoothing of the parameterization. A common strategy to perform global smoothing over a domain with cuts is to define transition functions connecting the two sides of the cuts, and to embed them

into a single energy minimization system [Bommes et al. 2009]. While we have transition functions implicitly associated to pairs of shared sides of rectangles in D, we cannot adopt this approach because in our case we need the smoothing process to optimize which rectangle of D each vertex of M belongs to.

Global smoothing is done in a sequence of passes, in a "quincux" style algorithm. The idea is that, at each pass, some vertices are kept fixed while the positions of the remaining vertices are optimized, and passes are alternated in such a way that all vertices can undergo the smoothing process.

At each pass, we extract a different set of interpolation domains $\mathbb{S} = \{E_0, E_1..E_i\}$ which forms a partition D, i.e., such that each rectangle D_i belongs to exactly one interpolation domain of $\mathbb{S}$. Set $\mathbb{S}$ is determined with a simple heuristic. Starting from an empty set, a Vertex domain E_0^V is inserted into $\mathbb{S}$, only if is composed of rectangles of D not already included in any $E_k \in \mathbb{S}$. Once all Vertex domains have been scanned, this process is repeated for Edge domains. Finally, the isolated rectangles D_i still not included in any $E_k \in \mathbb{S}$ are inserted as Face domains. Before each pass, we keep a count, for each edge and each vertex of D, of the number of consecutive passes that element lies on the boundary of the interpolation domain embedding it. Such count is used to prioritize insertion of Vertex and Edge domains into $\mathbb{S}$. Fig. 7 shows an example of four consecutive sets $\mathbb{S}_0, \mathbb{S}_1, \mathbb{S}_2, \mathbb{S}_3$ chosen in this way.

For a vertex v_i with associated parametric position p_i inside D_j, the unique interpolation domain $E_k \in \mathbb{S}$ including D_i is identified, and a temporary position $p_i' \in E_k$ is found by $p_i' = g_k(p_i)$. If an edge of M connects two vertices v_i and v_j whose temporary positions p_i' and p_j' lie in different interpolations domains, then both p_i' and p_j' are marked as fixed for the current pass; then, inside each $E_k \in \mathbb{S}$, temporary positions of non-fixed vertices are smoothed, again using a single patch conformal energy minimizer [Hormann and Greiner 2000]. After the smoothing, vertices are remapped into D by the functions g_k^{-1} of respective interpolation domain, and a new pass is started. Note that this process can cause a parametric position p_i to change rectangle in D, if this reduces the global energy.

The process converges to a minimum of global conformal energy. The reason is that, at each pass, functions g and their inverse preserve conformal energy, and the smoothing operation monotonically decreases it.

By finding positions $p_i \in D$ associated to mesh vertices, the optimization process implicitly determines boundaries of patches over M (recall that we do not require p_i to lie on the boundaries of rectangles in D, nor patch boundaries to pass thorough edges of M).

Results of this smoothing process are depicted in Fig. 11 and 9.

4.5 Recovering lost feature lines

As mentioned in Section 3, during the graph simplification we can choose *not* to preserve creases in Q (and therefore in M). When this is the case, we can demand to the smoothing phase the task of realigning patch borders to the geometric features of M.

Specifically, this can be done when the parametrization is being optimized over an interpolation domain. Edges of M tagged as feature edges can be snapped, and then constrained to lie on a 2D straight internal line $l = g(e_k)$, e_k the appropriate edge of a rectangle in D (see Figure 5).

4.6 Isometry

Each 2D rectangle in the domain D is assigned to an extension in each of its two dimensions. The dimensions are chosen to maximize the isometry of mapping f, by solving a separate system. For simplicity, the isometry is approximated by summing the area weighted contributions of solely the triangles of M which are mapped entirely into a single rectangle of D. The system has two variables for the two sides of each rectangle, but rectangles logically connected side-to-side are constrained to have matching lengths in the respective dimension, reducing the number of variables in the system.

As mentioned, one of the possible uses of the final parameterization is to build a semi-regular quad-remeshing M' of M (vertices of M' are sampled over M at each integer coordinates inside rectangles of D). If this is the case, sizes of rectangles need be constrained to be integer numbers. This is enforced after solving the system, by multiplying each resulting dimension by a scalar w (determining the required density of the re-sampling) and rounding each dimension to the nearest non zero natural number. Value of w is picked inside a user determined interval so to minimize rounding errors.

The process of determining the dimensions (and thus the aspect ratios) of rectangles in D is interleaved with the smoothing passes (Sec. 4.4), because the portion of M mapped inside each patch of D can vary during smoothing.

5 Results

In this section, we show a gallery of results obtained on several datasets commonly used as benchmarks. Our input fields come from quad meshes either kindly provided by the authors of [Bommes et al. 2009] (drill-hole, fandisk, fertility, joint, rockerarm), or produced with an independent implementation of the same method (bimba, bunny, cube-blob, fertility-sym, holes3, kitten) and cover a spectrum of mechanical and natural objects, with simple and complicated shapes. Table 2 provides statistics on the datasets and results. For each input mesh we provide: number of facets; number of irregular and crossing nodes of the graph of separatrices and number of domains induced by this graph; number of crossing nodes and domains for the simplified graph. The number of patches is invariably dramatically reduced. This drastic simplification of the domain comes at an expense of a slight deterioration of parametrization quality, measurable in the loss of quad quality in remeshings, in terms of edge orthogonality and aspect ratios, as reported in Table 1. The entire process took between 20 seconds (holes3) and 10 minutes (fertility) using commodity hardware.

In Fig. 10 we show results of the simplification phase from two datasets also used in [Bommes et al. 2011], for comparison. In the drill-hole dataset, exactly the same input mesh has been used, and our result provides a simpler domain. In the rockerarm dataset, a finer mesh has been used, which provides a more entangled input field. In spite of that, our result also provides a simpler domain. In both cases, original alignment is preserved (Fig. 10 just depict the graph of separatrices: some lines are jagged because they are traced in a discrete way on the underlying mesh; jags are eliminated with the subsequent phase of smoothing, as shown in results from the same datasets in Fig. 14). More comparisons are available as additional materials.

In Fig. 11 we show the effect of smoothing on the bimba dataset. Fig. 14 shows results from the two phases of the algorithm on several other datasets. In Fig. 12 we show how sharp creases can be preserved by freezing them during simplification. Note that also sharp creases that were not captured by separatrices, like the border of the big hole in the left image, can be recovered during the smoothing phase thanks to the snapping mechanism. In Fig. 13 we show how results can be improved by providing a better placement of singularities of the cross field: the result shown on top is obtained from an input mesh returned by the Mixed Integer algorithm (dataset fertMI); the result shown on bottom is obtained

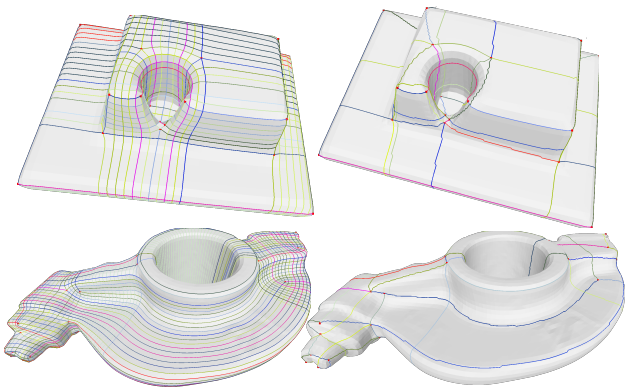

Figure 10: *Graph simplification: the input graph of separatrices may be more or less entangled, in all cases our graph reduction algorithm manages to dramatically reduce the complexity of the domain while preserving alignment to the input field.*

Model:	Avg. right angle discrepancy (deg)		Avg Min/Max Edge Ratio	
	input	output	input	output
rockerarm	5.3	4.9	0.83	0.68
holes3	4.8	4.0	0.81	0.60
bimba	7.3	10.2	0.79	0.64
fertility	5.9	7.8	0.78	0.60
drillhole	3.1	3.7	0.79	0.70

Table 1: *Original meshing quality vs. quality of remeshing.*

from an input mesh which has been computed by placing singularities of the cross field more symmetrically with respect to the shape (dataset fert-sym). In both cases, the initial graphs are extremely entangled (most edges belonging to separatrices). It is evident, especially from the top view, how a better placement of singularities allow us to obtain a smaller number of simpler domains, and a better remeshing (see also statistics in Table 2).

6 Concluding remarks

We have proposed a method for producing quad-based domains for global mesh parametrization, which improves domain simplicity, while maintaining alignment to an input cross field. The method has been implemented to take in input a cross field induced by a quad mesh - which may be already the base domain of a parametrization - and it produces a parametrization having an abstract complex of axis-aligned rectangles as base domain. Our results exhibit simpler domains than other proposals at the state-of-the-art and parametrizations are directly suitable for most applications.

The method consists of two main ingredients: an algorithm for simplifying the topology of the cross field, and an algorithm for smoothing parametrization across abstract quad domains. Even if we presented an implementation working on quad meshes (used as a discretization of the input cross field), the simplification algorithm is fully general, and could even be applied to any N-symmetry field [Ray et al. 2008], provided that an initial graph representing the topology of the field is given in input. However, finding such a graph in general, e.g., from a field defined on a triangle mesh, is still an open problem.

Our algorithm strives to preserve alignment with the input cross field (by penalizing drift on separatrices traced during simplification). The unique parameter used in the whole method (value k in the energy) allows us to trade off between topology simplification and faithfulness to the input field. Our method also allows preserving sharp creases, either through hard constraints during graph simplification, or through a snapping mechanism during smoothing.

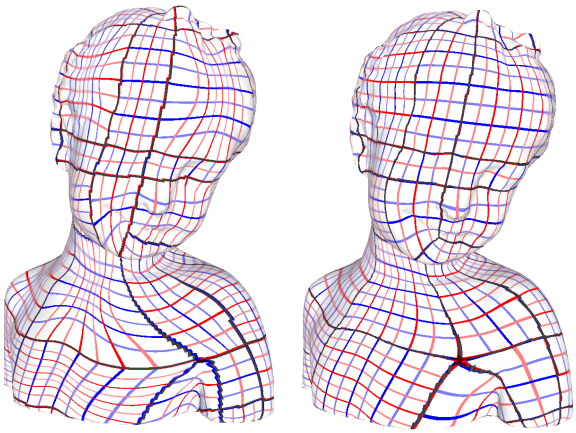

Figure 11: *Smoothing: an initial parametrization is computed by assigning vertices of the input mesh to domains induced from the simplified graph (left); parametrization is smoothed to make it conforming (right).*

One main limitation of the proposed approach is its reliance on the quality the pre-existing cross field $\mathcal{C}$. If $\mathcal{C}$ presents poorly placed, or too numerous singularities, the simplified graph will still be far too complicated to be useful in most contexts. However, in our experience, practically any graph is strongly improved, unless it is already optimal in terms of simplicity of the domain.

Acknowledgments

The research leading to these results is partly funded by the EU Community's FP7 ICT under the V-MusT.net Project (Grant Agreement 270404). We thank David Bommes, Denis Zorin, Ashish Myles and Carlos Hernández (www.tsi.enst.fr/3dmodels) for providing several datasets.

References

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2009. Mixed-integer quadrangulation. *ACM Trans. Graph. 28*, 3, 1–10.

BOMMES, D., LEMPFER, T., AND KOBBELT, L. 2011. Global structure optimization of quadrilateral meshes. *Computer Graphics Forum 30*, 2, 375–384.

CHEN, G., MISCHAIKOW, K., LARAMEE, R. S., AND ZHANG, E. 2008. Efficient morse decompositions of vector fields. *IEEE Trans. on Vis. and Comp. Graph. 14*, 848–862.

DANIELS, J., SILVA, C., SHEPHERD, J., AND COHEN, E. 2008. Quadrilateral mesh simplification. *ACM Trans. Graph. 27*, 5, 148:1–148:9.

DANIELS, J., SILVA, C. T., AND COHEN, E. 2009. Semi-regular quadrilateral-only remeshing from simplified base domains. *Computer Graphics Forum 28*, 5, 1427–1435.

DELMARCELLE, T., AND HESSELINK, L. 1994. The topology of symmetric, second-order tensor fields. In *VIS '94: Proceedings of the conference on Visualization '94*, 140–147.

Model:	Facets	Singular nodes	*Original Graph* Crossing nodes	Domain patches	*Simplified Graph* Crossing nodes	Domain patches
holes3 (Fig.1)	36487	16	7729	7749	0	20
cubeblob (Fig.14 top left)	9146	56	5546	5600	24	78
fertility (Fig.13 left)	3357	48	2217	2271	199	253
fertility symm. (Fig.13 right)	5785	46	4546	4598	79	131
kitten (Fig.14 mid left)	31198	30	31168	31198	49	79
bimba (Fig.7,11)	31618	26	31594	31618	58	82
bunny (Fig.14 bottom left)	35862	43	33938	33979	83	124
drill_hole (Fig.14 middle)	3077	26	1344	1368	58	82
joint (Fig.12)	8804	23	473	498	87	112
fandisk (Fig.5)	764	30	380	408	60	88
rockerarm (Fig.14 top right)	9413	36	4488	4524	62	98

Table 2: *Statistics on datasets and graph simplification.*

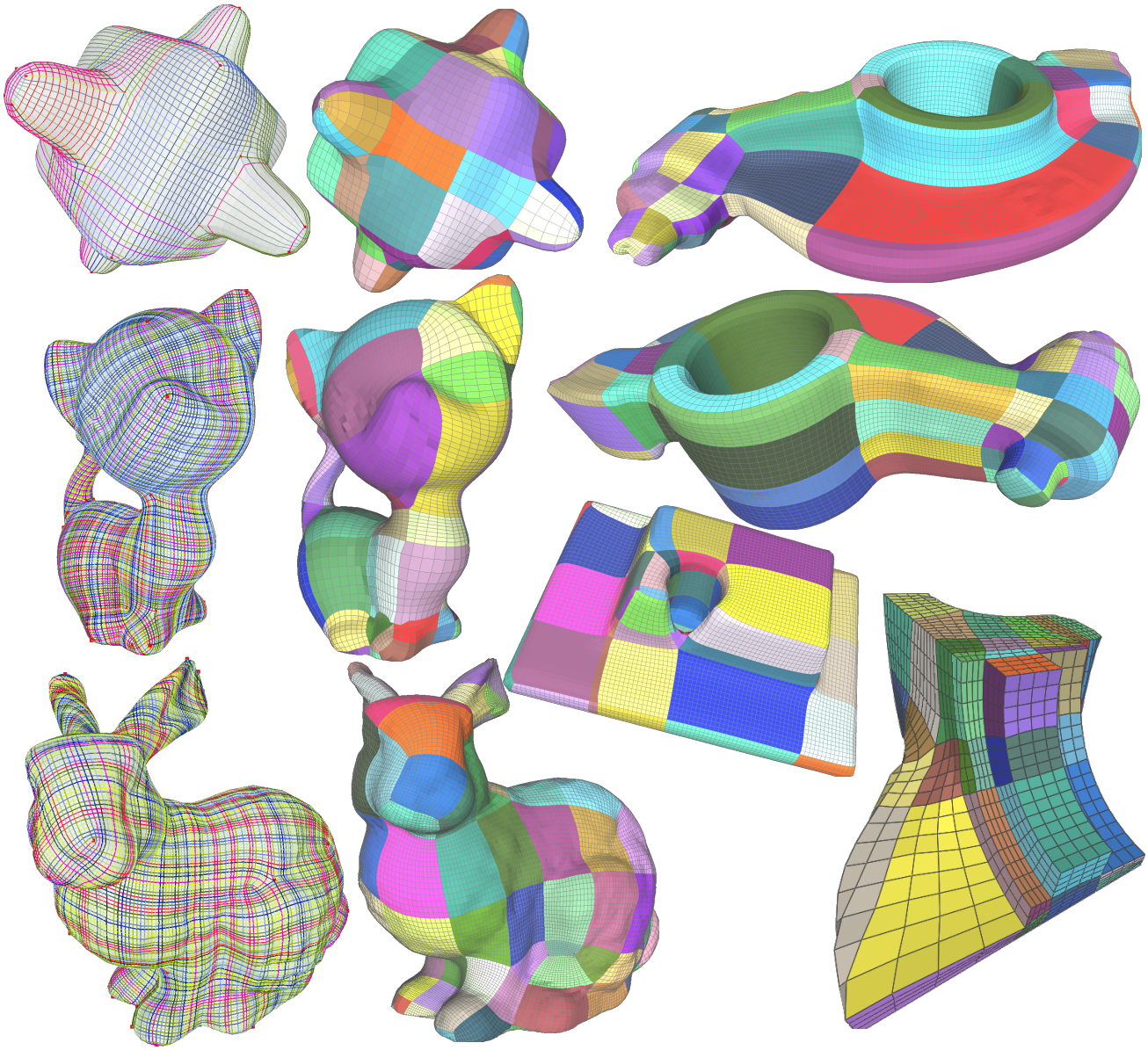

Figure 14: *Results of our algorithm on several datasets. For each dataset, the input graph of separatrices and a remeshing colored with faces of the resulting base domain are shown (input graphs are depicted in previous figures for some datasets).*

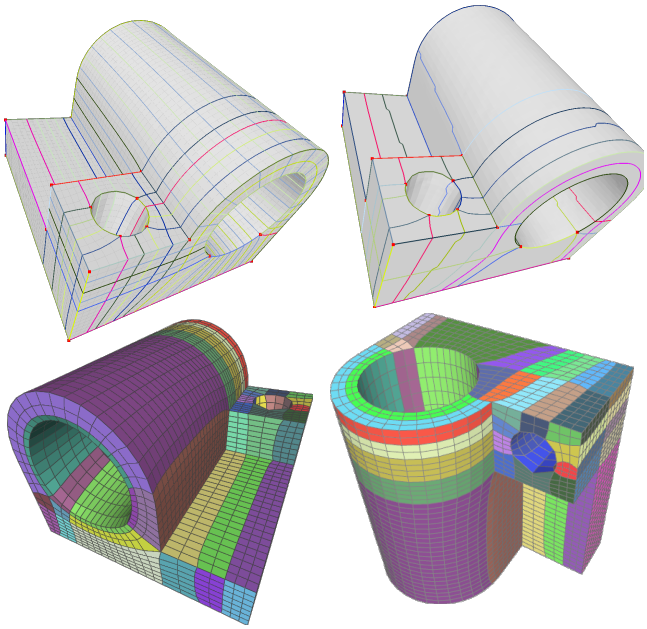

Figure 12: *Sharp features are preserved either by hard constraints during simplification (top left), or by snapping during smoothing (other three images).*

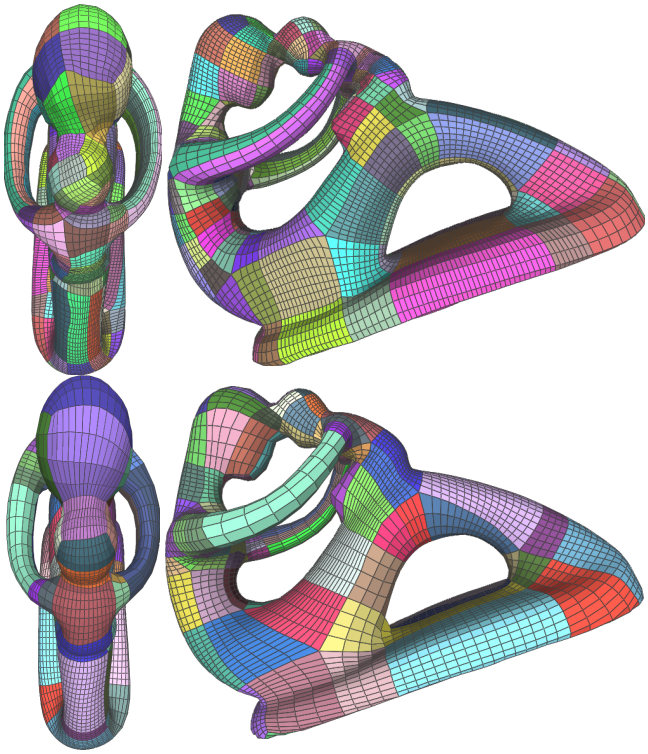

Figure 13: *Results obtained with Fertility model starting from a different layout of singularities. A better layout helps reaching a simpler domain (below) and thus a better remeshing.*

DONG, S., BREMER, P.-T., GARLAND, M., PASCUCCI, V., AND HART, J. 2006. Spectral surface quadrangulation. *ACM Trans. Graph. 25*, 3, 1057–1066.

EPPSTEIN, D., GOODRICH, M., KIM, E., AND TAMSTORF, R. 2008. Motorcycle graphs: Canonical quad mesh partitioning. *Computer Graphics Forum 27*, 5 (July), 1477–1486.

GU, X., GORTLER, S. J., AND HOPPE, H. 2002. Geometry images. In *SIGGRAPH*, ACM, T. Appolloni, Ed., 355–361.

HORMANN, K., AND GREINER, G. 2000. MIPS: An efficient global parametrization method. In *Curve and Surface Design: Saint-Malo 1999*, Innovations in Appl. Math. Vanderbilt Univ. Press, 153–162.

HORMANN, K., POLTHIER, K., AND SHEFFER, A. 2008. Mesh parameterization: Theory and practice. In *SIGGRAPH Asia 2008 Course Notes*, ACM Press, Singapore, no. 11, 1–81.

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, 147.

KHODAKOVSKY, A., LITKE, N., AND SCHRÖDER, P. 2003. Globally smooth parameterizations with low distortion. *ACM Trans. Graph. 22*, 3, 350–357.

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

LEE, A. W. F., SWELDENS, W., SCHRÖDER, P., COWSAR, L., AND DOBKIN, D. 1998. Maps: Multiresolution adaptive parameterization of surfaces. *Comp. Graph. Proc.*, 95–104.

LIN, J., JIN, X., FAN, Z., AND WANG, C. 2008. Automatic polycube-maps. In *Proc. of the 5th Int. Conf. on Advances in geometric modeling and processing*, Springer-Verlag, 3–16.

MYLES, A., PIETRONI, N., KOVACS, D., AND ZORIN, D. 2010. Feature-aligned t-meshes. In *ACM SIGGRAPH 2010 papers*, ACM, New York, NY, USA, SIGGRAPH '10, 117:1–117:11.

PIETRONI, N., TARINI, M., AND CIGNONI, P. 2010. Almost isometric mesh parameterization through abstract domains. *IEEE Trans. on Visualization and Comp. Graphics 16*, 4, 621–635.

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

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

SANDER, P. V., WOOD, Z. J., GORTLER, S. J., SNYDER, J., AND HOPPE, H. 2003. Multi-chart geometry images. In *Symp. on Geom. Proc.*, Eurographics Association, vol. 43 of *ACM Internat. Conf, Proc. Series*, 146–155.

TARINI, M., HORMANN, K., CIGNONI, P., AND MONTANI, C. 2004. Polycube-maps. In *ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, SIGGRAPH '04, 853–860.

TRICOCHE, X., SCHEUERMANN, G., AND HAGEN, H. 2001. Continuous topology simplification of planar vector fields. In *IEEE Visualization*.

ZHANG, M., HUANG, J., LIU, X., AND BAO, H. 2010. A wave-based anisotropic quadrangulation method. In *ACM SIGGRAPH 2010 papers*, ACM, New York, NY, USA, SIGGRAPH '10, 118:1–118:8.