[1]

Joint optimization of distortion and cut location for mesh parameterization using an Ambrosio-Tortorelli functional

Colin Weill–Duflos[1], David Coeurjolly[b], Fernando de Goes[c], Jacques-Olivier Lachaud[1]

[a] *Univ. Grenoble Alpes, Univ. Savoie Mont Blanc, CNRS, LAMA, 73000 Chambéry, France*
[b] *Univ Lyon, CNRS, INSA Lyon, UCBL, LIRIS, UMR5205, Villeurbanne, France*
[c] *Pixar Animation Studios USA*

Abstract

UV mapping is a classical problem in computer graphics aiming at computing a planar parameterization of the input mesh with the lowest possible distortion while minimizing the seams length. Recent works propose optimization methods for solving these two joint problems at the same time with variational models, but they tend to be slower than other cutting methods. We present a new variational approach for this problem inspired by the Ambrosio-Tortorelli functional, which is easier to optimize than already existing methods. This functional has widely been used in image and geometry processing for anisotropic denoising and segmentation applications. The key feature of this functional is to model both regions where smoothing is applied, and the loci of discontinuities corresponding to the cuts. Our approach relies on this principle to model both the low distortion objective of the UV map, and the minimization of the seams length (sequences of mesh edges). Our method significantly reduces the distortion in a faster way than state-of-the art methods, with comparable seam quality. We also demonstrate the versatility of the approach when external constraints on the parameterization is provided (packing constraints, seam visibility).

Keywords: Mesh parameterization, UV mapping, Variational model, Ambrosio-Tortorelli functional, Joint distortion and cut

1. Introduction

Computing a UV map for a mesh is a classical problem in computer graphics. UV maps have many applications, the most direct being texture atlas creation for a 3D model. Given an input three-dimensional discrete surface, the task addressed in this paper is to define a low distortion mapping of the mesh geometry to the plane while minimizing the cut length. Such cuts, or seams, may be required for topological considerations (for shapes that are not homeomorphic to a disk), or may help to better minimize the UV distortion (different cuts may lead to different local minima of the distortion energy). The problem of finding cuts and the problem of computing the parameterization are often addressed separately: cutting can be considered as a topological discrete problem, while optimizing the distortion is a geometrical problem usually solved with variational approaches.

Our method aims at simultaneously optimizing those two problems using a variational model, so that instead of predicting the distortion, the cut can adapt itself to the distortion during the process. There already exist works exploring this direction: Poranne et al. (2017) have proposed a variational method

[1]This work is supported by the French National Research Agency in the framework of the « France 2030 » program (ANR-15-IDEX-0002), by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) and by the StableProxies project (ANR-22-CE46-000).

Email addresses: `colin.weill-duflos@univ-smb.fr` (Colin Weill–Duflos), `david.coeurjolly@cnrs.fr` (David Coeurjolly), `fernando@pixar.com` (Fernando de Goes), `jacques-olivier.lachaud@univ-smb.fr` (Jacques-Olivier Lachaud) based on a per edge cut energy, Li et al. (2018) have presented several edge merging/cutting operations and have used them during the optimization process to reduce a given functional.

Our contribution consists of employing a dedicated variant of Ambrosio-Tortorelli functional to achieve this joint optimization. This functional measures the smoothness of a function while allowing discontinuities: in our context, the function will be the parameterization itself (with a parallel between smoothness and distortion) and the discontinuities will correspond to the locations of cuts. Minimizing this functional gives us a simple framework to simultaneously optimize the distortion and find the cuts faster than previous simultaneous optimization methods.

2. Related Works

Distortion minimization. Given a mesh homeomorphic to a disk (to avoid any topological issues), there are several ways of defining a "good" parameterization depending on the distortion metric that is considered. The survey by Sheffer et al. (2006) has described a few of these methods. Some works focus on finding conformal parameterization (Lévy et al., 2002), while other focus on finding a parameterization as isometric as possible, using energies measuring per triangle distortion such as ARAP, as-rigid-as possible, (Chao et al., 2010), Symmetric Dirichlet (Smith and Schaefer, 2015), or MIPS, most isometric parameterization, (Hormann and Greiner, 2012). In the following, we focus on the Symmetric Dirichlet energy as introduced by Smith and Schaefer (2015). Once the distortion metric has been specified, minimizing this energy is a problem in itself. Most optimization processes consist of two steps: (i) initialize the parameterization (with methods like Tutte (1963), which guarantees a injective parameterization) and (ii) minimize the distortion energy. To minimize these energies, we rely on quasi-Newton methods, and we need ways to approximate the Hessian with a positive symmetric definite matrix (PSD). Note that Smith et al. (2019) have provided a way to efficiently compute a PSD matrix approximating the Hessian for distortion energies.

Cutting methods. There are several works focusing on computing cuts for a parameterization. The method Sheffer and Hart (2002) searches for some high curvature point (likely to cause distortion) and then approximates a minimal path between these points. Recently, Zhu et al. (2020) have introduced a method that detects some feature points and connect them by approximating a Steiner tree problem. While these methods are efficient, our work aims at developing a method based on simultaneous optimization, in which little work has been done compared to these methods based on connecting high distortion points. Sharp and Crane (2018) have also provided a variational approach to compute cuts minimizing the distortion. However, their work remains limited to conformal parameterizations, while we aim at reducing an isometric distortion.

Simultaneous cuts and distortion optimization. Recent works aim at simultaneously finding the cut and the parameterization. Notable works are AutoCuts (Poranne et al., 2017), a variational model using an energy which is the sum of a distortion energy (such as Symmetric Dirichlet) and a per-edge energy approaching the cut length measure. The parameterization is treated as a triangle soup with the per-edge energy deciding whether neighbor triangles are attached to each other or not. The process can be automated, but user input is necessary in order to guarantee the bijectivity of the resulting parameterization. The weight between each term has to be defined by the user, and there is no guarantee that the result can reach a distortion or a cut length below a chosen threshold. The OptCuts method (Li et al., 2018) is based on a set of operations on the mesh topology such as cutting or merging, and uses a dual formulation of the energy problem to find the most appropriate operation to solve the problem. This method generally outperforms AutoCuts in terms of balance between distortion and cut length, with lower timings as well. This method can be adapted to guarantee bijectivity, and a distortion threshold to be reached can be set. Our method aims at providing a variational model, in a way similar to AutoCuts, but inspired by the Ambrosio-Tortorelli functional instead of using a handmade per-edge energy. We can then take inspiration from other Ambrosio-Tortorelli optimization related works to obtain an efficient and fast strategy to find our cuts.

Bijectivity. One of the problems of surface parameterization is to guarantee the bijectivity of the result. If the embedding proposed in Tutte (1963) is guaranteed to be valid, optimizing an energy such as Symmetric Dirichlet has no reason to maintain it. One needs to enforce the bijectivity during the distortion optimization. Smith and Schaefer (2015) have decomposed the problem into two subproblems. First, we need to make sure that triangles are not flipped during the optimization, which is solved by adding a constraint on the distortion energy and a specific line search in the Quasi-Newton solver. Then we need to prevent overlaps of non-adjacent regions which can be detected for instance by preventing boundary edge crossings. Smith and Schaefer (2015) have solved this problem using a barrier energy term. Su et al. (2020) have optimized this approach by building a shell around the initial parameterization to reduce the number of border edges and vertices. Alternatively, Jiang et al. (2017) have proposed to triangulate the remaining space around the parameterization, which in turns guarantees local injectivity on the resulting triangulation. In our proposal, we mainly focus on speeding up the distortion/cut joint optimization. Our approach is fully compatible with existing strategies (either using Smith and Schaefer (2015) or Jiang et al. (2017)), if global bijectivity is required.

Mumford-Shah related optimization. The Mumford-Shah functional was first introduced for image processing (Mumford and Shah, 1989), as a tool for image restoration and segmentation. It gives a way to represent an image as a piecewise-smooth function with a set of discontinuities. Since this functional is difficult to optimize (see next section), a lot of approximations of this functional have been proposed in the literature. We focus here on one relaxation of this functional, called Ambrosio-Tortorelli functional (Ambrosio and Tortorelli, 1990). It has seen many applications in image processing since then, some direct such as segmentation, denoising, and some variants designed for inpainting, magnification, deblurring, or image registration. While this functional has remained largely ignored in geometry processing, it has been used recently for mesh processing applications (Coeurjolly et al., 2016; Bonneel et al., 2018; Liu et al., 2020).

The article is organized as follows. First, we briefly recall the Ambrosio-Tortorelli functional in the continuous setting (Section 3). Then, we relate the functional to the mesh parametrization problem and we propose a new discretization scheme. This discretization induces a fast joint distortion/cut optimization algorithm (Section 4). Finally, Section 5 details an experimental validation of the proposed approach.

3. The Ambrosio-Tortorelli functional

Our joint distortion/cut variational model takes inspiration from the Ambrosio-Tortorelli functional. As detailed below, it can represent in the same framework the parameterization itself and the cut loci as functions.

In order to describe the Ambrosio-Tortorelli functional, it is necessary to introduce the Mumford-Shah functional (Mumford and Shah, 1989). The Mumford-Shah functional allows us, given an arbitrary function g, to find a new function u that will try to be as close as possible to g while also trying to be as smooth as possible everywhere except along a set K of discontinuities. This functional is commonly used in image processing, with a direct use in denoising: we want a smooth output close to the original input image, but we do not want to oversmooth meaningful image contours, so the function should have discontinuities along these contours. Ideally, the set K delineates these features. The functional is defined as:

$$\mathcal{MS}(K,u) := \alpha \int_{\Omega\setminus K} |u-g|^2 + \int_{\Omega\setminus K} |\nabla u|^2 + \lambda \mathcal{H}^1(K\cap\Omega)\,, \tag{1}$$

where Ω is the domain, g is the input data defined on Ω, u is the regularized data, K is the set of discontinuities, $\mathcal{H}^1$ is the Hausdorff measure, and (α,λ) are two real numbers corresponding to the weights given to the fitting and cut length terms respectively. Increasing coefficient α forces the output to be closer to the input, while increasing coefficient λ induces fewer discontinuities.

the first term of this functional describes the fitting of u to the input data g, the second term expresses the regularity of u while the third term measures the length of the discontinuities. Minimizing these three terms simultaneously lets us find a compromise between input fitting, smoothness of the result and how much discontinuity is allowed. Coefficients α and λ let us control this compromise.

However, this functional is hard to optimize since it is difficult to manipulate a set of discontinuities. Ambrosio and Tortorelli (1990) have provided a relaxation of the Mumford-Shah functional, as it approximates its minimizer with two smooth functions, instead of one smooth function and one set of discontinuities. It can be defined as:

$$\mathcal{AT}_\epsilon(u,v) := \alpha \int_\Omega |u-g|^2 + \int_\Omega |v\nabla u|^2 + \lambda \int_\Omega \left(\epsilon|\nabla v|^2 + \frac{1}{4\epsilon}(1-v)^2\right),$$

where function v is from domain Ω to $[0,1]$, and ϵ is a real positive number.

Ambrosio and Tortorelli (1990) have proven that this functional Γ-converges toward the Mumford-Shah functional as ϵ tends to 0. Function v converges to an indicator function of the space where discontinuities are not allowed: it is set to 1 where u is smooth, and to 0 where u can have discontinuities. A large value for ϵ will give a diffuse result, and as we decrease ϵ the function will be less and less diffuse and takes values closer to 0 and 1. The main interest of this formulation is that $\mathcal{AT}_\epsilon$ minimizers are the same as $\mathcal{MS}$ ones when ϵ tends to zero. Furthermore, for a given ϵ, an efficient alternating minimization scheme can be designed (see Alg. 1): by fixing v (respectively u) the resulting expression is convex quadratic in u (respectively v). We can thus find the minimizing u or v with a single step of a Newton method. We clarify the gradients and Hessian operators for the discretization of this functional in the next section.

Algorithm 1: Optimization method for the Ambrosio-Tortorelli functional

```
Data: $\epsilon_1$ the starting epsilon, $\epsilon_2$ the ending epsilon, $n$ a fixed number of optimization steps
Result: $v$ and $u$ approximation of the functional minimizer
$\epsilon \leftarrow \epsilon_1$
while $\epsilon > \epsilon_2$ do
    for $i \leftarrow 0$ to $n$ do
        $v \leftarrow \min_v(\mathcal{AT}_\epsilon(u,v))$
        $u \leftarrow \min_u(\mathcal{AT}_\epsilon(u,v))$
    end
    $\epsilon \leftarrow \epsilon/2$
end
```

4. Our discrete AT inspired model for joint parameterization/cuts

4.1. Discretization

We work on a three-dimensional surface, specifically a triangle mesh, defined by a set V of vertices and a set F of triangular faces. We denote as n_f the number of triangles of the mesh, and n_v the number of vertices.

Functions u and v must be discretized. Standard discretizations make u and v defined per vertex, but this approach tends to smooth the functions. This is not desirable, especially for v. We thus prefer u and v be discretized not at the same place on the mesh. Since u represents our parameterization, it is natural to sample it at either vertices or triangle corners: u is then linearly interpolated on each triangle as expected. Our approach consists in sampling v at faces instead, and u at vertices. This means we let some faces free to have as much distortion as needed in order to let other faces place themselves freely, acting as if there was a cut along these faces: on these faces, v is close to zero. We then have to cut alongside the distorted faces. These faces can be seen as being pulled on, the cuts then releasing the tension (see figure 1).

We can also notice that the gradient term in the Ambrosio-Tortorelli formula serves as a measure of distortion, acting as a weighted Dirichlet energy formulation ($|v\nabla \mathbf{u}|^2$). It then makes sense to see Ambrosio-Tortorelli as a process trying to reduce as much as possible this energy while allowing some exceptions: all we need then is a way to transform those exceptions into cuts. We also use Symmetric Dirichlet instead of Dirichlet, for local injectivity reasons, rescaled to be 0 when minimal. Since the gradient term already

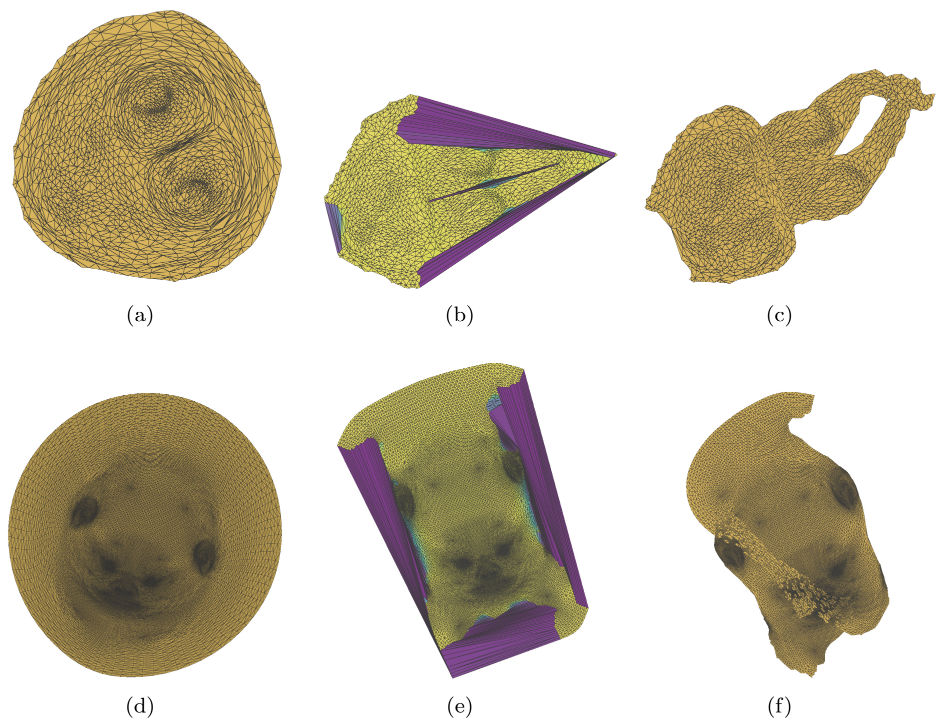


Figure 1: Parameterization at different stages of the method on the bunny head and face mesh: the initial parameterization (a, d), we then allow some faces (in magenta) to have distortion (b, e) so the other have more freedom of movement. We then compute cuts along these faces, giving us the result (c, f).

takes into account the distortion, and we do not have any data to be close to, we drop the first term of the functional, the fitting term.

The formulation below expresses the energy per triangle f:

$$E_f(\mathbf{u}, v) := A_f(v^2+\gamma)\underbrace{\left(\frac{|\nabla \mathbf{u}_f|^2 + |\nabla \mathbf{u}_f|^{-2}}{2} - 1\right)}_{\Psi_f(\mathbf{u})} + A_f\lambda\left(\epsilon|\nabla v_f|^2 + \frac{1}{4\epsilon}(1-v_f)^2\right)$$

$$= A_f\left((v^2+\gamma)\Psi_f(\mathbf{u}) - \lambda\epsilon v_f(Lv_f) + \frac{\lambda}{4\epsilon}(1-v_f)^2\right), \tag{2}$$

where A_f denotes the area of face f, $\nabla \mathbf{u}_f$ is the usual constant gradient over a triangle of the linearly interpolated $\mathbf{u}$ function sampled at triangle vertices. Furthermore, v_f denotes simply the sampled value of function f on face f, $|\nabla v_f|^2$ is the squared norm of the gradient of v within f, whose integral over the face f is approximated as $-A_f v(f) Lv_f$, with Lv_f the DEC Laplacian of v on the dual mesh (see Desbrun et al. (2005)), γ is a fixed constant, necessary to avoid a null coefficient in front of the distortion term: it would lead to instability during the optimization. Results presented here use $\gamma = 10^{-7}$. Finally, $\Psi_f(\mathbf{u})$ is the distortion measure of the face f.

4.2. Optimization

Optimizing this functional is done by alternating two steps, as seen previously: one step has $\mathbf{u}$ fixed and one step has v fixed:

- At v fixed, the only non-constant term is the distortion term. We optimize this term using the quasi-Newton method from Smith et al. (2019): we compute a modified Hessian (guaranteed to be SPD) of

Model	`bunnyhead`	`camelhead`	`bimba`	`hand`	`cat`	`armadillo`	`duck1`	`planck1`	`venus`	`tooth`	`circular box`
OptCut's time	2.5s	89s	13s	0.60s	0.65s	29s	2.08s	**1.19s**	0.61s	25.4s	22s
Our time	**1.6s**	**38s**	**6s**	**0.13s**	**0.11s**	**6.1s**	**0.77s**	1.32s	**0.36s**	**2.1s**	**3.8s**
Initial distortion	9.10	7.45	7.64	10.04	6.67	10.3	5.59	9.03	8.78	4.26	5.28
OptCut's distortion	**4.36**	4.26	4.50	5.2	4.38	5.22	4.49	4.54	4.37	4.24	4.53
Our distortion	4.39	4.26	4.50	5.3	4.50	5.22	4.49	4.56	4.39	4.25	4.54
OptCut's cut length	**2.88**	2.93	2.01	**4.5**	**1.28**	**2.07**	**1.09**	**1.61**	**1.70**	0.24	**1.13**
Our cut length	3.82	**2.56**	**1.90**	5.6	1.76	4.25	1.31	2.33	2.94	**0.10**	1.50

Table 1: Quantitative comparison between OptCuts method and our method, in terms of computation time, distortion and cut length. The `bimba`, `hand`, `cat`, `armadillo`, `duck`, `planck`, `venus`, `tooth` and `circular box` meshes come from the available input mesh in OptCuts Li et al. (2018) repository.

the distortion energy as well as its gradient. We can then compute how we want to modify the current parameterization, and we use a line search to avoid triangle flip (following Smith and Schaefer (2015), using the implementation provided by Jacobson et al. (2018)). We repeat until the gradient norm falls under a threshold.

- At $\mathbf{u}$ fixed, the expression is convex quadratic as seen for AT previously in Eq.(2). We can put the energy to optimize at this step under the form

$$E := v^T M \Psi(\mathbf{u}) v + \lambda \left(\frac{1}{4\epsilon} (v^T M v - 2Mv) - \epsilon v^T L v \right) + c \,, \tag{3}$$

which is of the form $E = v^T H v + 2b^T v + c$ (with c a constant term that can be ignored). Its minimum can be found at $-H^{-1}b$. We need the expression of H and b:

$$H = \frac{\lambda}{4\epsilon} I_n - \epsilon\lambda L + M\Psi(u), \quad b = -\frac{\lambda}{4\epsilon} M\, \mathbf{1}_n \,, \tag{4}$$

where L is a face Laplacian matrix (we used the DEC Laplacian on the dual mesh as in Desbrun et al. (2005)), $\Psi(u)$ is a diagonal matrix containing the distortion of each face (the i-th diagonal term corresponds to the i-th face), M is a face mass matrix, for example containing the area of each face on the diagonal, and finally I_n is the identity $n_f \times n_f$ matrix and $\mathbf{1}_n$ is a n_f-row vector filled with ones.

While on the second step finding the minimum only requires one iteration, the quasi-Newton of the first step has to be iterated multiple times before convergence (sometimes hundreds of times), making it much more time consuming than the second step.

4.3. Cutting

The scalar function v is used as a support for the cut extraction. Indeed, for triangles with values close to 1, the distortion term of the energy fully applies and is minimized during the optimization. On the contrary, triangles with values close to zero correspond to discontinuities in the energy minimization and thus will be used to delineate the precise location of cuts (see Figure 3).

We thus need a way to convert this scalar information into sequences of edges corresponding to cuts. To do so, we employ a simple strategy: we first define patches of triangles as simply connected components of traingles with values v lower than 0.5. Thanks to the Ambrosio-Tortorelli formulation with ϵ tending to zero, this simple process allows us to define thin strips of triangles. For each patch, we take the two farthest points belonging to this patch and find the shortest path between them: this defines the cut for this patch as a path of edges. The distance used is the distance on the underlying graph, with a weight of 0 for edges already belonging to the border. This trick forces the cut to go through the border if it is possible, to avoid having a cut parallel to an already existing border. Finally, we only keep cuts with lengths that are above 20% the length of the longest cuts, in order to avoid too small cuts.

This method was chosen for its simplicity and its effectiveness at capturing most of the areas to cut through without inducing too lengthy cuts, as seen on Figure 3. Depending on the application, one could be interested in iterating over this process: solve the AT functional to retrieve u and v, extract some long cuts, and iterate. However, in the following experiments, we only consider a single step for best performances.

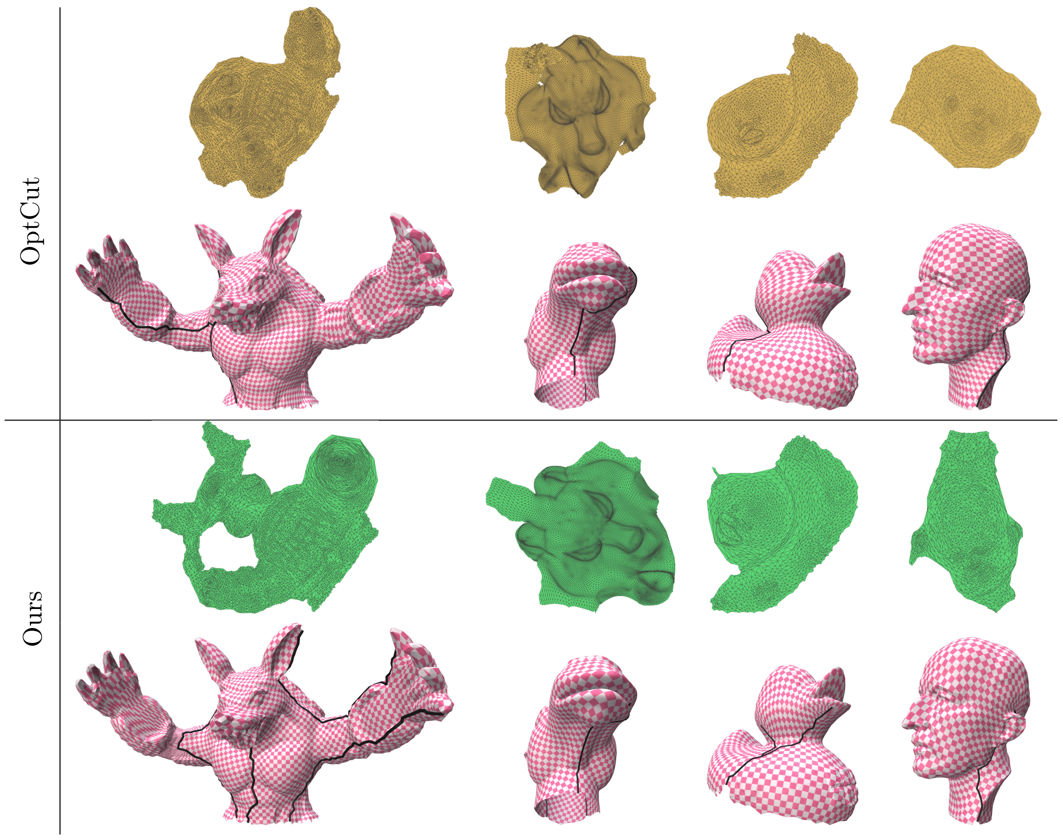


Figure 2: Visualization of the parameterizations evaluated in table 1 for the `armadillo`, the `camelhead`, the `duck` and `planck` meshes. Bottom row are our parameterizations, top are optcuts'.

4.4. Overall algorithm

Algorithm 2 gives an overall description of our method. This method can be run multiple times, for example if the cutting methods missed some places to cut. We can also use more complex energy when solving for u, for example, if we need to incorporate some border energy for border injectivity. This only changes the `Projected_Newton_Solver` part, which needs to incorporate the Hessian and gradient of the additional energies.

5. Results

This section gathers several experiments that show the potential of our method for mesh texture mapping. We first compare our approach to the state-of-the-art OptCuts method Li et al. (2018), both quantitatively and qualitatively. Then we examine the influence of the 4 parameters of our method. We finally show how our method can easily integrate user constraints in its formulation, like cutting in less visible zones or create a tighter UV map that fits a bounding box.

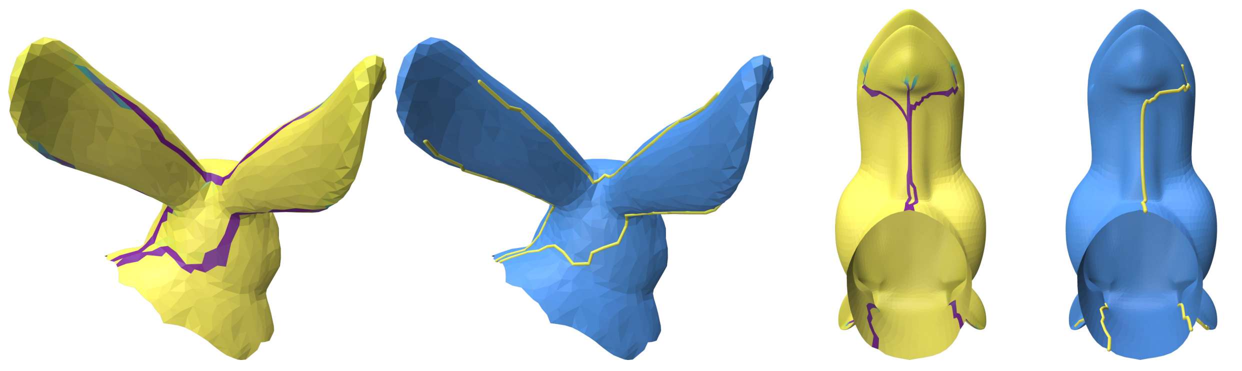
Note that the results presented here are of disk-like topology, as our method has a strong dependence on the initial cut. Indeed the method does not allow us to contest the edges included in the initial cut (we have not found a satisfying way to merge already cut edges). Furthermore the initial cut puts a strong constraint on the parameterization and consecutively on the evolution of v.

5.1. Influence of parameters

There are four parameters to tune: λ, ϵ_1, ϵ_2 and n the number of iterations between each change of ϵ (see the algorithm 1). We ended up using the following parameters by default: $\epsilon_1 = 0.1$,$\epsilon_2 = 0.01$ and $n = 3$.

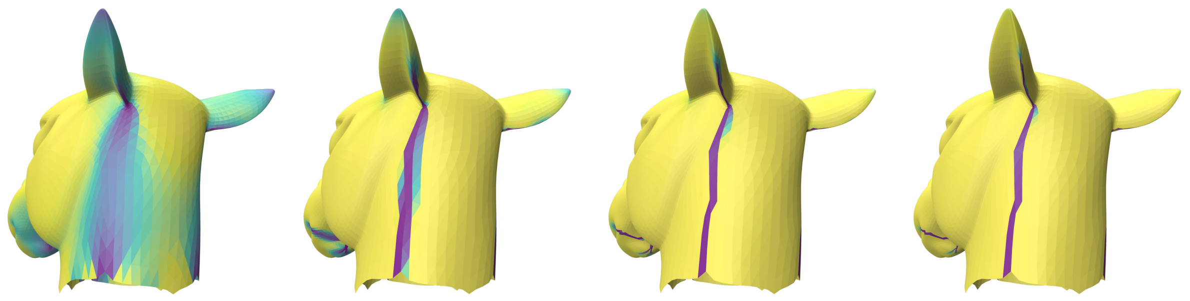
Algorithm 2: Pseudocode of our method for optimizing our model, which outputs the resulting parameterization

```
Data: $(V, F)$ a triangular mesh, $\epsilon_1$ the starting epsilon, $\epsilon_2$ the ending epsilon, $n$ a fixed number of
      optimization steps
Result: $U$ parameterization
Function Compute_AT_Parameterization(F, u)
    u ← Tutte(V, F);
    v ← $vec(1)$ ;
    $\epsilon \leftarrow \epsilon_1$ ;
    while $\epsilon > \epsilon_2$ do
        for $i \leftarrow 0$ to $n$ do
            u ← Projected_Newton_Solver( u, v) ;        // Compute parameterization using
             symmetric Dirichlet weighted by V
            v ← $H^{-1}b$ ;
        end
        $\epsilon \leftarrow \epsilon/2$
    end
    cuts ← Compute_Cuts( F, V, v) ;
    F, V ← Cut(F, V, cuts) ;
    u ← Projected_Newton_Solver( u) ;      // Compute final parameterization using non
     weighted symmetric Dirichlet
    return u;
end
Function Compute_Cuts( F, V, v)
    $S \leftarrow \{\}$;
    for $f$ in $F$ do
        if v (f) $< 0.5$ then
            $S \leftarrow S \cup \{f\}$;
        end
    end
    cuts ← {};
    foreach CC connected component in S do
        $P1, P2$ ← Get_Farthest_Points( CC);
        cuts ← cuts ∪ Get_Shortest_Path( CC, P1, P2);
    end
    return cuts;
end
```



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

Figure 3: Our cutting method chooses the longest path among each patch of "deformable faces" (in magenta on (a) and (c)). While it may not capture all of the area suggested (such as on the `camelhead` (d)), it is quite effective at capturing most of it and may filter some artifacts such as the loop at the base of the neck of the camel.



(a) $\epsilon = 0.1$ (b) $\epsilon = 0.05$ (c) $\epsilon = 0.025$ (d) $\epsilon = 0.0125$

Figure 4: Evolution of v during the iterations of the function `Compute_AT_Parameterization` of Algorithm 2 as the ϵ decreases. Starting from a smooth discontinuity map (a), v becomes sharper as ϵ tends to zero.

This results in 12 iterations of alternate optimization. We also use $\lambda = 1$.

Parameter λ can be considered as a cost for cuts and influence the final balance between distortion energy and the length of cuts. We choose $\lambda = 1$, but it may have to be changed depending on the number of cuts needed. Figure 6 illustrates the difference resulting from different λ. The three other parameters serve as convergence parameters: the wider the difference between ϵ_1 and ϵ_2, the more the problem starts from a diffuse and global point of view and ends at a local one. Parameter n governs how many iterations are necessary before convergence at a certain ϵ. Parameters ϵ_1 and ϵ_2 were chosen so that the starting problem is global and ends at a point where v seems reduced to one triangle wide lines along distorted faces, as seen on 4.

We choose $n = 3$, as it was the smallest value at which function v clearly distinguishes between rigid (v close to 1) and non-rigid faces (v close to 0). Higher values could be chosen in order to let v converge to a better solution. However, no significant difference was observed in tests with higher n. This could be due to the fact that, after ϵ changes its value for the first time, the number of steps for convergence is quite small (see figure 5): it seems that its impact happens at higher ϵ values and but ends up being erased with smaller ϵ values. This could also be due to the cutting method which is not refined enough to capture the difference a more precise v would make. It would make more sense to improve the cut before increasing the computation time significantly with a higher n.

We can also run the method several time to extend the cuts. Subsequent runs are usually faster than the initial ones, but the cuts added are also smaller. As seen on figure 7, they tend to prolong the initial

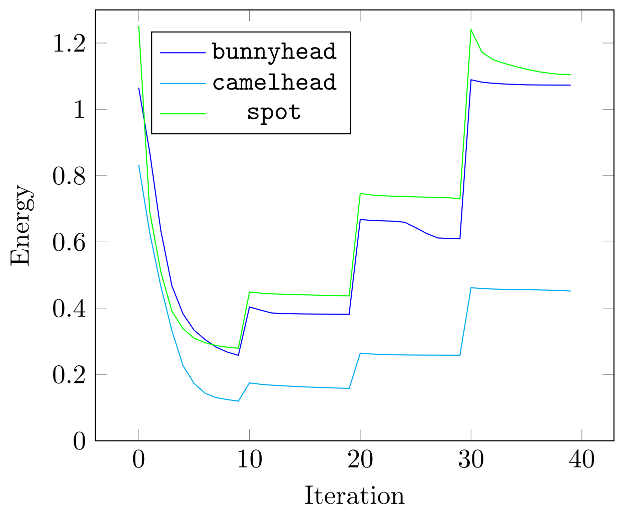


Figure 5: Evolution of our energy (defined in 2) when using $n = 10$ on various meshes (meaning ϵ changes every 10 iterations). After the first change of ϵ, it only takes a few iterations for the energy to converge.

choices instead of adding new isolated cuts.

5.2. *Comparison to state-of-the-art methods*

We choose to compare our method to the current implementation of OptCuts method (Li et al., 2018). Note that it performs better than when described in the original paper due to some recent optimizations. We were unable to compare our method to AutoCuts (Poranne et al., 2017), since we failed to reproduce their results: the convergence speed was much too slow and was not producing results close to the ones displayed in their paper. After discussions with the author, we learned that their results were obtained using a proprietary solver, whose current version is not compatible with their code. However, it seems that the current implementation of OptCuts performs better than AutoCuts in most cases, so the comparison with OptCuts is a good test for our approach. Table 1 sums up parameterization results obtained by OptCuts and our approach, in terms of computation speed, final distortion and final cut length. We use OptCuts distortion measures and cut length, respectively:

$$E_d := \frac{1}{\sqrt{\sum_{t\in F}|A_t|}}\sum_{t\in F}|A_t|\Psi(t)\,, \quad E_s := \frac{1}{\sqrt{\sum_{t\in F}|A_t|}}\sum_{i\in S}|e_i|\,. \tag{5}$$

Our method almost always outperforms OptCuts in terms of speeds. However OptCuts is often better in terms of cut length for distortion reduction (better means less cut length for the lowest distortion). Since our method does not allow targeting for a fixed distortion (while OptCuts does), we used our method first and then set OptCuts targeting the same distortion value we obtained. Bear in mind that the lowest distortion that a mesh can attain is 4 as it is the minimum of Symmetric Dirichlet energy, so a change from 4.2 to 4.1 is actually quite significant.

5.3. *Assisted cutting*

We wish to take into account some user input when choosing cuts. For example, some edges may be favored over others based on their visibility (less visible edges are preferred). To take this constraint into account, we can define λ as a scalar over the surface instead of a constant, and with a value depending on the user input. We tested this approach using visibility computed with `libigl` ambient occlusion (Jacobson et al. (2018)). Figure 8 illustrates how it affects the resulting distribution of v: areas to cut are more prone to be placed in low visibility areas such as cracks.

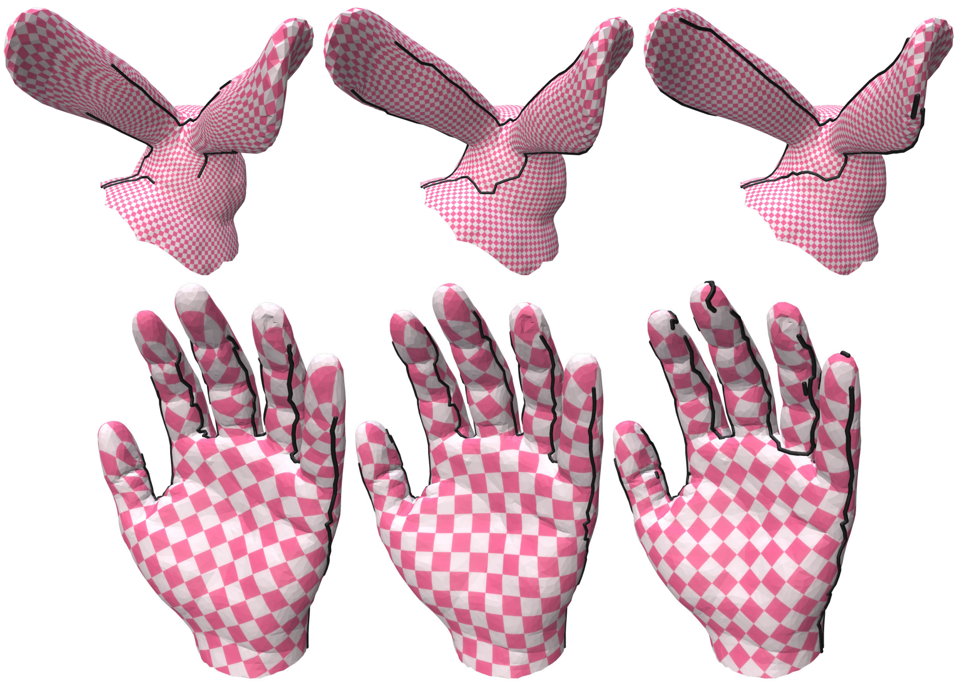


Figure 6: Results on the `bunnyhead` and `hand` meshes for $\lambda = \{3, 1, 0.1\}$ (reps. from left to right). Lower λ parameter implies longer cuts.

5.4. Packing constraints for the atlas

It may be desirable for a UV map to be as tight as possible in order to take less memory space. We follow Autocuts method Poranne et al. (2017) for fitting a parameterization within a predefined bounding rectangle. Figure 9 shows how UV map are then much more compact. However we observe a significant increase in the convergence time, depending on how tight the bounding box is: thee harder it is to fit the parameterization in the box, the longer it takes (sometimes taking 10 times as long as without).

6. Discussion and perspectives

We have presented a new variational model to compute a UV parameterization of a triangle mesh with a joint optimization of distortion and cuts. It is a discrete variant of the Ambrosio-Tortorelli functional, which offers theoretical guarantees for solving problems involving a penalization in the length of discontinuities. We have shown that this approach is effective in practice, faster than other simultaneous optimization methods, with quality often close to the state-of-the-art Optcuts method, and incorporates easily other constraints in its formulation like border injectivity, assisted cutting or compact texture maps. We can also note that the loci of deformable faces may have an arbitrary topology, so cuts can take various shapes. Finally, the code is available at `https://github.com/Lieunoir/UV-AT`.

In order to measure the distortion, we only tested the Symmetric Dirichlet energy, which was the closest to the original gradient term in the Ambrosio Tortorelli functional. As future works, other energies could be evaluated (such as ARAP), as they may not behave in the same way as Symmetric Dirichlet.

A limitation of the current cutting method is that it does not preserve the topology defined by v: if any cycle is present, we ignore it. It also does not capture all of the region defined by the underlying patch (if the region is a tree it will only choose the longest branch), meaning we have to run the method again to capture it all. Our current cutting method is well adapted to a semi-supervised setting where the user can decide to refine further, but cuts following the topology induced by v could indeed be interesting in an unsupervised model.

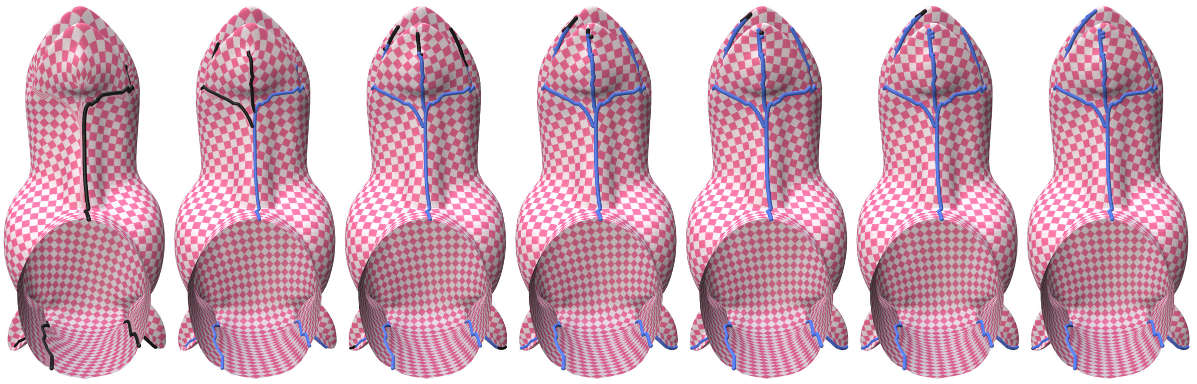


Figure 7: Results after multiple runs of our algorithm. This enables us to capture some natural cuts that our algorithm may have missed. After a few runs cuts our method converges, and the cuts stop expanding.

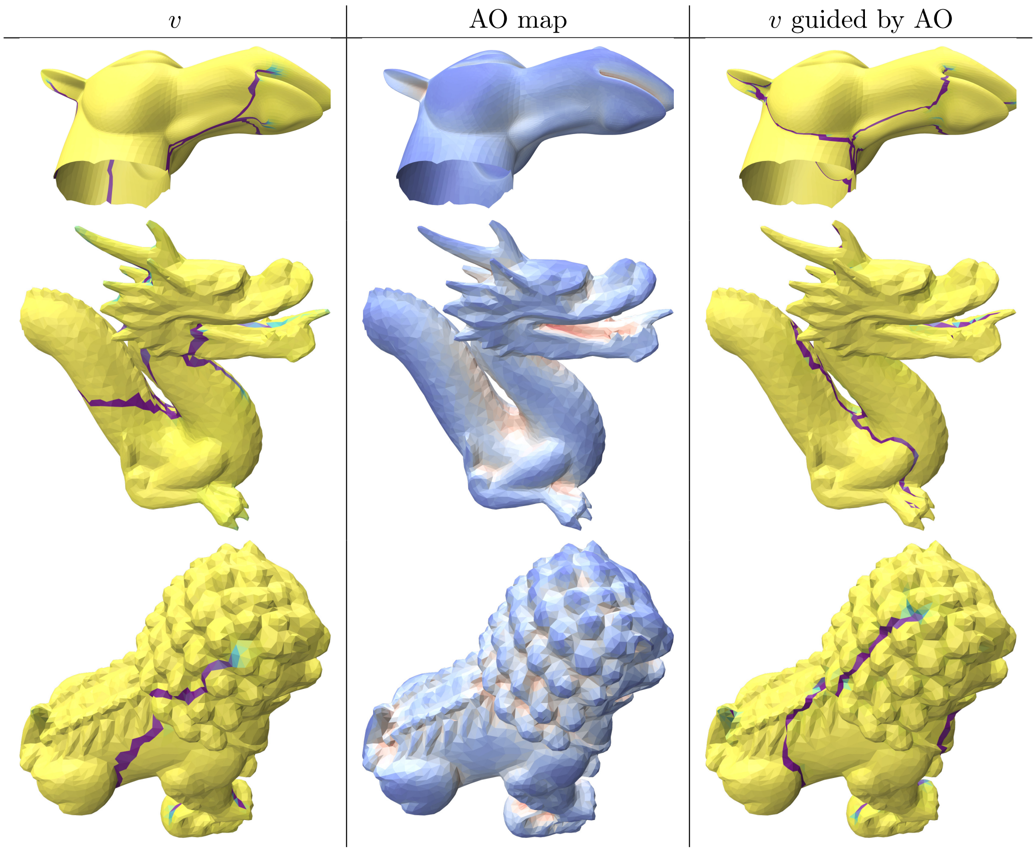


Figure 8: Optimized function v without (left) and with (right) the ambient occlusion (AO) taken into account (middle). When ambient occlusion is enabled, the faces identified as cut loci are guided to regions with lower visibility.

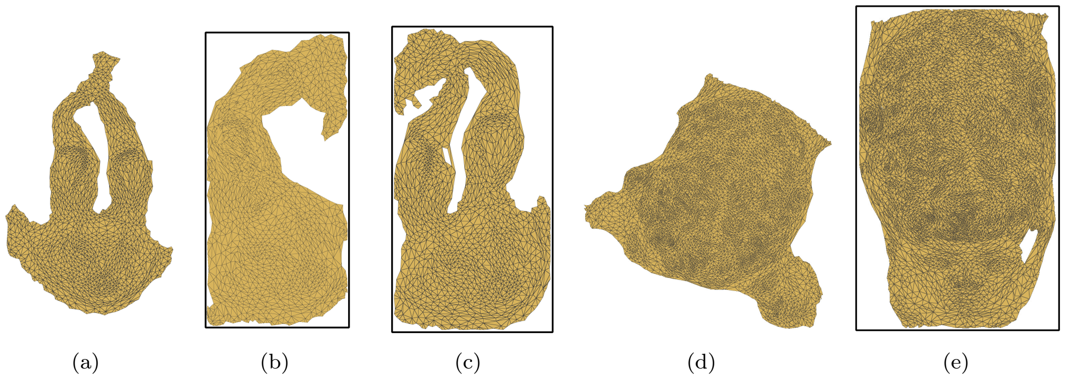


Figure 9: Resulting parameterizations without bounding box constraints (a, d) and with bounding box (b, c, e) on the `bunnyhead` and `david` mesh. The first example with `bunnyhead` (b) does not have border injectivity restrictions, enabling the many overlaps which are not present on the second example (c) (which has border injectivity restrictions). On the `david` mesh, an overlap free parameterization was obtained (e) without using our border injectivity restrictions, giving the same result with our without them.

In some cases, our method is not able to run properly because the energy diverges in the initialization. This problem comes from numerical issues of floating-point arithmetic. It is addressed by Shen et al. (2019), giving a alternative to Tutte's embedding for initialization. We have not run our method with this initialization, but we can safely assume that it does not affect significantly the results.

One way to improve the convergence speed would be to include some speedup optimization techniques. In particular, we could use the methods proposed by Liu et al. (2018) or Liu et al. (2021). We could also look for other ways to optimize the Mumford-Shah functional that would yield better results or give them faster: our alternate Ambrosio-Tortorelli optimization method is among the costliest ones. For instance, the recent competitive gradient and mirror descent algorithms (Schäfer and Anandkumar, 2019; Schäfer et al., 2020) seem to be particularly efficient in optimizing this type of functional, which are block-convex but not globally convex.

References

Ambrosio, L., Tortorelli, V.M., 1990. Approximation of functional depending on jumps by elliptic functional via t-convergence. Communications on Pure and Applied Mathematics 43, 999–1036. doi:https://doi.org/10.1002/cpa.3160430805.

Bonneel, N., Coeurjolly, D., Gueth, P., Lachaud, J.O., 2018. Mumford-shah mesh processing using the ambrosio-tortorelli functional. Computer Graphics Forum (Proceedings of Pacific Graphics) 37. doi:10.1111/cgf.13549.

Chao, I., Pinkall, U., Sanan, P., Schröder, P., 2010. A simple geometric model for elastic deformations. ACM Trans. Graph. 29. doi:10.1145/1778765.1778775.

Coeurjolly, D., Foare, M., Gueth, P., Lachaud, J.O., 2016. Piecewise smooth reconstruction of normal vector field on digital data. Computer Graphics Forum (Proceedings of Pacific Graphics) 35. doi:10.1111/cgf.13013.

Desbrun, M., Hirani, A.N., Leok, M., Marsden, J.E., 2005. Discrete exterior calculus. arXiv preprint math/0508341 .

Hormann, K., Greiner, G., 2012. Mips: An efficient global parametrization method. Curve and Surface Design: Saint-Malo 2000, 10.

Jacobson, A., Panozzo, D., et al., 2018. libigl: A simple C++ geometry processing library. Https://libigl.github.io/.

Jiang, Z., Schaefer, S., Panozzo, D., 2017. Simplicial complex augmentation framework for bijective maps. ACM Trans. Graph. 36. doi:10.1145/3130800.3130895.

Lévy, B., Petitjean, S., Ray, N., Maillot, J., 2002. Least squares conformal maps for automatic texture atlas generation. ACM Trans. Graph. 21, 362–371. doi:10.1145/566654.566590.

Li, M., Kaufman, D.M., Kim, V.G., Solomon, J., Sheffer, A., 2018. Optcuts: Joint optimization of surface cuts and parameterization. ACM Trans. Graph. 37. doi:10.1145/3272127.3275042.

Liu, H.T.D., Zhang, J.E., Ben-Chen, M., Jacobson, A., 2021. Surface multigrid via intrinsic prolongation. ACM Trans. Graph. 40. doi:10.1145/3450626.3459768.

Liu, L., Ye, C., Ni, R., Fu, X.M., 2018. Progressive parameterizations. ACM Trans. Graph. 37. doi:10.1145/3197517.3201331.

Liu, Z., Wang, W., Zhong, S., Zeng, B., Liu, J., Wang, W., 2020. Mesh denoising via a novel mumford-shah framework. Computer-Aided Design 126, 102858. doi:10.1016/j.cad.2020.102858.

Mumford, D., Shah, J., 1989. Optimal approximations by piecewise smooth functions and associated variational problems. Communications on Pure and Applied Mathematics 42, 577–685. doi:https://doi.org/10.1002/cpa.3160420503.

Poranne, R., Tarini, M., Huber, S., Panozzo, D., Sorkine-Hornung, O., 2017. Autocuts: Simultaneous distortion and cut optimization for uv mapping. ACM Trans. Graph. 36. doi:10.1145/3130800.3130845.

Schäfer, F., Anandkumar, A., 2019. Competitive gradient descent. Advances in Neural Information Processing Systems 32.

Schäfer, F., Anandkumar, A., Owhadi, H., 2020. Competitive mirror descent. arXiv preprint arXiv:2006.10179 .

Sharp, N., Crane, K., 2018. Variational surface cutting. ACM Trans. Graph. 37.

Sheffer, A., Hart, J.C., 2002. Seamster: Inconspicuous low-distortion texture seam layout, in: Proceedings of the Conference on Visualization '02, IEEE Computer Society, USA. p. 291–298.

Sheffer, A., Praun, E., Rose, K., 2006.

Shen, H., Jiang, Z., Zorin, D., Panozzo, D., 2019. Progressive embedding. ACM Trans. Graph. 38. URL: https://doi.org/10.1145/3306346.3323012, doi:10.1145/3306346.3323012.

Smith, B., Goes, F., Kim, T., 2019. Analytic eigensystems for isotropic distortion energies. ACM Transactions on Graphics 38, 1–15. doi:10.1145/3241041.

Smith, J., Schaefer, S., 2015. Bijective parameterization with free boundaries. ACM Trans. Graph. 34. doi:10.1145/2766947.

Su, J.P., Ye, C., Liu, L., Fu, X.M., 2020. Efficient bijective parameterizations. ACM Trans. Graph. 39. doi:10.1145/3386569.3392435.

Tutte, W.T., 1963. How to draw a graph. Proceedings of the London Mathematical Society s3-13, 743–767. URL: https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/plms/s3-13.1.743, doi:10.1112/plms/s3-13.1.743, arXiv:https://londmathsoc.onlinelibrary.wiley.com/doi/pdf/10.1112/plms/s3-13.1.743.

Zhu, T., Ye, C., Chai, S., Fu, X.M., 2020. Greedy cut construction for parameterizations. Computer Graphics Forum 39, 191–202. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13923, doi:https://doi.org/10.1111/cgf.13923, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13923.