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

Volume 33 (2014), Number 5

Remeshing-assisted Optimization for Locally Injective Mappings

Y. Jin [1,2] and J. Huang[†1,3] and R. Tong[1,2,3]

[1]State Key Laboratory of CAD & CG, Zhejiang University, China
[2]Artificial Intelligence Institute, Zhejiang University, China
[3]Cyber Innovation Joint Research Center, Zhejiang University, China

Abstract

Constructing locally injective mappings for 2D triangular meshes is vital in applications such as deformations. In such a highly constrained optimization, the prescribed tessellation may impose strong restriction on the solution. As a consequence, the feasible region may be too small to contain an ideal solution, which leads to problems of slow convergence, poor solution, or even that no solution can be found. We propose to integrate adaptive remeshing into interior point method to solve this issue. We update the vertex positions via a parameter-free relaxation enhanced geometry optimization, and then use edge-flip operations to reduce the residual and keep a reasonable condition number for better convergence. For more robustness, when the iteration of interior point method terminates but leaves the positional constraints unsatisfied, we estimate the edges in the current tessellation that block vertices moving based on the convergence information of the optimization, and then split neighboring edges to break the restriction. The results show that our method has better performance than the solely geometric optimization approaches, especially for extreme deformations.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

Computing a mapping under variants of constraints and distortion measurements for 2D triangular meshes has wide applications such as deformations. Usually it is required that the mapping should be locally injective, i.e. elements of the mapped mesh should have positive signed areas. A mapping without such a property may lead to serious failures in some applications, for instance, visual artifacts or invalid elements in texture mapping and remeshing.

However, it is challenging to compute locally injective mappings because of the highly non-linear inequality constraints. Though some methods have been proposed, they all suffer from certain limitations and are unsuitable for general applications. The convex combination based approaches [Tut63] [Flo03] require a convex boundary. Local optimization based methods [HG00] [SSGH01] are likely to get stuck in a local minimum. And path planning based methods [KSG03] [LYY08] cannot be extended for arbitrary deformation energies.

Recently, the strategy of global non-linear optimization attracts a lot of attentions [Lip12] [SKPSH13] [AL13], because it is more flexible in choosing different energies as well as incorporating various user controls. In general, these methods try to find an optimal solution in a feasible region characterized by a set of equality and inequality constraints. However, for some coarse meshes with extreme constraints, the feasible region is too small to contain an ideal solution due to the tessellation restriction. Besides, using the fixed tessellation will introduce numerical difficulties such as slow convergence or poor solutions because of the nearly degenerated triangles generated during the iterations.

A simple example is shown in Figure 1. The vertices b, c, d are fixed and a is targeted to the position a' which lies at the other side of the edge e_{bc}. Using the traditional solely geometry based optimization, when the vertex a moves towards a', Δ_{abc} will be nearly degenerated which leads to nu-

† Corresponding author: hj@cad.zju.edu.cn

merical difficulties, and the vertex a cannot reach a' without foldovers because it is blocked by the edge e_{bc}.

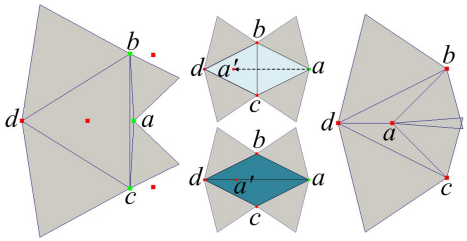


Figure 1: Illustration of a challenging case. The constrained vertices are marked in green and their target positions are marked in red.

Intuitively, if we can detect and remove this obstacle during the optimization, e.g. flip the edge e_{bc}, then a can be freely driven to the target position.

Based on this observation, we propose a remeshing-assisted optimization method for locally injective mappings of 2D triangular meshes with position constraints. To develop a relatively general method, we formulate the problem as an unconstrained optimization with penalty and barrier functions, and then iteratively solve the vertex position and update the tessellation according to the criteria based on residual, gradient and condition number of the energy.

The contributions of our approach are summarized as follows:

- A relaxation enhanced optimization strategy to improve constraint satisfaction and condition number alternatively, which features stable and faster convergence.
- An edge-flip based tessellation optimization algorithm to reduce residual, condition number and expand the feasible region, which accelerates convergence.
- An edge-split based local up-sampling scheme to further break the tessellation restrictions, which enhances the robustness.

The experiments show that our optimization solver provides the superiority in performance, distortions and capacity in handling challenging cases comparing to the state-of-the-art solely geometry optimization methods.

2. Related work

Computing locally injective mappings from triangular meshes to the plane has received much concern in geometry processing and modeling. It is in general a difficult problem and the existing methods are more or less limited to certain restrictions.

Some methods try to penalize degenerated and fold-over elements in the optimization. Bommes et al. [BZK09] proposed a local stiffening method. They added an adapted triangle weighting into the energy formulation to penalize high local distortions but it has no guarantees. Several other works used SVD of the deformation gradient to detect and resist local inversions [ITF04] [SA07] [LZX*08] [SHST12]. The idea is to rectify the matrix to be positive definite when it is not. However, the strategy has the effect of resistance but no avoidance.

To guarantee the injectivity, one representative work is the famous convex combination parameterization [Flo03], which is a generalization of Tutte's embedding [Tut63]. The parameterization is obtained by solving a sparse linear system and guarantees bijective in theory. However, it is limited to constraints with fixed convex boundary. Gortler et al. [GGT06] further studied the injective conditions for convex combination mappings when the target domain is non-convex. Xu et al. [XCGL11] provided a simple algorithm for non-convex boundary embeddings, which can either compute a valid embedding if it exists, or reject the input otherwise. Lipman [Lip13] recently introduced three sets of sufficient conditions for generating bijective simplicial mappings of manifold meshes. All these works however only consider the fixed boundary constraints, and cannot flexibly use different distortion measurements.

Some methods compute the locally injective mappings using specific distortion measurements. Weber et al. [WMZ12] used extremal quasi-conformal maps which are locally bijective. Lipman [Lip12] constructed a convex subspace of maps with bounded conformal distortions using quadratic energies. Several other methods develop a special distortion measurement which increases to infinity when elements are degenerated. For instance, "MIPS" [HG00] energy is inversely proportional to the area in the parameter-space element. Other energies, such as "stretch energy" [SSGH01] which is used to measure "texture stretch" and the area-preserving version of "MIPS energy" presented in [AZF13], naturally possess parameter-space area term as barrier function and thus have similar effects as well. The methods [YLY*12, SHSF13] restrict the mapping to be the composition of a series of special mappings. Aigerman et al. [AL13] introduced a method to project a mapping into the closest injective and bounded distortion one under Frobenius norm. Such methods cannot be easily extended to handle general distortion measurements, e.g. the non-linear Green's Strain energy [HG00].

Another strategy is to explicitly prevent foldovers during the iterations of optimization. For example, the methods [SSGH01, SLMB05] move a vertex in the kernel of its one-ring according to a locally defined energy. Schüller et al. [SKPSH13] incorporated an area based barrier term to arbitrary deformation energies to prevent inverted elements based on the idea of "interior point method". They also developed an associated numerical technique which can robustly and efficiently handle extreme deformations. However, as mentioned before, the tessellation may restrict the

feasible region, which leads to challenging numerical problems.

The idea of modifying tessellation has been explored for a long time. Fujimura et al. [FM98] presented a foldover-free 2D image warping method using the Delaunay triangulation and edge flips for giving trajectory of constrained features. The method [KSG03] begins with an unconstrained planar embedding and moves the constrained vertices to the target positions by a path planning strategy with adaptive remeshing. This method was further generalized to produce cross maps between two surfaces [KS04]. However, these methods might fail since they do not consider consistent neighbor ordering. The methods [ESG01, LYY08] integrate adaptive remeshing with the strategy of convex combination. They achieve better robustness, but inherit the limitations of convex combination based methods.

Our method is an integration of interior-point method and adaptive remeshing. It enjoys the advantages of flexible distortions control and imposing various constraints. And at the same time, it expands the feasible region by remeshing, which makes the optimization more efficient and robust.

3. Problem formulation

Given a mesh $M = \{V,T\}$ with vertices $V = \{v_i\}, v_i \in R^2$ and triangles T, the distortion of a mapping $\phi : R^2 \to R^2$ is usually formulated as:

$$E_d(\phi) = \int_M e(\phi,p)dp = \sum_{t\in T}\int_t e(\phi,p)dp, \qquad (1)$$

where, $e(\phi,p)$ measures the distortion of ϕ at point $p \in M$.

The mapping ϕ is usually parametrized and discretized in the triangular mesh via a set of basis $\psi = \{\psi_i\}$ *associated with triangles* T and the coefficients $u = \{u_i\}$ for every vertex i:

$$\phi(p,u,\psi) = \sum_{i\in V}\psi_i(p)u_i. \qquad (2)$$

If barycentric basis is used, $e(\phi(p,u_i,\psi_i))$ is often piecewise constant and $u_i = \phi(v_i)$, leading to a simple discretized formulation:

$$E_d(u,\psi) = \sum_{t\in T} e_t(u). \qquad (3)$$

We formulate position constraints $u_i = p_i, i \in \mathcal{C}$ as equalities $P(u) = Cu - p = 0$, and use inequality for local injective constraint for each triangle t:

$$G_t(u,\psi) = A'_t(u) - \delta A_t > 0, \quad t \in T, \quad 0 < \delta < 1, \qquad (4)$$

where $A_t, A'_t(u)$ are the signed area of triangle t before and after applying the mapping ϕ respectively.

Similar to [SKPSH13], the constrained optimization problem can be converted into an unconstrained optimization with objective function $E(u,\psi)$ using interior point method, namely imposing the positional constraints by penalty and the positive area constraints as barrier terms:

$$\begin{aligned} E(u,\psi) &= E_d(u,\psi) + E_c(u) + E_b(u,\psi) \\ &= E_d(u,\psi) + \alpha\|P(u)\|^2 + \beta\sum_{t\in T}\chi_t(G_t(u,\psi)), \end{aligned} \qquad (5)$$

where α and β are weights and χ_t is the barrier function for triangle t, which is defined as the same as in [SKPSH13].

4. Our Method

Most of the methods solve the above problem (Equation 5) by optimizing u only. We introduce ψ, i.e. the triangulation T of the mesh, as an additional degree of freedom in this optimization. Therefore, we adopt a "remeshing-assisted" optimization method, which aims to vary ψ as well as u to break the tessellation restrictions, reduce the energy sufficiently, keep a reasonable condition number, and thus speed up the convergence.

4.1. Overview

Our method utilizes an **"inner-outer iterative"** procedure until it converges with all the positional constraints satisfied. In the inner iteration, we apply edge-flip operations along with a new relaxation substepping once vertex coordinates are updated by the Newton-like optimization. While in the outer iteration, we introduce edge-split operations to further break the tessellation restrictions. Although this idea can be flexibly integrated into various solvers, in this paper, we apply it to a relaxation enhanced geometry solver which is a variant of "interior point method" proposed in [SKPSH13] to show the benefit of remeshing. The core procedure can be seen in Algorithm 1, where N_{in} and N_{out} are maximal numbers of the inner and the outer repeats (in default, $N_{\text{in}} = 1000$ and $N_{\text{out}} = 10$). The detail of the algorithm will be introduced below.

4.2. Relaxation enhanced geometry optimization

With fixed ψ to optimize the energy function (Equation (5)), a variant of the Levenberg Marquardt (LM) algorithm [SKPSH13] can be used. Note that the barrier term E_b is a non-linear and non-convex function, which plays an important role in the convergence of the optimization. Once some elements of the mesh are nearly degenerated, the Hessian of the energy function $\nabla^2 E$ may become ill-conditioned, resulting in poor or even failure quadratic approximation of E. The method thus adopts a parameter-dependent substepping strategy which relaxes the positional constraints when its Hessian is not positive definite [SKPSH13]. This strategy shares some similarities to path planing [LYY08], and solves a series of approximation to Equation (5). It achieves better convergence than directly solving Equation (5), but depends on the parameter μ which is used to regularize the Hessian to set intermediate

Algorithm 1 Remeshing-assisted optimization

Input:
Rest-pose planar mesh, M;
Positional constraints, $Cu = p$.
Output:
Deformed planar mesh, M'

```
repeat
  repeat
    UpdateVertices(M, M')
    if the Equation 7 or 8 is satisfied then
      break
    end if
    FlipEdges(M, M')
    RelaxVertices(M, M')
  until number of repeats ≥ N_in
  if ||P(u)||_2^2 > ε_c then
    SplitEdges(M, M')
  end if
until number of repeats ≥ N_out
```

target positions. Furthermore, the inconsistency of the objective functions during the optimization leads to difficulty in designing robust remeshing criteria for tessellation optimizing.

The success of "substepping" is largely due to preventing from generating high degenerated triangles, in other words, keeping the problem well-conditioned. Based on this understanding, we propose a strategy alternatively improving the satisfaction of constraints and the condition number. The key idea is to use an additional relaxation step (the subroutine **RelaxVertices** in Algorithm 1) after each vertex-update step (the subroutine **UpdateVertices**) to improve the condition number of the Hessian. The **UpdateVertices** subroutine is much the same as was done in [SKPSH13]. Namely, it searches a direction $\boldsymbol{s}$ via Newton's method,

$$\boldsymbol{s} = -(\nabla^2 E + \mu \boldsymbol{I})^{-1} \nabla E, \tag{6}$$

and calculates a step size decreasing the energy by backtracking line search method. In the **RelaxVertices** subroutine, we alter the positional energy term E_c temporally by replacing the final targets with the current positions and repeatedly regularize the Hessian to make it positive definite and relax the vertices with the same method as used in **UpdateVertices** until no regularization is needed or no vertex would move any more (satisfying the Equation 8). This step keeps the value E_c, but reduces the other two energy terms E_d and E_b. It improves the shape quality of the triangles (see Figure 2), which provides well-conditioned Hessian for the **UpdateVertices** subroutine in the next iteration to find a good search direction.

After each subroutine **UpdateVertices**, we will check the

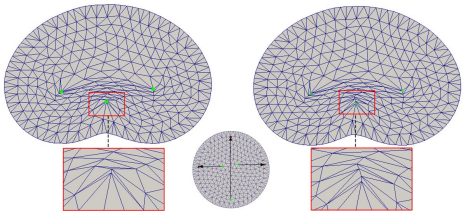


Figure 2: The deformed meshes of the fourth iteration before (left) and after relaxation (right). Note that the shape quality and size are largely improved after relaxation.

following two convergence criteria:

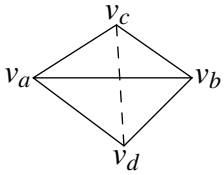
$$||\nabla E_n(u)||_2^2 < \varepsilon_g; \tag{7}$$

$$||u_n - u_{n-1}||_\infty < \varepsilon_f, \tag{8}$$

where $n(n \geq 1)$ is the current number of iterations, ε_g and ε_f are small constants (we set $\varepsilon_g = 10^{-6}$ and $\varepsilon_f = 10^{-8} D_M$ in default, where D_M is the diameter of the bounding box of the mesh M). We call the criterion of Equation 7 as "gradient convergence" and that of Equation 8 "step convergence". If either criterion is satisfied, the inner iteration will terminate.

4.3. Edge-flip based tessellation optimization

During the geometry optimization, the tessellation of the given mesh may impose strong restrictions on the solution, which leads to problems of slow convergence, poor solution, or even that no solution can be found. Our observation is that fixing u, and optimizing Ψ by adaptively flipping edges in the triangular mesh can break this restriction and reduce the residual as well as the condition number of the energy function.



Note that we need to maintain two meshes while applying the edge-flip operations: the rest-pose mesh M and the deformed mesh M'. Given a pair of triangles in the mesh M neighboring to an inner edge e_{ab} (see the inset), there are two possible tessellations of the diamond region. The mapping on this region can be discretized by two sets of bases, either the original bases $\Psi_{ab} = (\psi_{abc}, \psi_{adb})$, or the bases $\bar{\Psi}_{ab} = (\psi_{adc}, \psi_{dbc})$ after flipping the edge e_{ab} into e_{cd}.

Obviously, to keep the solution in the feasible region defined by foldover-free inequality constraints, the edge flip is allowed only if the new triangles $\Delta_{adc}, \Delta_{dbc}$ have positive signed areas in M, and satisfy the low-bound area condition:

$$G_t(u, \bar{\psi}) > 0, t \in \{\Delta_{adc}, \Delta_{dbc}\}, \tag{9}$$

where $\bar{\psi}$ is the set of basis associated with the new triangulation after edge flipping.

For each edge that can be flipped, using Ψ_{ab} and $\bar{\Psi}_{ab}$, we evaluate the energy values E_{ab} and $\bar{E}_{ab}$ respectively in this diamond region that represent the summation of the distortions and barrier energies, and without involving the penalty of position constraints. Then the operation of flipping edge e_{ab} is labeled with the value $\Delta_{ab} = E_{ab} - \bar{E}_{ab}$, which indicates the amount of residual decreasing by flipping edge e_{ab}. We push all the flipping operations associated with enough residual reduction $\Delta_{ab} > \eta E_{ab}$ into a max-heap, and then flip the edges in a descending order. After each flipping of an edge e_{ab}, boundary edges of the associated diamond region are removed away from the heap.

The above adaptive rule guarantees the monotonically decreasing of the residual, but it may significantly increase the condition number, and incur handicaps in the future iterations. In order to keep a reasonable condition number of problem, we cancel the flipping operation if it increases the condition number, namely:

$$\mathrm{cond}(\bar{E}_{ab}) > \mathrm{cond}(E_{ab}), \tag{10}$$

where $\mathrm{cond}(E_{ab})$ stands for the condition number of the energy function E_{ab}, which is given by [TBI97]:

$$\mathrm{cond}(E(u)) = \frac{||\nabla E(u)||_\infty ||u||_\infty}{E(u)}. \tag{11}$$

In this step of remeshing, the vertex positions are fixed, but the residual and condition number monotonically decrease because the tessellation is adaptively changed. Figure 3 shows a mesh before and after edge-flip operations.

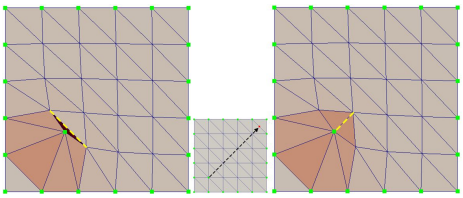


Figure 3: The deformed meshes before (left) and after (right) flipping edges.

4.4. Edge-split based local up-sampling

The edge-flip operations are able to reduce the energy and accelerate the convergence by adaptively choosing basis function to discretize the mapping. However, it may not be enough to get a solution satisfying some extreme constraints on coarse meshes. Such situation happens when certain edges of the deformed mesh block the way of the moving vertices yet edge-flip operations are not allowed due to the violation of the flipping criteria. Thus, we resort to other operations, e.g. edge-split, to break these edges and further expand the feasible region. Highly inspired by the work [LYY08], where the authors used edge-flip operations to align positional constraints without foldovers, we introduce a similar scheme for edge-split to solve this problem. In their work, they take the rays starting from the constrained vertices and pointing to their targets as the moving directions, which is used to detect fold-over triangles in advance and flip related edges of these triangles to remove the potential foldovers. The strategy of our edge-split operations are much the same which involves two steps: 1) calculate the moving direction $\overrightarrow{F_v}$ for each vertex v; and 2) determine the related edges to split.

We determine $\overrightarrow{F_v}$ according to the convergence information (CI) provided by the inner iteration. If the inner iteration is of gradient convergence (G), then the step provided by **UpdateVertices** as well as the gradient of the energy approach zero, and no direction is provided. Thus we take the negative gradient of the energy E_c as the direction. Otherwise, we check the last step $\boldsymbol{s}$. If $\boldsymbol{s}$ is very small, it implies that the subroutine **UpdateVertices** fails to find a large enough step $\boldsymbol{s}$ to reduce the energy along the Newton direction, which usually results from poor quadratic approximation to the energy. Thus, we approximate $\overrightarrow{F_v}$ by the negative gradient of the energy function. If it is large enough, the Newton direction is still good, thus $\boldsymbol{s}$ can be used to approximate $\overrightarrow{F_v}$. In all, the moving direction $\overrightarrow{F_v}$ is calculated as follows:

$$\overrightarrow{F_v} = \begin{cases} -\frac{\partial E_c(v)}{\partial v} & \mathrm{CI} = \mathrm{G}, \\ -\nabla E(v) & \mathrm{CI} \neq \mathrm{G}, |\boldsymbol{s}|_\infty < \varepsilon_f; \\ \boldsymbol{s} & \mathrm{CI} \neq \mathrm{G}, |\boldsymbol{s}|_\infty \geq \varepsilon_f; \end{cases} \tag{12}$$

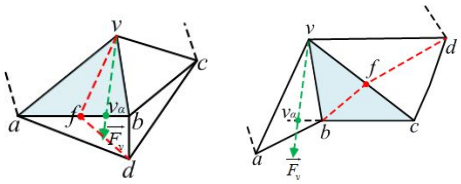
where ε_f is given in 8.

With the direction given on the vertex v, we can use the similar scheme in [LYY08] to choose the related edges of its neighboring faces $N(v)$ to split and handle for all vertices with non-vanishing $\overrightarrow{F_v}$ one by one. The scheme [LYY08] detects the foremost potential fold-over triangles via the intersection point v_α defined as:

$$v_\alpha = \arg\min_{p \in K} ||\overline{vp}||, \tag{13}$$

where, $K = \{x | x \text{ is the intersection of } \overrightarrow{F_v} \text{ and } \bar{e}, \forall e \in \partial N(v)\}$, $\bar{e}$ is the extension line containing e, $\partial N(v)$ is the boundary of $N(v)$, and all the boundary edges whose extension lines contain v_α form the set $C_\alpha(v)$. Then it determines different edges for flipping according to three different cases: 1) v_α is on $C_\alpha(v)$ except its ends (Figure 4 (a)); 2) v_α is not on $C_\alpha(v)$ (Figure 4 (b)); and 3) v_α is on the ends of $C_\alpha(v)$.

For conservative consideration, we assume that all vertices except the current moving vertex are fixed, and all the new vertices generated by edge-split operations are free. To handle the cases 1) and 2), we use the same rule as in [LYY08] for choosing the edges, namely splitting e_{ab} in Figure 4 (a) and e_{vc} in Figure 4 (b) (the detail explanation can be seen in the appendix). As for case 3), we take it as the combination of cases 1) and 2), which needs to split 4



(a) Case 1) (b) Case 2)

Figure 4: Illustration of the first two cases.

related edges. For special cases when $|C_\alpha(v)| > 1$, we also use the same processing as in [LYY08].

5. Experiments

We implemented the algorithm in C++ and ran it on a computer with 8-core CPU clocked at 3.40GHz and 12GB memory. We used the LLT decomposition method of CHOLMOD library [CDHR08] to solve the linear system in the optimization, and OpenMesh library [BSBK02] for remeshing operations. Besides, we chose Dirichlet [EDD*95], LSCM [LPRM02], ARAP [LZX*08] and Green's Strain [HG00] as the distortion energies in Equation 5, and experimented on 2D triangle meshes with deformation applications to show its advantages. In all the results, the weight of the positional constraints is fixed: $\alpha = 5 \times 10^7$ and the barrier parameter β is set 100 for Green-Strain energy and 0.01 for others. In the following, we will show our results and compare them with the state-of-the-art methods.

We first show the advantage of our parameter-free relaxation enhanced geometry solver to the methods [Lip12, AL13] and [SKPSH13].

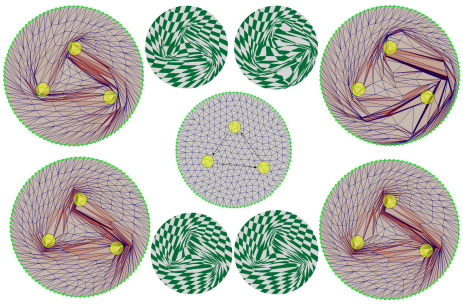


Figure 5: Comparison of our geometry solver on a coarse disk shape with "LSCM" energy (right bottom) to methods [Lip12] (left top), [AL13] (right top) and [SKPSH13] (left bottom). The middle disk is the rest mesh with constraints. The four small meshes textured with checkerboard are used to visualize their distortions. Here and the below, the bounded conformal distortions for the later two methods are set to the maximal distortions of our result. All the methods achieve the foldover-free results.

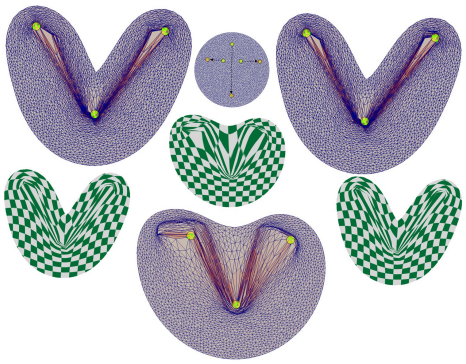


Figure 6: Comparison of our geometry solver applied on a dense disk shape (top right) with "Dirichlet" energy to the methods [SKPSH13] (top left) and [AL13] (bottom). All methods achieve the foldover-free results while [Lip12] doesn't converge in this deformation.

To compare the performance to the interior point method [SKPSH13], we evaluate the most time-consuming steps: decomposing Hessian and solving the Newton's step. We can see from Table 1 that, for mild deformation (e.g. in Figure 5), our method has slight advantages over [SKPSH13], since the Hessian is usually well enough for small amount of regularizations in the iterations for both methods. However, for extreme deformation (e.g. in Figure 6), the relaxation subroutine of our algorithm begins to take great effect in improving the convergence and it is usually able to find a solution with lower distortions and faster convergence rate. Note in the table that our method needs much less number of iterations of the two costly steps to obtain a gradient convergence solution.

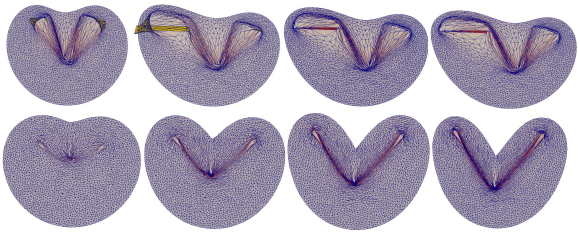


Figure 7: Deformation sequences obtained from the iterations for the dense disk shape with the method [AL13] (top row) and ours (bottom row). The fold-over triangles are colored in yellow. Note that our method produces smooth deformations, but the shape in the sequence from the method [AL13] may change abruptly even with fold-over triangles.

Comparing to the convexification methods [Lip12, AL13], our method has several advantages. First, our result distributes distortions more smoothly. In Figure 5, our result has lower average conformal distortion, which is 8.4

comparing to 16.5 ([Lip12]) and 36.4 ([AL13]) respectively. Our method is also efficient. It solves linear systems in each iteration via the efficient "Cholesky decomposition", and costs 0.41s for the coarse disk (Figure 5) and 4.5s for the dense disk (Figure 6). For both methods [AL13] and [Lip12], they need to solve a quadratic program in each iteration. The method [AL13] costs 4.2s (11 iterations) and 21.5s (5 iterations) respectively for the two meshes (the time is evaluated on the matlab platform). And the method [Lip12] costs 7.2s (9 iterations) for the coarse disk, but fails to converge for the dense disk. Next, due to the the nature of the interior point method and good property of the barrier function in Equation 5, our method always searches a solution in the feasible region and can generate a smooth deformation sequence during the iterations, which can not be achieved for both [Lip12] and [AL13] (seen the Figure 7). This property makes our algorithm potentially useful in the applications such as animation and morphing. Finally, our methods can flexibly choose non-linear distortion energies, such as Green's Strain energy (see Figure 10), which becomes a restriction of the methods [Lip12, AL13].

Usually, the relaxation enhanced geometry solver is sufficient to handle mild deformations. When the edge-flip operations are integrated into the solver, it equips the ability to reduce the energy and keep a reasonable condition number and thus speed up the convergence further.

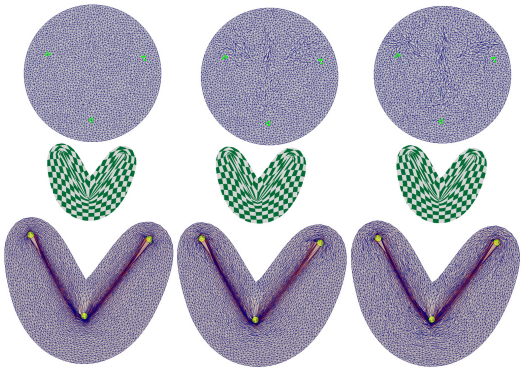


Figure 8: Comparison results with different values of η. Meshes on the top row are the rest shapes with remeshing, and the bottom row are their deformed results with the same constraints as in Figure 6. From left to right, $\eta = 10.0\%, 1.0\%, 0.1\%$.

The new solver has two parameters which affect its convergence and results: η and δ. The parameter η controls the number of edges to be flipped. Figure 8 shows results of the same deformation as in Figure 6 but with different values of η ($\eta = 10.0\%, 1.0\%, 0.1\%$). Note that as η varies from large to small, the number of flipped edges increases, and the maximal conformal distortions as well as the convergence rate vary differently. We found that too small or too large value of η may weaken the convergence and choosing the value between $0.1\% - 1.0\%$ works well.

Another parameter in our method, δ, provides additional control on the low bound of area, which can prevent degenerate triangles with prescribed values. We show the effect of using different δ in Figure 9. In our experience, too large or too small values of δ may slow down the convergence, since large values may shrink the solution space and small values may easily produce nearly degenerated elements and thus make the Hessian bad-conditioned.

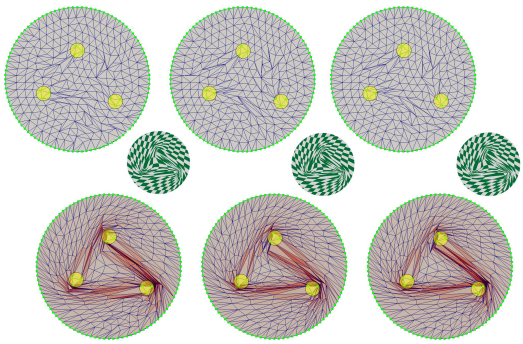


Figure 9: Results with different low-bound of area distortions. Meshes on the top row are the rest shapes after remeshing, and the bottom row are their deformed results with the same constraints as in Figure 5. From left to right, $\delta = 10^{-5}, 10^{-3}, 10^{-1}$.

Figure 10 demonstrates two other examples which validate the benefits of the edge-flip assisted solver.

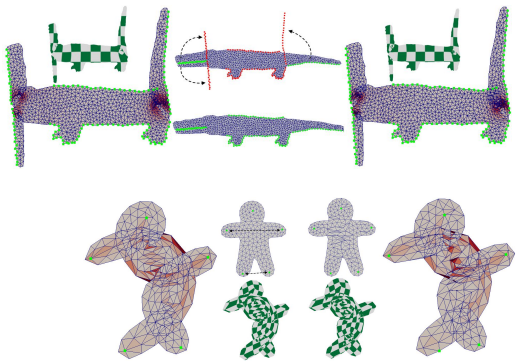


Figure 10: Results of our edge-flips assisted solver for the deformation of the alligator (top row) and woody (bottom row) shape. The left and right column show the results without and with edge-flip operations.

For some large deformations with extreme constraints, the solely geometry based solver may not obtain an ideal solution with constraints satisfied. Our edge-flip assisted solver presents its unique advantages. Figure 11 shows two such

examples, where both methods [Lip12] and [AL13] as well as the our solely geometry based solver fail to obtain a solution with constraints satisfied. Generally speaking, the edge-flip assisted solver has larger conformal distortions than the solely geometry based methods when they all converge with gradient convergence criterion (Table 1). However, since the edge-flip strategy can largely improve the behavior of convergence, it is easier to find a gradient convergence solution than the solely geometry solvers in extreme cases, and thus can obtain lower distortion results with faster convergence rate.

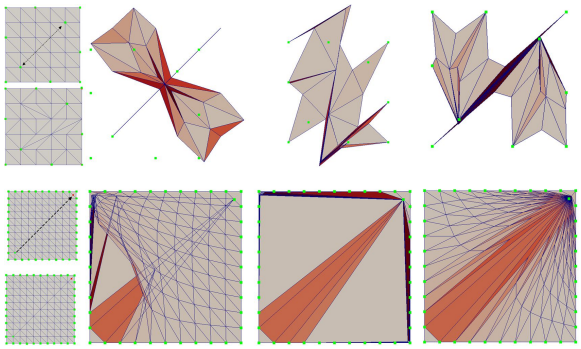


Figure 11: Comparison results of two extreme deformations with our edge-flip assisted solver (the last column) to [Lip12] (the top of the first column), [AL13] (the bottom of the second column), and our solely geometry based method (the third column). The first column displays the rest meshes before and after remeshing.

Our edge-flip based solver is able to handle various meshes with extreme constraints. But for some coarse meshes with more extreme positional constraints (see the example in the top middle of Figure 12), it becomes hard to satisfy the constraints. Thus we need edge-split as the locally up-sampling operation to further break the tessellation restrictions. In the figure, the left deformation is obtained through the first inner iteration with constraints unsatisfied (the iteration terminates with step convergence criterion, and the number of Hessian decomposition and vertex updating are: 163-D/78-U). After some edge-split operations, it is then deformed into the shape on the right by the solver with gradient convergence ((84-D/38-U)).

To see the convergence behavior of each method, we plot graphs of the energy value and its gradient norm in each iteration in Figure 13. In the figure, we can see that the edge-flip assisted solver has remarkable advantages than others, especially for the coarse mesh. For the performance, though we have to rebuild the Hessian when edge-split operations are applied in each inner iteration, it consumes much less time than solving the Newton's step due to its fast global convergence rate. In our edge-flip strategy, the condition number criterion plays an important role in the convergence of the algorithm. We tested our algorithm without this criterion on

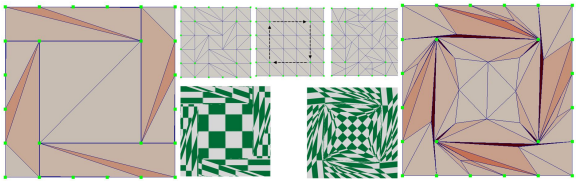


Figure 12: A deformation applied with LSCM energy requires edge-split operations. The left is the result with edge-flip assisted solver; the right is the result with edge-split assisted solver; three small meshes on the middle top are the rest meshes with flips, without remeshing and with both flips and splits; And the middle bottom shows their textured results.

both the coarse (Figure 5) disk and the dense disk (Figure 6), and found that both of them don't converge. The result implies that flipping an edge can reduce the energy but may have its negative impact on the convergence as well, and our condition number criterion largely improves its performance.

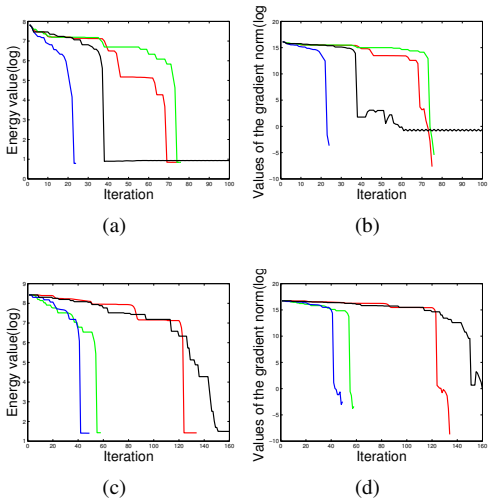


Figure 13: Convergence plots of E and $\|\nabla E\|_2^2$ on the coarse (a,b) and dense (c,d) disks with constraints shown in Figure 5 and 6 respectively. (Red, method of [SKPSH13]; Green, our method without edge-flips; Blue, our method with edge-flips; Black, our method without the condition number criterion).

6. Conclusion

We proposed a remeshing-assisted optimization method for 2D locally injective mappings. It applies a variant of "interior point method" based optimization with a parameter-free relaxation strategy and integrates adaptive edge-flip and

Table 1: Statistics for the results. 'M/A' denotes the maximal and average conformal distortions; 'CI' means convergence information ('G' - gradient convergence;'S' - step convergence;'M' - not converge); 'H/U' represents the number of Hessian factorization and vertex updating; 'Ours(NF)' is our method without edge-flip; and 'Ours(F)' is our method with edge-flip.

Mesh/Energy	Method	δ/η	M/A	$\|P(u)\|_2^2$	CI	H/U	Time(s)
Coarse disk (233V/542F; LSCM)	[SKPSH13]	10^{-5}/No	81.4/8.4	3.9×10^{-14}	G	214/76	0.44
	Ours(NF)	10^{-5}/No	81.4/8.4	3.9×10^{-14}	G	187/77	0.41
	Ours(F)	$10^{-5}/10^{-3}$	92.0/7.4	2.3×10^{-14}	G	54/25	0.19
	Ours(F)	$10^{-3}/10^{-3}$	90.8/7.5	2.7×10^{-14}	G	54/24	0.19
	Ours(F)	$10^{-1}/10^{-3}$	146.8/7.0	2.5×10^{-14}	G	72/31	0.21
Dense disk (2483V/4844F; Dirichlet)	[SKPSH13]	10^{-5}/No	193.3/5.1	1.7×10^{-13}	G	356/135	9.8
	Ours(NF)	10^{-5}/No	177.5/5.1	2.2×10^{-13}	G	144/59	4.5
	Ours(F)	$10^{-5}/10^{-3}$	731.6/5.2	7.4×10^{-14}	G	121/52	4.6
	Ours(F)	$10^{-5}/10^{-2}$	1287.8/5.0	7.1×10^{-14}	G	113/50	4.5
	Ours(F)	$10^{-5}/10^{-1}$	167.8/5.0	7.2×10^{-14}	G	188/77	6.2
Alligator (928V/1236F; Green-Strain)	[SKPSH13]	10^{-1}/No	7.7/1.3	1.2×10^{-11}	G	145/55	0.68
	Ours(NF)	10^{-1}/No	7.7/1.3	1.2×10^{-11}	G	61/27	0.4
	Ours(F)	$10^{-1}/10^{-2}$	9.7/1.3	8.2×10^{-12}	G	37/17	0.38
Woody (200V/333F; ARAP)	[SKPSH13]	10^{-3}/No	15.0/2.4	1.4×10^{-16}	G	1277/452	1.23
	Ours(NF)	10^{-3}/No	15.0/2.4	1.4×10^{-16}	G	1028/428	0.14
	Ours(F)	$10^{-3}/10^{-2}$	14.9/2.3	1.2×10^{-12}	G	676/278	0.09
Dense grid (212/200; LSCM)	[SKPSH13]	10^{-2}/No	14389.0/2041.2	7.6×10^{-4}	G	894/394	0.26
	Ours(NF)	10^{-2}/No	14390.0/2074.2	7.6×10^{-4}	G	755/333	0.26
	Ours(F)	$10^{-2}/10^{-3}$	2800.9/53.3	1.1×10^{-11}	G	51/26	0.08
Coarse grid (36V/50F; ARAP)	[SKPSH13]	10^{-3}/No	2680.0/395.1	8.9×10^{-8}	M	3207/1000	0.5
	Ours(NF)	10^{-3}/No	55020.0/8015.3	7.8×10^{0}	M	6942/2856	1.14
	Ours(F)	$10^{-3}/10^{-3}$	3270.0/302.2	2.4×10^{-12}	G	2161/843	0.37

edge-split operations to the geometry solver. Our method has the ability to break the tessellation restrictions, expand the feasible region as well as reduce the energy and its condition number, and thus is able to speed up the convergence and obtain an ideal solution. The results show that our remeshing-assisted approach has better performance than the solely geometry based optimization, especially for large deformation with extreme constraints.

Our method, like the method [SKPSH13] may fail in the case when deforming one shape to another with very different boundaries. An example can be seen in Figure 14, where the disk shape fails to be deformed into the woody shape with our method but is successful with the method [Lip12]. This is possibly due to the behavior of the interior point method, which solves a smooth deformation step by step, making the paths of the moving vertices crossing with each other.

Another limitation is that our method can only handle 2D planar meshes, which restricts its applications. We will extend the method to surface and 3D volume meshes and use it to solve other geometry problems in the near future.

Besides, it could be interesting to adopt a hierarchical strategy to compute mappings on dense meshes for efficiency, which finds a solution on low resolution meshes with our solver and obtains the solution of the original problem with the multiresolution technique.

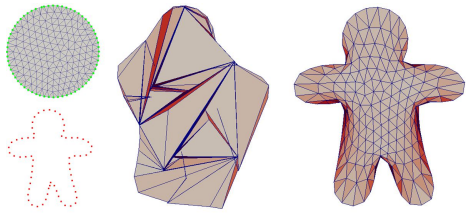


Figure 14: A failed example with our method, where the boundary of the disk shape is required to target to the boundary of the woody shape (the first column). Our method obtains a unsatisfying result (the second column) while the method [Lip12] performs the deformation successfully (the last column).

Acknowledgement

We thank the Interactive Geometry Lab (IGL) for providing some meshes used in the paper (http://www.igl.ethz.ch/),

Yaron Lipman for sharing his matlab code of the algorithms [Lip12, AL13] and the anonymous reviewers for their valuable comments. This work is supported by the funding from National Natural Science Foundation (No. 61170141) of China. Ruofeng Tong is partially supported by the National Basic Research Program (No. 2011CB302205) of China. Jin Huang is partially supported by the National High Technology Research and Development (863) Program of China (No.2012AA011503) and the Fundamental Research Funds for the Central Universities (2014XZZX006-08).

References

[AL13] AIGERMAN N., LIPMAN Y.: Injective and bounded distortion mappings in 3d. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 106. 1, 2, 6, 7, 8, 10

[AZF13] ATHANASIADIS T., ZIOUPOS G., FUDOS I.: Efficient computation of constrained parameterizations on parallel platforms. *Computers & Graphics 37*, 6 (2013), 596–607. 2

[BSBK02] BOTSCH M., STEINBERG S., BISCHOFF S., KOBBELT L.: Openmesh-a generic and efficient polygon mesh data structure. OpenSG Symposium 2002. 6

[BZK09] BOMMES D., ZIMMER H., KOBBELT L.: Mixed-integer quadrangulation. *ACM Transactions on Graphics 28*, 3 (2009), 77–77. 2

[CDHR08] CHEN Y., DAVIS T. A., HAGER W. W., RAJAMANICKAM S.: Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS) 35*, 3 (2008), 22. 6

[EDD*95] ECK M., DEROSE T., DUCHAMP T., HOPPE H., LOUNSBERY M., STUETZLE W.: Multiresolution analysis of arbitrary meshes. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM, pp. 173–182. 6

[ESG01] ECKSTEIN I., SURAZHSKY V., GOTSMAN C.: Texture mapping with hard constraints. *Computer Graphics Forum 20*, 3 (2001), 95–104. 3

[Flo03] FLOATER M.: One-to-one piecewise linear mappings over triangulations. *Mathematics of Computation 72*, 242 (2003), 685–696. 1, 2

[FM98] FUJIMURA K., MAKAROV M.: Foldover-free image warping. *Graphical Models and Image Processing 60*, 2 (1998), 100–111. 3

[GGT06] GORTLER S. J., GOTSMAN C., THURSTON D.: Discrete one-forms on meshes and applications to 3d mesh parameterization. *Computer Aided Geometric Design 23*, 2 (2006), 83–112. 2

[HG00] HORMANN K., GREINER G.: *MIPS: An efficient global parametrization method*. Tech. rep., DTIC Document, 2000. 1, 2, 6

[ITF04] IRVING G., TERAN J., FEDKIW R.: Invertible finite elements for robust simulation of large deformation. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2004), Eurographics Association, pp. 131–140. 2

[KS04] KRAEVOY V., SHEFFER A.: Cross-parameterization and compatible remeshing of 3d models. *ACM Transactions on Graphics (TOG) 23*, 3 (2004), 861–869. 3

[KSG03] KRAEVOY V., SHEFFER A., GOTSMAN C.: Matchmaker: constructing constrained texture maps. *ACM Transactions on Graphics (TOG) 22*, 3 (2003), 326–333. 1, 3

[Lip12] LIPMAN Y.: Bounded distortion mapping spaces for triangular meshes. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 108. 1, 2, 6, 7, 8, 9, 10

[Lip13] LIPMAN Y.: Construction of injective mappings of meshes. *CoRR abs/1310.0955* (2013). 2

[LPRM02] LÉVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (TOG) 21*, 3 (2002), 362–371. 6

[LYY08] LEE T.-Y., YEN S.-W., YEH I.-C.: Texture mapping with hard constraints using warping scheme. *IEEE Transactions on Visualization and Computer Graphics 14*, 2 (2008), 382–395. 1, 3, 5, 6, 10

[LZX*08] LIU L., ZHANG L., XU Y., GOTSMAN C., GORTLER S.: A local/global approach to mesh parameterization. *Computer Graphics Forum 27*, 5 (2008), 1495–1504. 2, 6

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *Eurographics Symposium on Geometry Processing* (2007), Eurographics Association, pp. 109–116. 2

[SHSF13] SCHNEIDER T., HORMANN K., S. FLOATER M.: Bijective composite mean value mappings. *Computer Graphics Forum 32*, 5 (2013), 137–146. 2

[SHST12] STOMAKHIN A., HOWES R., SCHROEDER C., TERAN J. M.: Energetically consistent invertible elasticity. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2012), Eurographics Association, pp. 25–32. 2

[SKPSH13] SCHÜLLER C., KAVAN L., PANOZZO D., SORKINE-HORNUNG O.: Locally injective mappings. *Computer Graphics Forum (proceedings of EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing) 32*, 5 (2013), 125–135. 1, 2, 3, 4, 6, 8, 9

[SLMB05] SHEFFER A., LÉVY B., MOGILNITSKY M., BOGOMYAKOV A.: Abf++: fast and robust angle based flattening. *ACM Transactions on Graphics (TOG) 24*, 2 (2005), 311–330. 2

[SSGH01] SANDER P., SNYDER J., GORTLER S., HOPPE H.: Texture mapping progressive meshes. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM, pp. 409–416. 1, 2

[TBI97] TREFETHEN L. N., BAU III D.: *Numerical linear algebra*, vol. 50. Siam, 1997. 5

[Tut63] TUTTE W. T.: How to draw a graph. *Proc. London Math. Soc 13*, 3 (1963), 743–768. 1, 2

[WMZ12] WEBER O., MYLES A., ZORIN D.: Computing extremal quasiconformal maps. *Computer Graphics Forum 31*, 5 (2012), 1679–1689. 2

[XCGL11] XU Y., CHEN R., GOTSMAN C., LIU L.: Embedding a triangular graph within a given boundary. *Computer Aided Geometric Design 28*, 6 (2011), 349–356. 2

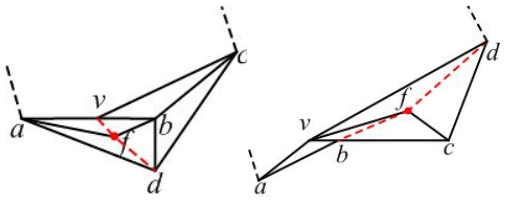
[YLY*12] YU H., LEE T.-Y., YEH I.-C., YANG X., LI W., ZHANG J. J.: An rbf-based reparameterization method for constrained texture mapping. *IEEE Transactions on Visualization and Computer Graphics 18*, 7 (2012), 1115–1124. 2

Appendix

In the appendix, we present the edge-split operations used in the method, which are highly inspired by [LYY08].

The case when v_α is inside $C_\alpha(v)$. An illustration is

shown in Figure 4 (a), where $C_\alpha(v_A) = \{e_{ab}\}$. Note that when v moves across the edge e_{ab} along the direction $\overrightarrow{F_v}$, the colored triangle Δvab will fold over. To prevent the folding, we split the vertex's opposite edge e_{ab} into two segments. Then v can move to the position of v_α without foldovers as is shown in Figure 15 (a).

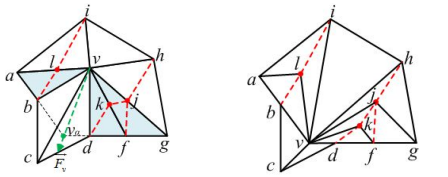


(a) (b)

Figure 15: Results of handling the cases 1) (a) and 2) (b).

The case when v_α is outside $C_\alpha(v)$. An example of this case is given in Figure 4 (b) where $C_\alpha(v_A) = \{e_{bc}\}$. Note that splitting the opposite edge is useless in this case, since the colored triangle Δvbc will fold over first as the vertex v moves along the direction $\overrightarrow{F_v}$. Instead, we split another edge e_{vc} and then freely move v to v_α without foldovers (see Figure 15 (b)).

Other cases. The above two cases consider situations when $|C_\alpha(v)| = 1$. However, it is also possible when $|C_\alpha(v)| > 1$. In these cases, we find our all the connected components, and split related edges of each connected component in order using the similar strategy above. An example is illustrated in Figure 16. In Figure 16 (a), $C_\alpha(u) =$



(a) Rest-pose mesh (b) Deformed mesh

Figure 16: Illustration of a case when $C_\alpha(u) > 1$.

$\{e_{gf}, e_{fd}, e_{ba}\}$, and v_α is not on $C_\alpha(v_A)$. This situation can be regarded as an extension of the second case above, in which moving the vertex v to v_α will cause the triangles $\Delta vab, \Delta vdf, \Delta vfg$ fold-over. To handle it, we divided the edges into two connected components, $C_1 = \{e_{ab}\}$ and $C_2 = \{e_{df}, e_{fg}\}$, determine all the potential edges using the rule described in the second case for each component, and split them from the farthest to the nearest one from v_α. Note that the newly generated vertices increase the degrees of freedom to warp the 1-ring neighbor to a convex-like region which makes it possible to move v to the position v_α without foldovers (Figure 16 (b)).