Progressive Embedding

HANXIAO SHEN, New York University
ZHONGSHI JIANG, New York University
DENIS ZORIN, New York University
DANIELE PANOZZO, New York University

Tutte embedding is one of the most common building blocks in geometry processing algorithms due to its simplicity and provable guarantees. Although provably correct in infinite precision arithmetic, it fails in challenging cases when implemented using floating point arithmetic, largely due to the induced exponential area changes.

We propose Progressive Embedding, with similar theoretical guarantees to Tutte embedding, but more resilient to the rounding error of floating point arithmetic. Inspired by progressive meshes, we collapse edges on an invalid embedding to a valid, simplified mesh, then insert points back while maintaining validity. We demonstrate the robustness of our method by computing embeddings for a large collection of disk topology meshes.

By combining our robust embedding with a variant of the matchmaker algorithm, we propose a general algorithm for the problem of mapping multiply connected domains with arbitrary hard constraints to the plane, with applications in texture mapping and remeshing.

1 INTRODUCTION

Piecewise linear surface-to-plane maps, or parametrizations, are ubiquitous in computer graphics, geometry processing, mechanical engineering, and scientific visualization. Depending on the applications, the maps are required to exhibit different properties, most commonly, low distortion, local injectivity, and global bijectivity.

The last two properties are challenging to guarantee for discrete maps. Most algorithms with guarantees use Tutte embedding as a component. Tutte embedding is a construction that is guaranteed to create bijective mappings under minimal assumptions, if both domains are simply connected and the target planar domain is convex. However, the guarantee only holds if the computation is performed in arbitrary precision rather than floating point arithmetic, as it is commonly done. Failure due to floating point approximation is not as uncommon as one would assume, as the algorithm is likely to create an extreme variation of scale and aspect ratios in complex mapping cases. To quantitatively evaluate this issue, we computed Tutte embeddings on 2718 models (all the genus 0 models from Thingi10k [Zhou and Jacobson 2016]) using double precision, and observed 80 failures. To the best of our knowledge, this problem has not been addressed before in the literature.

This rate of failure is problematic for batch processing large geometrical collections (for example for processing geometric deep learning datasets) or when the embedding has to be computed many times (for example in cross-parametrization [Kraevoy et al. 2003; Schreiner et al. 2004]). In these scenarios, a failure rate of 2.9% may not be tolerable, since it is not realistic to manually fix hundreds of problematic cases, and if failure happens on large meshes with millions of triangles it might not even be possible to fix them by hand.

A simple solution to this problem is the use of multi-precision (or rational) arithmetic [Granlund 2018]: if enough bits are used to represent the mantissa and exponent of the floating point representation, Tutte embedding will succeed, since the solution of a linear system can be computed exactly. However, the result in high precision is not directly usable by downstream applications, and requires to be rounded (or “snapped” [Halperin and Packer 2002]) to floating point coordinates. This is a surprisingly challenging problem for which, to the best of our knowledge, no solution applicable to our setting exists (Section 3.1).

Instead, we propose a progressive algorithm to directly generate an embedding using floating point coordinates. We start from an initial, possibly invalid, floating point planar parametrization, and

Fig. 1. The Tutte embedding of this Hele-Shaw polygon (left) contains 46 flipped triangles, due to numerical rounding errors. Our progressive embedding (right) produces a valid embedding, without any inverted element and with lower distortion. The colors represent the distortion of the triangles, measured using the symmetric Dirichlet energy.
we make it valid by collapsing all flipped and degenerate parts of the (possibly invalid) embedding produced, e.g., by a floating-point Tutte algorithm. We re-insert one vertex of the original mesh at a time, preserving the validity of the map at every step. This approach is inspired by [Schreiner et al. 2004], which proposes a progressive algorithm for computing cross-parametrizations based on progressive meshes [Hoppe 1996]. Our algorithm differs since we do not know a valid position for the inserted vertices, and we thus have to compute it as the vertices are added back. We provide a formal proof of correctness of our method in arbitrary precision (obtaining the same formal guarantees as Tutte embedding), and we practically demonstrate its superior robustness by parametrizing a large collection of 10k models.

Using our new embedding method and the matchmaker algorithm [Kraevoy and Sheffer 2004; Kraevoy et al. 2003] as a foundation, we develop an algorithm for mapping between multiply-connected domains with arbitrary constraints, supporting fully general self-overlapping domains as the target. We experimentally show that our algorithm is very robust, producing valid and distortion-optimized maps even for challenging cases where the original matchmaker algorithm fails due to numerical problems. We demonstrate the practical utility of our algorithm for UV mapping and quadrangulation applications.

To foster replicability of results and to maximize the practical impact of our algorithm, we also attach a reference implementation. https://github.com/hankstag/progressive_embedding

2 RELATED WORK

2.1 Planar Embedding of Graphs and Meshes

Fáry’s theorem [Fáry 1948] states that any planar graph can be embedded in the plane with straight edges. Tutte [Tutte 1963] extends this result to the case of fixed convex boundary with a spring analogue, and [Floater 1997] established its connection to the parameterization methods in the geometry processing community, and extend Tutte’s uniform weight to arbitrary positive ones. In both cases, the problem is reduced to solving a linear system of equations and the resulting embeddings’ minimum area might even be negative exponential with respect to the number of vertices. There has been active effort in the graph drawing community to address these issues, by bounding the total area when drawing on integer grids, or equivalently, controlling the minimum resolution [Chambers et al. 2011] under fixed diameter. Most notably, [Schnyder 1990] shows an algorithm to embed a planar graph onto integer grids inside a triangle region, and [Chambers et al. 2011] proves an upper polynomial bound on the area while keeping a specified convex boundary shape: the proof is constructive and may (potentially) be used as a basis for a practical algorithm. In all cases, rounding problems will affect these algorithms as the size of the graph grows (Section 3.1).

Orbifold Tutte Embedding. Multiple extensions of Tutte’s theorem to map surfaces to different co-domains have been proposed. In particular, the theorem has been extended to map surfaces to a Euclidean orbifold [Aigerman and Lipman 2015], to a hyperbolic orbifold [Aigerman and Lipman 2016], and to a spherical orbifold [Aigerman et al. 2017]. All three methods support hard positional constraints and ensure the generation of a bijective map between the surface and the orbifold in infinite precision arithmetic. These methods also suffer from similar numerical issues as Tutte’s, and extending our algorithm to orbifold embeddings is an interesting direction for future work.

2.2 Progressive Meshes

The well-known progressive meshes algorithm [Hoppe 1996; Sander et al. 2001] shows how a triangle mesh can be simplified by collapsing one edge at a time, and reconstructed applying the inverse topological operations in the inverse order. This scheme has been introduced as an efficient way to store, transmit, and render large meshes, where the per-vertex properties of the removed vertex are stored together with the information required to insert them back. This work has been later applied to compute inter-surface mappings [Schreiner et al. 2004], by jointly simplifying two meshes into a common base mesh, then starting from optimizing their isometric distortion while reinserting the vertices in the base mesh.

We use the same idea to eliminate problematic regions of an existing embedding (either flipped, or with a high distortion), and then reinserting one vertex at a time, while preserving the quality of the triangulation. Differently from progressive meshes, in our case we do not have geometrical information available that could help us decide where the vertex should be inserted to obtain a valid embedding.

2.3 Distortion-Minimizing Mappings

In this section, we focus on the recent works closely related to generating distortion-minimizing discrete locally injective and globally bijective discrete maps, and we refer to [Floater and Hormann 2005; Hormann et al. 2007; Sheffer et al. 2006] for a comprehensive treatment of earlier parametrization methods without these properties.

A discrete locally injective map requires that triangles maintain their orientation (i.e. they do not flip) and if the sum of (unsigned) triangle angles around each internal vertex is precisely $2\pi$ [Weber and Zorin 2014]. Three main families of methods have been proposed to deal with this challenging constraint: barrier, convexification, and hybrid algorithms.

Barrier Algorithms. Barrier algorithms require a valid initial solution, and then optimize its quality without leaving the feasible space. The key idea is to adopt quality metrics diverging to infinity when triangles become degenerate, thus inhibiting flips. Popular choices strive to preserve angles [Degener et al. 2003; Hormann and Greiner 2000] or lengths [Aigerman et al. 2014; Foranne and Lipman 2014; Sander et al. 2001; Smith and Schaefer 2015; Sorkine et al. 2002]. Alternatively, a barrier functions can be added to existing energies to enforce local injectivity [Schüller et al. 2013]. These non-linear energies are difficult to minimize, stemming a series of methods specifically targeting this problem. They include coordinate descent [Hormann and Greiner 2000; Labsik et al. 2000], parallel gradient descent [Fu et al. 2015], Anderson Acceleration [Peng et al. 2018], as well as other quasi-newton approaches [Clain et al. 2017; Kovalsksy et al. 2016; Liu et al. 2018; Rabinovich et al. 2017; Shtengel et al. 2017; Smith and Schaefer 2015; Zhu et al. 2018].

All these methods support hard-constraints if they are already satisfied in the initial map, which is the key idea used in MatchMaker...
Our progressive embedding can be used to robustly generate the initial map, that can then be improved by any of the previous techniques (Section 5).

**Projection Algorithms.** An essential component of these methods is a convexified form of the injectivity constraints [Kovalsky et al. 2015; Lipman 2012]. While these methods naturally support hard injectivity constraints, they might fail to find a feasible solution, with no output generated. The only known way to guarantee that a feasible solution exists is to formulate the convexified constraints using a reference frame derived from a valid (although potentially very high distortion) solution.

**Hybrid Algorithms.** Hybrid algorithms are an interesting mix between these two approaches [Fu and Liu 2016; Poranne et al. 2017]. The initial guess is produced by separating all triangles and isometrically rotating them into the UV space. A barrier method is then used to prevent them from flipping, while trying to seal the seams. This approach might fail to seal all the seams, not producing a valid map.

**Globally Bijective Maps.** For simply connected domains, bijective maps are locally injective maps whose boundary does not intersect. All embeddings described in Section 2.1 satisfy this property. These methods have been extended to non-convex, self-overlapping polygons [Weber and Zorin 2014] and polyhedrons [Campen et al. 2016], but they still require a fixed boundary. Few methods can produce bijective maps while letting the boundary free, relying on either collision detection [Smith and Schaefer 2015] or scaffolding elements [Gotsman and Surazhsky 2001; Jiang et al. 2017; Müller et al. 2015; Zhang et al. 2005]. All free boundary methods require a starting point: our algorithm can be used to generate it, enabling these algorithms to create bijective maps with hard constraints (Section 4).

**Hard Positional Constraints and Refinement.** Matchmaker [Kraevoy et al. 2003] introduced hard positional constraints for texture mapping applications. The algorithm uses a two-step approach, first generating a valid map, and then optimizing its geometrical quality. The method is one of the few using refinement to guarantee the existence of feasible solutions. The method has been extended by adding an intermediate warping stage to align the constraints in [Lee et al. 2008]. We show in section 4 how our embedding can be used within Matchmaker to increase its robustness, and we also show how to extend Matchmaker to support self-overlapping polygonal target domains.

**Cross-Parametrization.** Cross-parametrization, i.e. the computation of a map between two surfaces, is another problem that often relies on planar embeddings. [Schreiner et al. 2004] and [Kraevoy and Sheffer 2004] proposed the first provably guaranteed solutions to compute maps between surfaces, by reducing the problem to mapping both surfaces to a common subdomain by either using Tutte’s embedding or a simplification approach. A similar construction that cuts open the surface into a single topological disc has been proposed in [Aigerman et al. 2014], and extended to allow even the optimization of the seams positions in [Aigerman et al. 2015]. Floating point rounding errors have not been considered in any of these works, which are more prone to fail as the resolution of the mesh increase or whenever the user-provided constraints introduce a high distortion (Section 5).

**Global Parametrization.** Field-aligned parametrization methods [Bommes et al. 2009] strive to compute a locally injective map [Bommes et al. 2013] whose gradient is aligned with a user-provided directional field. We refer an interested reader to [Bommes et al. 2012] for a comprehensive overview of these techniques. Our embedding algorithm can be used to compute parametrizations to a target self-overlapping polygon, enabling to robustly generate these parametrization if a valid boundary polygon is provided (Section 5).

3 PROGRESSIVE EMBEDDING

3.1 Analysis of Tutte Embedding in Floating Points

We discuss in detail when Tutte embedding implemented in floating point may fail, and also show that straightforward solutions with off-the-shelves geometry processing tools do not solve these issues.

**Tutte Embedding implemented in Floating Points.** We use the implementation of Tutte embedding in libigl [Jacobson et al. 2016], and apply it to all the 2718 genus 0 models of the Thingi10k dataset [Zhou and Jacobson 2016], after cleaning them up and improving their quality using TetWild [Hu et al. 2018], to ensure that no degenerate triangles are present. We also ensure that the meshes are 3-connected by refining them locally. For every model, we randomly pick and delete a triangle, and map the resulting boundary to an equilateral triangle. We compute the Tutte embedding, and check for flips using CGAL’s exact floating point predicates [Brönnimann et al. 2018]. The check fails for 80 models, due to the numerical errors introduced in the mapping. In retrospect, this is not surprising since it is well-known that Tutte planar drawing may admit exponential area when drawing on integer grids. Two problematic cases are shown in Figure 2, where the embedding introduces a large variation of scale, and the flip occurs on triangles with small areas.

**Multi-Precision Tutte Embedding with Snap Rounding.** A straightforward way to address this problem is to increase the number of bits used in the floating point representation. We double the number of bits using the library MPFR [Fousse et al. 2007], which is directly...
integrated into Eigen, and can thus be used with the Tutte embedding in libigl with minimal code changes. With this setup, all the problematic cases are solved. However, the runtime is increased by around one order of magnitude, and, most importantly, the results generated cannot be rounded back to floating point since trivial rounding introduces flips. Snap rounding [Packer 2018] could be used to avoid them, but it will collapse possibly large regions of the mesh: 6.3% of the vertices of the model shown in the bottom left of Figure 8 are collapsed when using a snap rounding resolution of $10^{-16}$ times the diagonal of the bounding box of the embedding.

Multi-Precision Tutte Embedding with Quality Optimization. The problem with rounding to floats is induced by the small triangles (and correspondingly small edges) which leads to flips after snapping. A possible way to address this issue is to use a mesh optimization algorithm, using multi-precision representation, before rounding to floats. We tested two approaches: (1) SLIM [Rabinovich et al. 2017] adapted to run in multiprecision, and (2) minimizing the symmetric Dirichlet energy by moving one vertex at a time using coordinate descent [Hormann and Greiner 2000]. The first approach is prohibitively slow, due to the linear solve in high precision and the very small steps due to the elements with almost zero area. The second one succeeds on 57 models, but still fails on 23, even after 24 hours of running time.

3.2 Progressive Embedding

Our approach draws on the ideas of progressive meshes [Hoppe 1996] and inter-surface mappings [Schreiner et al. 2004], which are, in turn, closely related to theoretical ideas from PL topology (e.g., [Hudson and Shaneson 1969]).

Our algorithm does not require Tutte embedding and can be used to construct an embedding from scratch or from a random initial mapping (Figure 3). It can be accelerated by using an existing, possibly invalid, embedding as a starting point. A triangle is invalid if its signed area is negative, or if its quality measure is below a threshold (we use the symmetric Dirichlet energy [Smith and Schaefer 2015] with respect to a canonical equilateral triangle whose area is the area of the target boundary polygon divided by the number of triangles and mark invalid if it is above $\tau = 1e20$). Using a quality measure in addition to signed area is important, since triangles with small, positive areas might cause numerical problems during the vertex insertion (Phase 2 below).

Starting from an invalid embedding, our algorithm (1) performs edge collapses until the simplified mesh has no invalid triangles (Algorithm 1), and (2) progressively inserts back each vertex in the same order (Algorithm 2), with feasibility of insertion ensured at each step.

Stage 1: Simplification. At this stage, we iteratively find an interior edge that can be collapsed until all invalid elements are removed from the initial embedding, or a single interior vertex is left (Algorithm 1). A theorem in [Mijatović 2003] and our Theorem A.8 guarantees that a sequence of collapses reducing the mesh to a mesh...
with a single interior vertex can always be found; as the boundary embedding is convex, we can also always find a position for this vertex to create a valid embedding for the fully simplified mesh. The simplification algorithm starts by tagging all invalid triangles (Line 1), and attempts to collapse all their edges. If this procedure is successful in eliminating all invalid triangles, the algorithm terminates, otherwise all the triangles adjacent to tagged triangles are tagged (Line 12), and their edges collapsed. Note that we only allow edge collapses on internal edges to avoid changes to the boundary. This algorithm is guaranteed to terminate, since in the worst case it will tag the entire mesh and Theorem A.8 ensures that at least one edge will be collapsible. If only one internal vertex is left (Line 4), we move it to the barycenter of the boundary vertices (which is inside the convex boundary by construction, but might fail in degenerate cases, as discussed in Section 6). Once the algorithm terminates, the resulting simplified mesh has no inverted triangles and by Proposition A.9, also has sums of triangle angles at each vertex equal to $2\pi$.

Stage 2: Insertion. Starting from the valid embedding computed after Stage 1, we perform a sequence of splits reverting the collapses, while maintaining embedding validity at every step (Algorithm 2). Lemma A.11 ensures that this is always possible (in infinite precision), as the area of each triangle after insertion is always positive, whenever the newly inserted points lie in the valid sector (Figure 5). The algorithm first computes candidate directions in the valid sector, then performs a flip-avoiding line search [Smith and Schaefer 2015] (Line 3) to find candidate positions $P(v_m)$ for the newly inserted vertex $v_m$, such that the 1-ring neighborhoods are valid. In our experiments, we use step length $\alpha = 0.8$, and cap the number of line search iterations to 75. If at least one candidate is found, the split is performed using the candidate resulting in a valid sector, then performs a flip-avoiding line search [Smith and Schaefer 2015], optimizing one vertex position at a time using Newton iterations, similarly to [Fu et al. 2015; Hormann and Greiner 2000; Labsik et al. 2000]. We favor this local approach since it is more robust to low quality elements, which would otherwise badly affect both the numerical stability and the step size of global optimization methods. Since our goal is to improve the minimal quality of the mesh, we minimize the symmetric Dirichlet energy only for the invalid triangles (using reference shape as an equilateral triangle with area equal to the average of triangles in the 2D domain), and use the scaffold energy [Jiang et al. 2017] (i.e. we use the element itself as the reference triangle for the symmetric Dirichlet energy) for the valid ones, which allows them to move more freely. In our experiments, the local smoothing is performed for 10 iterations after every insertion step. If a valid insertion candidate cannot be found, we keep improving the quality with batches of 50 smoothing iterations until a candidate is found (Algorithm 2 line 11). Furthermore, at each smoothing phase, we perform a greedy coloring of the edge graph [Kucera 1991], and the vertices inside each color are optimized in parallel.

4 MATCHMAKER++

The computation of locally injective maps is important in geometry processing (Section 2), with the majority of the methods focusing on efficient and scalable quality optimization. However, few method guarantees positional constraints: the notable MatchMaker algorithm [Kraevoy et al. 2003] reduces the problem to a number of
convex planar embeddings, which are computed with Tutte’s algorithm. However, as we observe in some cases (Section 5.2), such embeddings can be numerically challenging. Replacing Tutte embedding with progressive embedding enables matchmaker to robustly compute maps with very challenging configurations of constraints.

In this Section, we describe an extension of [Kraevoy et al. 2003] that (1) makes use of progressive embedding to increase robustness, and (2) supports weakly self-overlapping polygons as co-domains [Weber and Zorin 2014].

Overview. Combining our progressive embedding algorithm and the matchmaker algorithm, we describe an algorithm for solving the following problem: Given a simply-connected 3d mesh, equipped with a set of user-defined hard positional constraints at vertices, compute a valid piecewise-linear parametrization, such that (1) the map is valid in the following sense: there are no flipped triangles, and for each vertex, the map restricted to the one ring of triangle of that vertex is bijective, unless it is a singular boundary vertex, as defined below and (2) the parametrization bitwise exactly satisfies the user-defined positional constraints. We tackle this in three steps: we decompose the target domain into convex polygonal subdomains, match these domains to the subdomains of the source domain, compute an initial bijective map by stitching progressive embeddings for each subdomain, and finally globally optimize the mapping distortion.

User input. We distinguish between two cases, chosen by the user (1) the required map is a global embedding, (2) the map is an immersion. For the first case, the constraint specification is more flexible: the user only has to provide a set of point or line constraints. For the second case, the target domain is not a subset of the plane, but rather, an everywhere flat surface with overlaps. We require the user to prescribe constraints for the whole boundary of the polygon to define the target domain unambiguously (some parts may be marked as movable, but an initial position is needed) and to provide a path connecting each point or line constraint to the boundary, which allows to define its location on the target surface implied by the boundary specification. In this case, target boundary polygon has to be weakly self-overlapping [Weber and Zorin 2014], otherwise, the map does not exist.

Phase 1: Subdivision of the Target Domain. In the global embedding case, the target domain is generated by triangulating the bounding box of the input with Triangle [Shewchuk 1996]. In the second case, the self-overlapping domain is triangulated using a modification of the Shor-Van Wyck algorithm [Shor and Van Wyck 1992], described in [Weber and Zorin 2014]. In both cases we ensure that the hard positional constraints are vertices of the triangulation. We then merge triangles into convex polygons in a greedy manner, by dropping edges if the resulting subdomains are convex. While this merging step is optional (the algorithm works also without it) it reduces the number of subdomains and vertices, making the next steps more efficient. In the case when an immersion is computed, by construction, it will be an embedding on subdomains computed starting from the Shor-Van Wyck triangulation.

Phase 2: Path Tracing on the Original Mesh. After the target domain is subdivided into convex polygons, we match this decomposition to the input mesh. The goal is to find non-intersecting paths connecting each of these pairs in 3D mesh, subdividing the whole mesh into same number of patches.

At the tracing stage, we perform a reordering of the paths to make sure that no previously traced path will block the future ones (Figure 7). In the case of embedding, the algorithm finds paths that connect all positional constraints to the boundary without creating additional loops (than the existing boundary). In practice, these paths are found by dropping one segment on the boundary, and then grow the minimum spanning tree (over the edges of the polygonal mesh) from the incomplete boundary loop. We first trace the paths on the minimum spanning tree and then the remaining ones, connecting the boundary to the constraints. The correctness of this procedure can be found in [Praun et al. 2001].

For the tracing of each path on the surface, we follow [Kraevoy et al. 2003] to find the shortest path connecting two endpoints, and add Steiner points on the edges if no path, not intersecting other paths, can be found.

Phase 3: Bijective Mapping. After establishing a correspondence between each patch of the 3D mesh and a convex polygon in the target domain, we can first subdivide the 2D paths in the target domain to match the number of vertices on the corresponding 3D path to obtain the one-to-one correspondence between them. We observe that up to this point, the algorithm is largely combinatorial (while some vertices are inserted on edges, their geometric position is trivially determined and is very unlikely to result in numerical problems; none were observed in our experiments). At this point, the map is defined for boundaries of the subdomains corresponding to the convex subdomains in the target. Next, we extend the map to the interior of each region using our progressive embedding algorithm (Section 3.2). Notice that when the mesh patch is not 3-connected, we need to split the edges with two endpoints on the boundary.

Phase 4: Quality Optimization. The map obtained in the previous steps is valid, according to the definition of the weakly self-overlapping map [Weber and Zorin 2014]. Therefore, its quality can be optimized using any locally injective map improvement algorithm (Section 2). We opt for [Rabinovich et al. 2017], since it is efficient for large models and the implementation is readily available [Jacobson et al. 2016]. The implementation is modified to support hard positional constraints, by eliminating the corresponding variables.

5 RESULTS AND DISCUSSION

Table 1. Statistics of the input and output meshes in the planar embedding test (Section 5.1). From left to right: Name of the dataset, number of vertices, number of faces, number of invalid elements (positive area, but with energy above \(1e20\)) after Tutte embedding, number of flipped elements after Tutte embedding, progressive embedding (Section 3) running time in seconds.

<table>
<thead>
<tr>
<th>Name</th>
<th>#V</th>
<th>#F</th>
<th>#invalid</th>
<th>#flipped</th>
<th>PE(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Octopus</td>
<td>5034</td>
<td>10063</td>
<td>2351</td>
<td>524</td>
<td>245.1</td>
</tr>
<tr>
<td>Swirl</td>
<td>11754</td>
<td>23503</td>
<td>9317</td>
<td>638</td>
<td>2273.1</td>
</tr>
<tr>
<td>Deer</td>
<td>8720</td>
<td>17434</td>
<td>15728</td>
<td>7831</td>
<td>3916.8</td>
</tr>
<tr>
<td>Rabbit</td>
<td>7253</td>
<td>14500</td>
<td>8743</td>
<td>4233</td>
<td>1198.6</td>
</tr>
<tr>
<td>HeleShaw</td>
<td>3505</td>
<td>5355</td>
<td>437</td>
<td>46</td>
<td>62.1</td>
</tr>
<tr>
<td>Retinal</td>
<td>3791</td>
<td>7282</td>
<td>3533</td>
<td>3533</td>
<td>95.0</td>
</tr>
<tr>
<td>Arch</td>
<td>973</td>
<td>1941</td>
<td>790</td>
<td>270</td>
<td>21.9</td>
</tr>
<tr>
<td>Propeller</td>
<td>787</td>
<td>1569</td>
<td>484</td>
<td>70</td>
<td>11.6</td>
</tr>
</tbody>
</table>

Table 2. Statistics of the input and output meshes of the MatchMaker++ test (Section 5.2). From left to right: Name of the dataset, number of vertices, number of faces, number of invalid elements (positive area, but with energy above \(1e20\)) after Tutte embedding, number of flipped elements after Tutte embedding, progressive embedding (Section 3) running time in seconds, and MatchMaker++ (Section 4) running time in seconds.

<table>
<thead>
<tr>
<th>Name</th>
<th>#V</th>
<th>#F</th>
<th>#invalid</th>
<th>#flipped</th>
<th>PE(s)</th>
<th>MM++(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fertility</td>
<td>16508</td>
<td>33028</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>582.4</td>
</tr>
<tr>
<td>3 holes</td>
<td>7440</td>
<td>14886</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>107.4</td>
</tr>
<tr>
<td>Robot Cat</td>
<td>4117</td>
<td>7512</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>0.8</td>
</tr>
<tr>
<td>Aircraft</td>
<td>2523</td>
<td>4656</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>0.5</td>
</tr>
<tr>
<td>Twirl</td>
<td>5562</td>
<td>10402</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>1.1</td>
</tr>
<tr>
<td>Filigree</td>
<td>49872</td>
<td>100000</td>
<td>32</td>
<td>0</td>
<td>72.4</td>
<td>30.8</td>
</tr>
<tr>
<td>Botijo</td>
<td>43786</td>
<td>83788</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>3.9</td>
</tr>
<tr>
<td>Beetle</td>
<td>20619</td>
<td>39276</td>
<td>0</td>
<td>0</td>
<td>NA</td>
<td>1.1</td>
</tr>
<tr>
<td>Casting</td>
<td>21236</td>
<td>39438</td>
<td>67</td>
<td>40</td>
<td>27.4</td>
<td>1.3</td>
</tr>
<tr>
<td>Oil pump</td>
<td>54135</td>
<td>103778</td>
<td>5</td>
<td>0</td>
<td>2.3</td>
<td>4.8</td>
</tr>
</tbody>
</table>

5.1 Progressive Embedding

**Planar Embedding for the Thingi10k Dataset [Zhou and Jacobson 2016].** By computing Tutte’s embedding for the genus-zero models in 2718 surface mesh models on a triangle boundary, we observed there are 80 cases where the generated parametrization has flipped elements due to floating point rounding errors. Using our progressive strategy, we are able to fix all failed cases. A selection of the parametrization results are shown in Figure 2.

Integration with OptCuts [Li et al. 2018]. OptCuts is a joint optimization method to create UV seam from a 3D model, balancing seam length and parametrization quality. It relies on a valid initialization, which for genus 0 model, is compute through randomly cutting two adjacent edges as seams, then flatten it on the plane with Tutte embedding. In Figure 8, we show two examples where this initialization fail. Both models can be processed if progressive embedding is used instead of Tutte embedding, allowing OptCuts to proceed and optimize the UV map.

**Mapping an Hele-Shaw Polygon to a Square.** Hele-Shaw flow is a two-dimensional Stokes flow of mixing liquids between two parallel flat surfaces separated by a small gap. In Figure 1, we show an example mesh generated using the Hele-Shaw simulation proposed in [Segall et al. 2016]. One way to compute a bijective map of the interior of the polygon between different frames is a cross-parametrization using a square as the common domain, with no internal constraints. Tutte embedding fails in this case, introducing 46 flipped faces (Figure 1, left), while progressive embedding produces a valid map with lower distortion (Figure 1, right).

5.2 Matchmaker++ [Kraevoy et al. 2003]

**Self-Overlapping Locally-Injective Maps.** By introducing Shor Van Wyck algorithm into the matchmaker pipeline, we are able to mapping a surface mesh with disk topology to self-overlapping boundaries as in [Weber and Zorin 2014]. Similarly to [Weber and Zorin 2014] our algorithm can generate locally-injective, self-overlapping parametrizations (Figure 9), which are commonly used by quadrangulation algorithms [Bommes et al. 2012].

Comparison with [Kovalsky et al. 2015]. We parametrized the global parametrization benchmark introduced in [Myles et al. 2014], using the seams in the obj files, and fixing in random positions 3 random points of each mesh. This is a challenging task, since the random constraints introduce a large distortion. Our method succeeded on all 102 models: a selection of the most challenging ones is shown in Figure 10. We also run the same experiment using the most recent projection method [Kovalsky et al. 2015] (which is one of the few methods that supports similar constraints without requiring a fully specified target domain), using LSCM [Lévy et al. 2002] as an initial guess. The method failed on 28 models over 102...
Fig. 9. Two seamless maps with hard positional constraints and fixed boundaries are generated by our algorithm.

Fig. 10. A selection of locally injective parametrizations computed by our algorithm by fixing 3 random points to 3 random points in UV space.

(27%). We show three failed cases using their method with flipped elements in the output, and the quality is considerably lower than our approach, as shown in Figure 11. Note that this is a comparison that favours our method, since we are allowed to remesh the map, while [Kovalsky et al. 2015] preserves the original connectivity.

Stress Test. To further evaluate the robustness and applicability of our algorithm, we performed an additional stress test, by parametrizing the 102 models of [Myles et al. 2014] into a planar space filling curve, and adding 3 random positional constraints. These experiments push the algorithm to the limit: MatchMaker fails on 5 if Tutte embedding is used, while it succeeds in all cases, producing bijective maps exactly satisfying the hard positional constraints, with progressive embedding (Figure 12).

6 LIMITATIONS AND CONCLUDING REMARKS.

We introduced a robust algorithm to compute planar embeddings, and demonstrated its practical utility in common geometry processing tasks. Our algorithm is provably correct in infinite precision and is designed to work robustly with floating point coordinates: unfortunately we cannot guarantee that an output is produced in the latter case since a solution of the local point placement problem might not exist. Consider the example in Figure 13: the bounding box of the triangle has short sides (the difference between the floating point coordinate representation is only in the least significant bit of the mantissa). Assume that our algorithm needs to split off a vertex from the vertex with numerically flat angle $A$, placing the resulting point in the interior. In this situation, our algorithm fails, since the average of the coordinates (in floating point) of the boundary triangle does not lie inside the triangle due to numerical rounding.

Except for this extreme case, we have not observed any other failure cases for our algorithm, which produced robustly thousands of embeddings, and, when paired with matchmaker, enables the robust generation of constrained locally injective maps.
REFERENCES


A PROOFS

The proof of the existence of the collapse sequence for two-dimensional manifold meshes can be found, e.g., in [Mijatović 2003], where it is derived from the shellability of two-dimensional manifold meshes homeomorphic to a disk (i.e., the possibility of removing triangles one-by-one, keeping the topology of the remaining part of the mesh unchanged), and make use of a specific composition of Pachner moves equivalent to edge collapse. We present a different proof, based on proving the existence of a collapsible edge, which is aligned with the structure of our algorithm and helps us to show the existence of the inverse vertex split sequence.

A.1 Existence of the Collapse Sequence

We assume that the input mesh connectivity (V0, F0) is manifold, i.e., each edge is shared by no more than two triangles, and the triangles incident at a vertex can be arranged in a sequence so that two sequential triangles share an edge. For interior vertices, the sequence is circular, i.e. the first and last triangles also share an edge. With the topology of a 2D disc, the graphs of edges of such meshes are planar i.e., can be embedded in the plane, with positions \( P_0 = \{ p_i \in \mathbb{R}^2 \} \) assigned to vertices \( v_i \). By the Fáry’s theorem, [Fáry 1948] there is a straight-edge embedding of this graph in the plane with non-intersecting edges (Fáry embedding). In subsequent lemmas, we use geometric images of vertices and edges under this embedding. Only the existence of this embedding, but not the specific construction, is used to prove the existence of the collapse sequence.

The following sequence of lemmas focuses on the one-ring neighborhood of an interior vertex, and shows that at least one of the adjacent edges satisfies the link condition. This observation further leads to Theorem A.8: a valid sequence of edge collapses can be used to reduce the mesh to a mesh with a single interior vertex. A vertex of the mesh is interior if it does not lie on the boundary, and an edge is interior if its two endpoints are interior vertices.

**Definition A.1.** An interior edge \( \overline{v_i v_k} \) satisfies the link condition if \( |N_i \cap N_j| = 2 \), where \( N_j \) is the set of the adjacent vertices of \( v_j \).

**Lemma A.2.** Let \( v_0 \) be an interior vertex of degree \( d \) (Figure 14). We enumerate its neighbors clockwise around the vertex (using Fáry embedding), denoting them \( v_1, v_2, \ldots, v_d \). Assume \( \overline{v_0 v_1} \) violates the link condition, i.e., \( N_0 \cap N_1 \) contains a vertex \( v_k \), \( k = \min (N_0 \cap N_1 \setminus \{ 2, d \}) \). (1) If the triangle \( \Delta_{v_0 v_1 v_k} \) is oriented counterclockwise, then the set \( N_i \), consisting of adjacent vertices of \( v_i \), lies within \( \Delta_{v_0 v_1 v_k} \), for any \( 1 \leq i < k \). (2) If \( \Delta_{v_0 v_1 v_k} \) is oriented clockwise, then \( N_i \) is within \( \Delta_{v_0 v_1 v_k} \) for \( k < i < d \).

Proof. Without the loss of generality, consider \( \Delta_{v_0 v_1 v_k} \) oriented counterclockwise. Consider the segment \( \overline{v_0 v_k} \), for \( 1 < i < k \). The half-line starting at \( v_0 \) and containing this segment is between half-lines containing \( v_1 \) and \( v_k \), because the vertices were numbered counterclockwise. Therefore the half-line contains a point in the interior of \( \Delta_{v_0 v_1 v_k} \), by continuity. If \( v_1 \) is outside or on the boundary of \( \Delta_{v_0 v_1 v_k} \) then the half-line connects an interior and non-interior point different from \( v_0 \), and intersects \( \overline{v_0 v_k} \), which contradicts the assumption on the embedding. Thus, \( v_1 \) is in the interior of \( \Delta_{v_0 v_1 v_k} \).
Similarly, all points in \( N_i \) are either in the interior of \( \Delta v_0 v_i v_k \), or on its vertices, as the edges of the embedding do not intersect except at vertices.

Without loss of generality, we assume that the first case of the definition and take a closer look at \( \Delta v_0 v_i v_k \). Intuitively, one can think of an edge that violates the link condition as having two endpoints which are connected to (at least) three common vertices. Therefore, on one of the sides of the edge, there would be at least two vertices connected to it. The next lemma establishes the fact that if a sequence of edges violates the link condition, then the "lower"(smaller indices) side of the edge always has only one vertex connected to its endpoints.

**Lemma A.3.** Suppose an edge \( e_0 \) violates the link condition, and \( k \) is defined as in Lemma A.2. Suppose, W.L.O.G., \( \Delta v_0 v_1 v_k \) is oriented counterclockwise, and let \( 1 < i < k \). If additionally for all \( 1 < t < i \), \( \overline{v_0 v_i} \) violates the link condition, then \( v_{i-1} \) is the only vertex with index in the range \( 1 \leq t \leq i-1 \) connected to \( v_i \).

**Proof.** We prove the Lemma by induction. The base case, \( i = 2 \) the proposition clearly holds. Suppose for all \( i \geq j \) holds. Since \( \overline{v_0 v_j} \) violates the link condition, \( N_0 \cap N_j \) contains \( v_{j+1} \) and \( v_n \) for some \( j + 1 < n \leq k \) (see Figure 15), by the inductive assumption, \( v_{j+1} \) is in the interior of \( \Delta v_0 v_j v_k \) by Lemma A.2. For \( m \leq j-1 \), \( v_m \) is outside \( \Delta v_0 v_j v_k \), because the half-line \( \overline{v_0 v_j} \) is between \( \overline{v_0 v_m} \) and \( \overline{v_0 v_j} \) by the choice of numbering. It follows that in order to connect \( v_{j+1} \) to a previous vertex \( v_m \), \( \overline{v_{j+1} v_m} \) would have to intersect the boundary of \( \Delta v_0 v_j v_k \), which contradicts fact that we are using an intersection-free Fáry embedding. This proves the induction step.

We conclude that under the assumptions of Lemma A.2, first case, \( v_1, \ldots, v_k \) are interior vertices in the triangle \( \Delta v_0 v_i v_k \), thus interior vertices of the mesh. Then \( \overline{v_0 v_1} \ldots \overline{v_0 v_{k-1}} \) are interior edges. The next lemma shows that at least one of them satisfies the link condition.

**Lemma A.4.** Following the first case in Lemma A.2. If for all \( n < k-1 \), \( \overline{v_0 v_n} \) violates the link condition, the interior edge \( \overline{v_0 v_{k-1}} \) satisfies the link condition.

**Proof.** By definition, \( N_0 \cap N_{k-1} \) is not empty. By Lemma A.3, the only vertex with index less than \( k-1 \) contained in \( N_{k-1} \) is \( v_{k-2} \). On the other hand, the only remaining vertex of \( N_0 \) with index greater than \( k-1 \) inside \( \Delta v_0 v_{k-1} v_k \) is \( v_k \). So we have exactly two vertices in \( N_0 \cap N_{k-1} \), i.e., \( \overline{v_0 v_{k-1}} \) satisfies the link condition.

**Definition A.5.** A fan of triangles \( F(v_0; v_1 \ldots v_{d+1}) \), centered at \( v_0 \), with \( v_i \) enumerated counterclockwise around \( v_0 \), is a sequence of non-repeating triangles \( \{ \Delta v_0 v_i v_{i+1} \} \) if \( i = 1 \ldots d \). A fan is closed, if \( v_{d+1} = v_1 \), otherwise it is open.

**Definition A.6.** Given a triangulation of a polygonal planar domain, with two interior vertices \( v_0, v_m \), whose neighbors are \( N_0 \) and \( N_m \), a collapse operation from \( v_m \) to \( v_0 \) connects all vertices \( v \in N_m \setminus N_0 \) to \( v_0 \), and removes \( v_m \) with incident edges. A collapse operation is valid if \( \overline{v_0 v_m} \) satisfies the link condition, and neither of the end points is a boundary vertex.

To define a collapse operation in a reversible way, in addition to specifying the pair of vertices, we define a fan \( F(v_0; v_1 \ldots v_{d+1}) \) in the mesh obtained after the collapse. The vertices \( v_1 \ldots v_d \) are the vertices that were connected to \( v_m \) before the collapse. In other words, we record a collapse operation, transforming the mesh \( (V_i, F_i) \rightarrow (V_{i+1}, F_{i+1}) \), as the pair \( C_i = (v_m, F(v_0; v_1 \ldots v_d)) \), where \( v_m \) is the removed vertex in \( V_i \), and \( F \) is a fan of triangles in \( F_{i+1} \).

**Lemma A.7.** A 3-connected and planar mesh, is still 3-connected and planar after any interior edge collapse \( C = (v_m, F(v_0; v_1 \ldots v_d)) \).

**Proof.** The link condition ensures that the mesh remains manifold after an edge collapse [Dey et al. 1999]. 3-connectedness of a triangle mesh is equivalent to the requirement that no two boundary vertices are connected by an interior edge. As no collapses involving boundary vertices are allowed, if there are no such edges before the collapse, no such edge may appear after the collapse: the only new edges connect vertices of the fan of \( v_m \) to \( v_0 \), which is interior.

**Theorem A.8.** If the edge graph of a mesh \( (V, F) \) is planar and 3-connected, there is always an edge that can be collapsed to obtain a planar and 3-connected mesh with one less vertex, unless there is only one interior vertex left.

**Proof.** Suppose no edge in \( (V, F) \) can be collapsed. This means that either there are no edges with two interior endpoints, or all such edges violate the link condition. But by Lemma A.4, the second option is not possible. If there are no edges connecting two interior vertices, then all edges incident at interior vertices have the other endpoint on the boundary. Then all edges in the link of an interior vertex have two endpoints on the boundary. By 3-connectedness, these edges should be boundary edges. Therefore, the link of each
interior vertex forms a complete boundary loop. As we assume the mesh to be simply connected, then there is only one boundary loop. So the whole boundary has to coincide with the link of any interior vertex, from which it follows that there is only one.

We remark that when there is only one interior vertex left, if the boundary vertices $v_i$, $i \geq 1$, are assigned positions $p_i$, so that they form a star-shaped simple polygon, there is a position (within the interior of the kernel of the boundary) $p_0$ for the remaining interior vertex $v_0$ that results in a valid straight-edge embedding.

As a result of sequentially collapsing edges, we obtain a sequence $(V_i, F_i, C_i), i = 1, 2, \ldots, k$ with the following properties: $|V_i| = |V_{i-1}| - 1$, $(V_k, F_k)$ is a valid triangulation, boundary vertices are the same for all $V_i$, and $C_i$ is a valid collapse.

**Proposition A.9.** Suppose the vertices $v_j$ of the disk-topology manifold mesh $(V, F)$ are assigned parametric positions $p_j$ in the plane, and the map is bijective on the boundary so that the triangles all have positive orientation. Then the sum of the angles of triangles incident at an interior vertex is $2\pi$.

**Proof.** Assign, e.g., unit length to all edges of the mesh; this associates a a surface $M$ with the mesh, with each combinatorial triangle corresponding to an equalateral triangle. Then the positions $p_j$ define a PL map from $M$ to the plane. By Theorem 1 from [Lipman 2014], this map is globally bijective; the statement of the proposition immediately follows.

### A.2 Vertex split

**Definition A.10.** Let $(V, F)$ be a mesh with a valid straight-edge embedding in the plane given by vertex positions $P$. Consider a closed fan of triangles $F(v_0, v_1 \ldots v_d+1)$ centered at an interior vertex $v_0$, with $v_{d+1} = v_1$, and an open sub-fan $F(v_0; v_1 \ldots v_k)$. Vertex split introduces a new vertex $v_m$ with a position $p_m$, $F(v_0; v_1 \ldots v_k)$, replaces it with a fan $F(v_m; v_1 \ldots v_k)$, and adds triangles $\Delta v_0 v_1 v_m$ and $\Delta v_0 v_k v_m$. We denote such a split by $(v_0, v_m; p_m, F(v_0, v_1 \ldots v_k))$.

A split $S = (v_m, p_m, F(v_0; v_1 \ldots v_k))$, in terms of connectivity modification, is the inverse of a collapse $C = (v_m, F(v_0, v_1 \ldots v_k))$: the connectivity of the mesh obtained by applying the split is identical to the mesh that the collapse was applied to.

The following lemma establishes that we can perform a split reversing any collapse while maintaining the validity of the embedding, if the initial embedding is valid.

**Lemma A.11.** Consider a fan of triangles $F(v_0; v_1 \ldots v_k)$ (Figure 5), with positive signed areas $\{A(\Delta v_i v_j v_{i+1})\}_{1 \leq i \leq k}$ and with angles of triangles incident at $v_0$ summing up to $2\pi$. The kernel of the fan has a non-empty interior. Then a new vertex position $p_m$ corresponding to a new vertex $v_m$ located in the interior of the fan, can be split off $p_0$, so that $\min_{1 \leq i \leq k} A(\Delta v_m v_i v_{i+1}) > 0$, $A(\Delta v_0 v_1 v_m) > 0$, $A(\Delta v_0 v_m v_k) > 0$, and angles of triangles in both resulting fans at $v_0$ and $v_1$ sum up to $2\pi$.

**Proof.** Define a function $f(x) = \min_{i} A(\Delta v_i v_{i+1} v_{i+2})$. This is a continuous function of the coordinates $x$ of the point $v$. Because $f(p_0) > 0$, there is a disk $B(p_0, \varepsilon)$, of radius $\varepsilon > 0$, such that $f(x) > 0$ for any $x \in B(p_0, \varepsilon)$, i.e. for all $i$, $A(\Delta v_i v_{i+1} v_{i+2}) > 0$, if we pick $p_m$ inside $B(p_0, \varepsilon)$. Suppose we initially place $p_m$ at $p_0$, with new triangles added as a result of the split having zero angles at $v_1$ and $v_k$. We note that the for each of $v_0$ and $v_1$ in this degenerate configuration the angles of incident triangles sum up to $2\pi$. The angles of triangles also change continuously as functions of vertex position $p_x$, so does their sum. On the other hand, if each triangle remains positively oriented ($A(\Delta v_i v_{i-1} v_{i+1}) > 0$), then the sum of the angles can only change discretely, and has to be of the form $2\pi n$, $n \in \mathbb{Z}$ ($n$-fold cover). We conclude that $n$ has to remain one, as it is one for the initial position.

Consider the intersection $C$ of the half-planes bounded by lines containing $p_0 p_1$ and $p_0 p_2$ (for each segment, we choose the half-line on the side of the interior of the fan). If $p_m \in C \cap B(p_0, \varepsilon)$, then $A(\Delta v_0 v_1 v_m) > 0$, $A(\Delta v_0 v_m v_k) > 0$ also hold.

The following theorem is a straightforward application of Lemma A.11.

**Theorem A.12.** Suppose we have a sequence of valid collapses $(V_i, F_i, C_i), i = 0 \ldots N - 2$, where $N = |V_0|$, the number of interior vertices in the initial mesh, all $(V_i, F_i) i = 0 \ldots N - 1$ are 3-connected planar, and the last mesh $(V_{N-1}, F_{N-1})$ with a single interior vertex has a valid straight-edge embedding in the plane with vertex positions $P_{N-1}$. Suppose $C_i = (v^{(i)}_0, \cdots, v^{(i)}_{k_i})$.

Then the sequence of vertex splits $S_i$ that are inverses of $C_i$, results in a valid straight-edge embedding of $(V_0, F_0)$, given by vertex positions $P_0$.

**Proof.** $V_{N-1}, F_{N-1}$ with positions $P_{N-1}$ is valid by assumption. Each step of vertex split with $S_i$ results in a straight-edge embedding of $(V_i, F_i)$ by Lemma A.11. By induction, the embedding of $(V_0, F_0)$ obtained by the sequence of splits reversing the sequence of collapses is a straight-edge embedding.

### B MATCHMAKER

![Fig. 16. Illustration of the MatchMaker algorithm. The regions surrounded by solid black lines with the same numbering have one to one correspondence throughout the 3 subfigures (a), (b), and (c). (a) unconstrained parametrization embedded in a triangulation of the bounding box. The dashed lines are the triangulation of the regions between the mesh and virtual boundary. (b) triangulated virtual boundary with three constrained parametric positions (edges are splitted accordingly). (c) result of Tutte embedding for every individual patch glued together.]

MatchMaker [Kraevoy et al. 2003] tackles the problem of planar parametrization of a surface with hard positional constraints. The general idea is to decompose the mesh into patches and map them to convex domains using Tutte embedding to generate a bijective
Fig. 17. The scenario that the red tracing path is invalid. (a) The extended unconstrained parametrization $M'_0$, a new tracing path is generated (red line) to match the purple edge in (b). (b) Triangulated virtual boundary $M_1$. The internal edge (purple line) separate the mesh into two sub-regions, while the two hard constraints land in the same sub-region. It differs from the configuration in (a), where the constraint points are in different sub-regions. (c) The purple tracing path is valid since the hard constraints lies in the same sub-region, while red tracing path blocks future necessary tracing paths, such as the green one.

parametrization. We briefly summarize the pipeline of the original MatchMaker method.

**Triangulate Virtual Boundary.** Use a conventional unconstrained parametrization method to get a planar mesh $M_0$ (the dark region in Figure 16(b)). Embed $M_0$ in a rectangular bounding box, and use the constrained Delaunay triangulation to triangulate the region between the planar mesh and the virtual boundary, to obtain $M'_0$ (Figure 16(a)). Similarly, create a constrained Delaunay triangulation of the same bounding box with hard constraints at prescribed position using [Shewchuk 1996], to obtain the mesh in Figure 16 (a). We call this mesh $M_1$.

**Matching Patches.** For each internal edge of $M_1$, trace an edge path between the two corresponding points on $M'_0$ using Dijkstra’s shortest path algorithm. This process is performed sequentially, one edge each time. Every new paths traced must meet the following criteria: (1) it does not intersect with any other paths; (2) it does not block necessary future paths. This is achieved using the minimum spanning tree as described in Section 4. Steiner points may be needed to make sure a valid path can always be found (red dot in Figure 17 (c)). As the result of this step, extended planar mesh $M'_0$ is subdivided into patches, and every patch is matched with a triangle in $M_1$.

**Embedding.** At the previous step, the problem is reduced to mapping a patch of a planar mesh to a convex boundary, i.e., a problem solved by Tutte embedding [Tutte 1963]. Additional edge splitting operation is needed to match the boundary vertices number of triangles in $M_0$ to vertices number on corresponding paths in $M'_1$, as shown in Figure 16 (b).

**Smoothing and Post-processing.** The parametrization generated at the embedding stage can be optimized using any technique preventing triangles from changing their orientations.