Efficient Embeddings in Exact Arithmetic

UGO FINNENDAHL, TU Berlin, Germany
DIMITRIOS BOGIOKAS, TU Berlin, Germany
PABLO ROBLES CERVANTES, TU Berlin, Germany
MARC ALEXA, TU Berlin, Germany

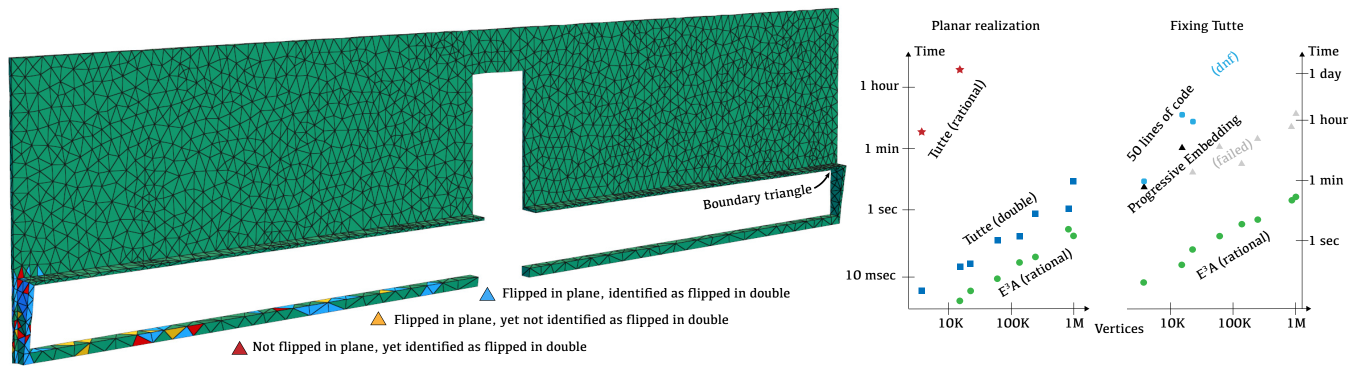


Fig. 1. The mesh above has only 3785 vertices and well-shaped triangles. Realizing it in the plane using Tutte's embedding algorithm in double precision creates 121 flipped triangles (blue and yellow). These flipped triangles cannot be reliably detected in double precision: yellow triangles are flipped but have positive area in computation based on doubles; red triangles have negative area in double, but exact computation reveals that they are correctly oriented. We create several subdivisions of the mesh to analyze how different strategies for fixing the problems scale. Solving the linear system for the Tutte embedding in rationals fails for all but very small meshes. Our efficient embedding in exact arithmetic (E^{3}A) based on Schnyder realizers [Schnyder 1989] is faster than Tutte embeddings in double precision while guaranteed. We also develop an algorithm for fixing planar triangulations. It is three orders of magnitude faster than Progressive Embedding [Shen et al. 2019] and a recent robust untangling approach [Garanzha et al. 2021].

We provide a set of tools for generating planar embeddings of triangulated topological spheres. The algorithms make use of Schnyder labelings and realizers. A new representation of the realizer based on dual trees leads to a simple linear time algorithm mapping from weights per triangle to barycentric coordinates and, more importantly, also in the reverse direction. The algorithms can be implemented so that all coefficients involved are 1 or −1. This enables integer computation, making all computations exact. Being a Schnyder realizer, mapping from positive triangle weights guarantees that the barycentric coordinates form an embedding. The reverse direction enables an algorithm for fixing flipped triangles in planar realizations, by mapping from coordinates to weights and adjusting the weights (without forcing them to be positive). In a range of experiments, we demonstrate that all algorithms are orders of magnitude faster than existing robust approaches.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; **Mesh models**; **Mesh geometry models**.

Authors' addresses: Ugo Finnendahl, TU Berlin, Berlin, BER, Germany, finnendahl@tu-berlin.de; Dimitrios Bogiokas, TU Berlin, Berlin, BER, Germany, d.bogiokas@tu-berlin.de; Pablo Robles Cervantes, TU Berlin, Berlin, BER, Germany, pablo.roblescervantes@campus.tu-berlin.de; Marc Alexa, TU Berlin, Berlin, BER, Germany, marc.alexa@tu-berlin.de.

0730-0301/2023/8-ART1 $15.00
https://doi.org/10.1145/3592445

Additional Key Words and Phrases: parametrization, Schnyder labeling, integer coordinates

ACM Reference Format:
Ugo Finnendahl, Dimitrios Bogiokas, Pablo Robles Cervantes, and Marc Alexa. 2023. Efficient Embeddings in Exact Arithmetic. *ACM Trans. Graph.* 42, 4, Article 1 (August 2023), 17 pages. https://doi.org/10.1145/3592445

1 INTRODUCTION

Generating a planar *parameterization* of a triangulation in 3D is an essential step in computer graphics and geometry processing pipelines. In many scenarios, guaranteeing that the triangulation is *embedded* is crucial. It implies *local injectivity*, i.e., all triangles are non-degenerate and consistently oriented across edges, as well as global bijectivity, i.e., each point in the domain is uniquely assigned to exactly one vertex, one edge, or one triangle.

The importance of these properties is reflected by a large number of works aiming at (locally) injective mappings. They are most commonly based on minimizing a function in the position of the vertices. The energies, in most cases, are designed so that their local or global minimizers yield the desired injectivity. An early example for this type of approach is this: fix the boundary in the planar realization in strictly convex position and consider interior edges as springs, meaning we want to minimize the sum of their squared lengths. This is a quadratic function in the vertex positions of the realization. Tutte, famously, showed that the minimizer defined as the solution of a linear system exists, is unique, and provides an embedding [Tutte 1963]. This general approach has been extended and modified in many ways, see the surveys by Floater and Hormann [2005] and Sheffer et al. [2007]. More recent work has attempted to better model the similarity between the original triangulation in 3D and the planar realization, while still being efficient [Lipman 2012; Sawhney and Crane 2017]. For the resulting more complicated non-linear functions to be optimized, smarter techniques for the minimization have been introduced, with a particular focus on injectivty [Du et al. 2020, 2022; Garanzha et al. 2021], and some providing guarantees [Rabinovich et al. 2017; Su et al. 2020].

The theoretical foundations for injectivity or bijectivity in the above stream of papers are mostly based on real numbers, yet practical implementations of the algorithms use floating point representations. This seriously limits claims about robustness and guarantees, as already the computation of the signed area of a triangle in floating point may not reflect the true orientation of the triangle. This is a well known and an important topic in computational geometry [Hoffmann 1989], because many elementary algorithms easily break when implemented in floating point arithmetic. There are different strategies to handle this problem [Schirra 2000]. Using exact arithmetic [Granlund et al. 2020] would allow implementing algorithms as developed for real numbers [Yap 1997], but is often found to be too slow in practice. A widespread solution is the dynamic increase in floating point resolution when necessary to guarantee an exact result [Shewchuk 1997].

Reading the literature on injective parameterizations, one may think that the problems of finite precision are somehow irrelevant in practice for mesh parameterization. However, as nicely demonstrated by Shen et al. [2019] this is not true: taking all topological spheres in the Thingi10K data [Zhou and Jacobson 2016], removing a single triangle, and generating a parameterization using Tutte embeddings by solving a linear system in IEEE754 double format, 80 out of 2718 planar triangulations are not embedded. We repeated and extended this experiment. The smallest example (without degenerate triangles) we found has only 193 vertices. To understand the range of problems even for small and inconspicuous inputs, consider the mesh in Figure 1: it has 3785 vertices and all 7566 triangles are well-shaped (i.e., have no unnecessarily extreme aspect ratios). We remove the first triangle in the face list and realize the mesh as explained above. The number of flipped triangles computed in exact arithmetic is 121. More severely, when checking for flipped triangles in double precision, we find that 17 flipped triangles are reported as correctly oriented; another 14 correctly oriented triangles are reported to be flipped. This highlights that not only the numerical optimization may fail to provide a result that is consistent with theoretical results, also checking the validity in floating point may be misleading. Perhaps even more important in practice, downstream algorithms relying on the correct orientation of triangles may be more 'sensitive' to near degeneracies than the rather simple computation of signed area. We demonstrate this problem by running the minimization of symmetric Dirichlet energy [Smith and Schaefer 2015] on Tutte embeddings computed in double precision and see the optimization failing on realizations that are embedded (Section 7). We also show that solving the linear system in exact rational representation is infeasible even for small meshes (Section 7).

We suggest a new approach to guaranteeing an embedding based on *Schnyder labelings* [Schnyder 1989]. They have been introduced to generate drawings of planar graphs on small integer grids with a boundary consisting of a single triangle, i.e. three vertices. Put simply, barycentric coordinates for the vertices are generated by 'counting' triangles divided by different trees, the so-called *Schnyder* woods [Schnyder 1990] (we explain the trees in Section 2). The proof that this generates an embedding only relies on the fact that each triangle contributes positively, suggesting that each triangle could be equipped with an arbitrary positive weight [Dhandapani 2010]. The properties of the algorithms immediately suggest that the mapping from weights to coordinates is linear and can be computed in linear time. This also means, while every realization of a triangulation with positive weights is embedded, not all embedded triangulations are represented by positive weights.

In this work, we provide a new view on Schnyder realizers based on a dual tree (see Section 4). This view has the following advantages:

- We feel our approach is simpler than earlier algorithms, better exhibiting the properties of the mapping and easier to implement efficiently. In particular, it directly shows that the coefficients in the linear map are only in $\{-1, 0, 1\}$.
- It directly reveals that also the 'reverse' mapping from Euclidean coordinates to weights can be computed in linear time (to our knowledge, this direction was so far computed by solving a linear system [Barrera-Cruz et al. 2014]). Even better yet, each weight only depends on a constant number of Euclidean coordinates.

These mappings immediately lead to a trivial algorithm for fixing a realization: map from Euclidean coordinates to weights, increase the weights, and check the realization in Euclidean space. This can be done in integer coordinates, and the algorithms terminates with an embedding because any set of positive weights leads to an embedding. However, we show that the properties of Schnyder realizers seen through the dual tree admit a faster alternative with less distortion, since they provide guarantees even for negative weights.

The significant advantage of the algorithms we provide is that the number of bits we use in the representation of the geometry affects only the *quality* of the parameterization, but not whether or not any triangles are flipped. We offer several practical comparisons in Section 7. For testing the initial embedding compared to Tutte we optimize symmetric Dirichlet energy and find that embeddings generated with Schnyder realizers are faster and, on average, provide a better starting point. Comparing algorithms for fixing planar realizations, we find that our algorithm is orders of magnitude faster.

A fundamental drawback of the method is the assumption that the input has disk-topology and only three vertices on the boundary. This limits practical utility. We make some remarks in Section 6 on how to handle non triangular boundaries and discuss other limitations and possible extension in Section 8.

2 BACKGROUND

As our results require an understanding of Schnyder labelings, Schnyder realizations, and corresponding primal trees representing the subdivision of the triangulations, we provide a brief introduction here. For more (technical) details, including proofs, we refer readers to the original publication of Schnyder [1990]. The topic of Schnyder labelings has been studied extensively from a combinatorial perspective, see for example Felsner [2001] or Brehm [2000].

We consider an orientable closed triangulated genus 0 surface with unoriented graph $G = (V, E)$, vertex set V and edge set $E \subseteq \binom{V}{2}$. The graph is *maximal planar* as adding any edge to the edge set would make the graph non-planar. We fix a triangular face t_{out} to be the outer face. The set of all inner triangles in the triangulation is denoted by $\mathcal{T} \subseteq \binom{V}{3}$. Such a graph is topologically the same as a triangle mesh with disk-topology and three edges on the boundary.

An *orientation* of any triangle $t \in \mathcal{T} \cup \{t_{\text{out}}\}$ of G is a cyclic ordering of its vertices. Note that there are always two possible orientations of any t. Two neighboring triangles t, t' are said to have *compatible* orientations, if their common edge $\{i, j\}$ appears as ij in the orientation of t and as ji in the orientation of t'. A *global* orientation of G is a choice of orientation for every $t \in \mathcal{T} \cup \{t_{\text{out}}\}$ such that every two neighbouring triangles have compatible orientations. Since G is maximal planar, there always exist two global orientations of G

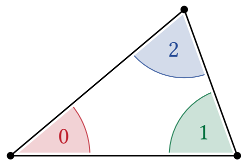
An *orientation* around a vertex $v \in V$ of G is a cyclic ordering of its neighbours, such that $\{u, v, u'\} \in \mathcal{T} \cup \{t_{\text{out}}\}$ for every $u, u' \in V$ that appear as uu' in the cyclic ordering. Notice that, again, around every vertex v there exist two possible orientations. A global orientation on G induces an orientation around each vertex. Namely, the one that $u'vu$ is the orientation of $\{u, v, u'\} \in \mathcal{T} \cup \{t_{\text{out}}\}$ for every $u, u' \in V$ that appear as uu' in the orientation around v.

2.1 Schnyder Labeling

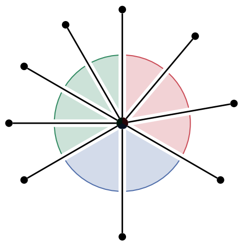
Let G be a maximal planar graph. We fix a global orientation of G and we call it the *positive* orientation for every $t \in \mathcal{T} \cup \{t_{\text{out}}\}$ and for every $v \in V$. The only other orientation, if ever needed is referred to as *negative* orientation. Note that a global orientation can be computed in linear time. Whenever G is embedded in the plane, the positive orientation of each $t \in \mathcal{T}$ is the counter clockwise (CCW) orientation, the positive orientation of t_{out} is the clockwise (CW) orientation and the orientation around every $v \in V$ is again the CCW orientation. A *Schnyder labeling* of G labels each corner of every interior triangle with one of the three labels 0, 1 and 2 such that

(1) in a single triangle $t \in \mathcal{T}$, each label appears exactly once. Moreover the three labels always appear in the same order in the positive cyclic orientation of t (see Fig. 2a), and
(2) around a vertex $v \in V$, each label appears in a single non empty interval. Moreover the three labels always appear in this order in the positive cyclic orientation around v (see Fig. 2b).

In this work the order is always 0, 1, 2. Moreover, for any label $c \in \{0, 1, 2\}$ we name the remaining two labels $c_1, c_2 \neq c$ the *complementary* labels of c. Schnyder [1990] showed that each maximal planar graph has at least one Schnyder labeling that can be constructed in linear time. As the labels are usually colored in red, green and blue, we will use the word color and label interchangeably.



(a) In each triangle the corners are colored in the CCW order 0,1,2.



(b) For each vertex the colored corners appear in the CCW order 0,1,2.

Fig. 2. The two properties of a Schnyder labeling.

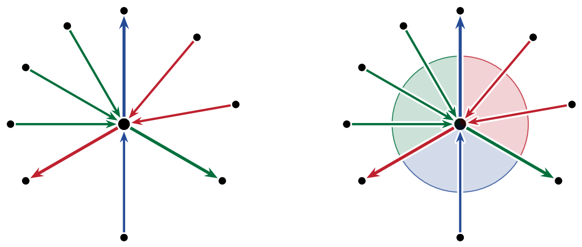
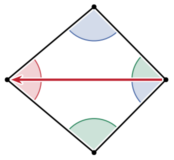


Fig. 3. Visualization of the three properties of a Schnyder realization (left) within a Schnyder labeling (right).

2.2 Schnyder Realization

Note that, due to labeling rules, there cannot be two neighboring interior triangles, whose common edge is opposite of same color corners inside these two triangles.



This means that all three labels 0, 1, 2 must appear in the four adjacent corners of every interior edge. In particular, there always must be one end of the edge such that both adjacent corners have the same label c. If we orient each edge toward this end and label the edge itself with c, we arrive at a *Schnyder realization.* In general, a Schnyder realization of a maximal planar graph G is a choice of orientation and 3-labeling for every interior edge of G, defined by:

(1) Each interior vertex has exactly three outgoing edges, one for each label 0, 1 and 2.
(2) The labels of the outgoing edges of every interior vertex always appear in the order 0, 1, 2 in the positive cyclic orientation around the vertex.
(3) For every interior vertex, all incoming edges, which appear in the positive cyclic orientation around the vertex, between two outgoing edges are labelled by the unique color not appearing in these two outgoing edges.

By a counting argument, it can be deduced that in every Schnyder realization, the three outer vertices have only incoming edges. Thus, the corresponding Schnyder labeling is monochromatic around each one of them. Moreover, using the three inner triangles incident to the three outer edges, it is shown that each one of them has a distinct color and that the three colors appear in the negative

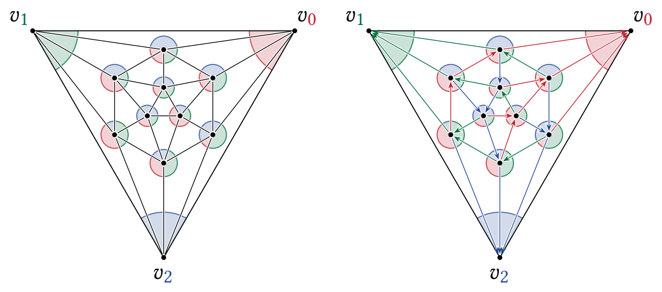
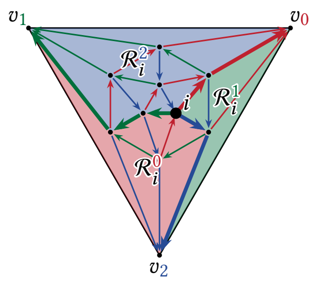
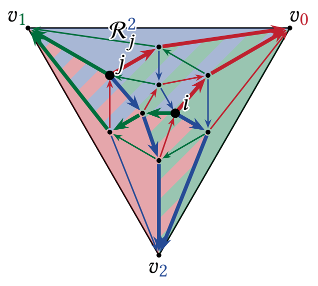


Fig. 4. A Schnyder labeling left and the induced Schnyder realization right.



(a) The paths from i to v_0, i to v_1 and i to v_2 divide the triangulation into the three regions.



(b) A vertex j is within the 2 region of vertex i. Therefore the 2 region of j is a subset of the 2 region of i.

Fig. 5. Every vertex i defines three regions $\mathcal{R}_i^0$, $\mathcal{R}_i^1$ and $\mathcal{R}_i^2$.

orientation of the outer face (which is CCW, if G is embedded). We can therefore assign each boundary vertex a color and thus naming them $v_0, v_1, v_2 \in V$.

The correspondence between the Schnyder labeling and the Schnyder realization described above has the following properties:

(1) Each outgoing edge divides two different colored corner regions that have a different color than this edge.
(2) Each incoming edge always enters in the corner region that has the same color as the edge.

These properties can be seen in Fig. 3. This means that Schnyder labelings and Schnyder realization have a one-to-one correspondence and thus, every maximal planar graph also has at least one Schnyder realization. An example is visible in Fig. 4.

2.3 Primal Trees and regions

Let $c \in \{0, 1, 2\}$ be any color and c_1, c_2 be the complementary colors of c. Schnyder [Schnyder 1990] showed that the directed subgraph induced by the c-labeled edges is a tree. Then, adding the oriented edges $v_{c_1}v_c$ and $v_{c_2}v_c$ to this tree creates a directed spanning tree. We call these three spanning trees T_0, T_1 and T_2. They have the following properties:

(1) The only sink of T_c is v_c, since every interior vertex has a c-labeled outgoing edge and, by construction, v_{c_1}, v_{c_2} have one as well.
(2) There is a unique directed path inside T_c from any vertex to v_c, since T_c is connected, has no cycles and no other sinks.
(3) Two paths of different color starting in an interior vertex do not meet in another vertex. This follows directly from the fixed cyclic ordering of the colored edges.

Since T_c is a tree, between any two vertices $i, j \in V$ there is at most one oriented path from i to j inside T_c. If it exists, we denote this path by $i \rightarrow_c j$.

Given a vertex $i \in V$ the three colored paths to the boundary vertices partition the interior triangles into three disjoint regions $\mathcal{R}_i^c \subseteq \mathcal{T}$, which are according to the color c of the opposite boundary vertex – the single boundary vertex that is not in the region. This is visualized in Fig. 5a.

2.4 Barycentric coordinates and Weights

Schnyder [1990] assigned barycentric coordinates or b_i^0, b_i^1, b_i^2 to each vertex i, by counting the triangles in each colored region divided by the number of all triangles: $b_i^c = \frac{1}{2n-5}|\mathcal{R}_i^c|$ with $c \in \{0, 1, 2\}$. He then showed that these coordinates form barycentric representations, which are a subset of all embeddings. For his proof it is necessary that if a vertex i is in a c colored region of another vertex j, formally if there exists some $t \in \mathcal{T}$, such that $i \in t \in \mathcal{R}_j^c$, then $\mathcal{R}_i^c \subseteq \mathcal{R}_j^c$ and therefore $b_i^c \leq b_j^c$.

This approach was extended [Dhandapani 2010] by weighting each triangle $t \in \mathcal{T}$ with a weight $w_t \in \mathbb{Z}$ such that the sum of weights is unequal to zero.. So the barycentric coordinates (b_i^0, b_i^1, b_i^2) of a vertex i are defined by $b_i^c = \frac{1}{\sum_{t\in\mathcal{T}} w_t} \sum_{t\in\mathcal{R}_v^c} w_t$ with $c \in \{0, 1, 2\}$. As long as all weights are greater than zero, this still produces a barycentric representation as the property $\mathcal{R}_i^c \subseteq \mathcal{R}_j^c \Rightarrow b_i^c \leq b_j^c$ holds. Notice that any scaling of the weights produces the same barycentric coordinates.

In this work we choose to not divide by the normalization factor $\sum_{t\in\mathcal{T}} w_t$ and record it as an additional coordinate, instead. This way the barycentric coordinates are able to stay in $\mathbb{Z}$ and the weights are no longer scaling invariant. More formally, we define the following two spaces:

- The *weight* space $\mathcal{W} = \mathbb{Z}^{|\mathcal{T}|}$. Each element $\mathbf{w} \in \mathcal{W}$ is an assignment of weights to the inner triangles $t \in \mathcal{T}$, where $|\mathcal{T}| = 2|V| - 5$ according to Euler's formula. We represent $\mathbf{w}$ by a $|\mathcal{T}|$-vector of integers $(w_t)_{t\in\mathcal{T}}$.
- The *barycentric* space $\mathcal{B} = \mathbb{Z}^{2(|V|-3)+1}$. Each element $\mathbf{b} \in \mathcal{B}$ records two of the three unnormalized barycentric coordinates of all inner vertices $i \in V \setminus \{v_0, v_1, v_2\}$ and also records the sum N of all three unnormalized barycentric coordinates of some vertex once, as it is the same for every vertex. This space has dimension exactly $2(|V| - 3) + 1 = 2|V| - 5$. To ease the notation, we represent $\mathbf{b}$ by a $|V|$-vector of 3-vectors $(\mathbf{b}_i)_{i\in V}$, where $\mathbf{b}_i = (b_i^0, b_i^1, b_i^2)^T$, such that $b_i^0 + b_i^1 + b_i^2 = N$ for every $i \in V$. In this notation, we also write $\mathbf{b}_{v_0} = (N, 0, 0)^T, \mathbf{b}_{v_1} = (0, N, 0)^T, \mathbf{b}_{v_2} = (0, 0, N)^T$, although they are not really recorded in $\mathcal{B}$.

In Section 4 we see that the mapping between weight space and barycentric space is bijective. One of our contributions is that we show that the mapping can be calculated in linear time for both directions. It is interesting to note that the mapping from weights to a single vertex coordinate also needs linear time. This is opposite to the mapping from barycentric space to a single weight, which can be calculated in constant time.

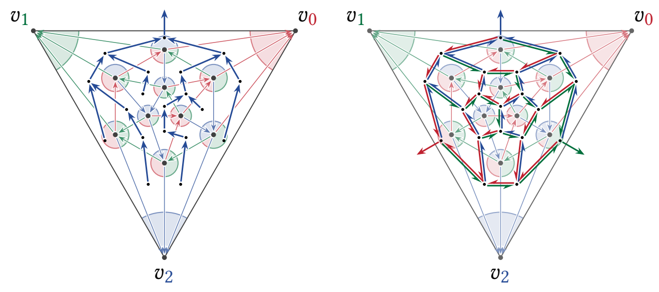
3 RELATED WORK

Graph drawing methods in integers. Every planar graph has a planar straight-line drawing by the theorems of István [1948]; Stein [1951]; Wagner [1936]. Their results did not bound the grid-size by any polynomial in the number of vertices n. De Fraysseix et al. [1990] first introduced an asymptotically optimal straight-line embedding on a $(2n-4)\times(n-2)$ grid in $O(n\log n)$ time. Chrobak and Payne [1995] reduced the time complexity to $O(n)$. Using Schnyder labelings and the induced trees [Schnyder 1989], Schnyder [1990] built upon his findings and presented two methods for drawing planar graphs on a $(2n-5)\times(2n-5)$ grid and a more compact version using a $(n-2)\times(n-2)$ grid, both in linear time. All these methods assume a triangulated planar graph with a three edge boundary. Therefore all methods need to add edges until the graph is triangulated and remove them after the embedding. Tutte [1960, 1963] avoids this by introducing convex drawings, meaning that every face will be embedded as a convex polygon, and proved that every 3-vertex-connected planar graph has a convex drawing. The embedding can be found by solving a (sparse) linear system of equations. The methods of both De Fraysseix et al. [1990] as well as Schnyder [1990] were generalized to the same setting, namely convex embeddings, retaining the linear time complexity ([Chrobak and Kant 1997; Kant 1996] and [Battista et al. 1999; Bonichon et al. 2007; Felsner 2001; Schnyder and Trotter 1992]).

Our method is based on the non-compact method of Schnyder.

Applications of Schnyder woods. Besides integer grid embeddings, Schnyder woods were used in different applications. Aleardi et al. [2009] extend Schnyder's definitions and algorithms to closed orientable surfaces of arbitrary genus to encode meshes of arbitrary genus. The algorithms work in linear time (for fixed genus). Felsner and Zickfeld [2008] draws a connection to orthogonal surfaces by extending schnyder labelings to non triangular faces. Recently, Barrera-Cruz et al. [2019] presents an improved method for morphing between two planar drawings of the same triangulation while maintaining straight-line planarity using Schnyder embeddings. He and Zhang [2010] developed a greedy routing algorithm, a message is forwarded to a neighbor that is closer to the destination, that use Schnyder woods. Their method uses fewer bits to represent the local coordinates compared to the classic approach. Another application are cycle separators, which separate a graph into two vertex sets with similar number of vertices. Castelli Aleardi [2019] use a heuristic to balance Schnyder woods to improve the shape of the regions.

Exact arithmetic and parametrizations. In the realm of generating meshes (e.g., for simulation) it is standard practice to use robust arithmetic [Brönnimann et al. 2023; Shewchuk 1996; Si 2015]. This can be done using exact computation [Yap 1997], but it is more common to use multi-precision types and increase resolution as needed to guarantee reliable predicates [Shewchuk 1997]



(a) A Schnyder labeling and the corresponding dual tree $\mathcal{S}^2$.

(b) A Schnyder labeling and all three dual trees.

Fig. 6. A Schnyder labeling induces three dual trees that each spans over all triangles.

Interestingly, while parameterization methods for triangle meshes are very similar in nature, it is uncommon to use such methods for checking the orientation of triangles. Most methods that make 'guarantees' are using floating point types and carefully select thresholds for the signed area of a triangle so that despite inexact computation the status of a correctly oriented triangle is reliable [Garanzha et al. 2021]. On the other hand, this decreases the available resolution of the floating point type – we suspect that the problem cases we found for Progressive Embedding [Shen et al. 2019] are resulting from this limitation.

One of the (likely) very few parameterization methods that avoids such problems by using exact arithmetic is for the 3D case, i.e., mapping a tetrahedral mesh into a convex domain [Campen et al. 2016]. Interestingly, their representation is also essentially barycentric and they map any non-rational function to a rational one to avoid dealing with numerical issues.

4 CONSTRUCTION

We present two novel linear time algorithms: The first one (Algorithm 1) computes barycentric coordinates $\mathbf{b}\in\mathcal{B}$ from weight vectors $\mathbf{w}\in\mathcal{W}$; the second (Algorithm 2) computes the inverse, i.e. $\mathbf{w}\in\mathcal{W}$ from $\mathbf{b}\in\mathcal{B}$. For this, we extend the theory discussed in Section 2, introducing new notions, namely dual binary trees, dual regions, and dual edge labeling. We feel that the dual picture simplifies the algorithms, both conceptually and implementing them in practice.

4.1 Dual binary trees and dual edge labeling

We want to contribute a dual labeling that arises from a Schnyder realization and the dual graph. A Schnyder realization induces three different colored dual binary trees each one containing all triangles. Given a triangle, these trees can be used to divide all vertices into different regions.

Let $c\in\{0,1,2\}$ be any color and c_1,c_2 the complementary colors of c. There exists a dual spanning tree of T_c in the dual graph G^*, which consists of all edges in G^* that correspond to the edges of

$G \setminus T_c$ [Aigner and Ziegler 2009], i.e. all edges with color c_1 or c_2. We call these dual spanning trees $\mathcal{S}^0, \mathcal{S}^1, \mathcal{S}^2$. We orient each $\mathcal{S}^c$ by first making the outer triangle the root of the tree and then orienting each dual edge towards the root. We visualized an example of $\mathcal{S}^0, \mathcal{S}^1$ and $\mathcal{S}^2$ in Fig. 6. This tree has the following properties.

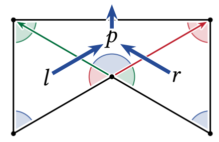
(1) The root only has one child, namely the only inner triangle incident to the boundary edge $\{v_{c_1}, v_{c_2}\}$.
(2) It is a rooted *binary* tree, as a triangle has at most three neighbouring triangles from which one is the parent, leaving a maximum of two children.

Next, we will prove that there are three equivalent descriptions of these dual trees.

Lemma 4.1. *Given a maximal planar graph G, a Schnyder labeling and its corresponding Schnyder realization, the following are equivalent:*

i *For every $c \in \{0, 1, 2\}$, $\mathcal{S}^c$ is the dual tree defined above.*
ii *For every $c \in \{0, 1, 2\}$, $\mathcal{S}^c$ is a directed graph in G^*, containing the dual edges of all edges in G opposite of a c labeled corner, oriented outwards from the triangle that has this c labeled corner.*
iii *For $\{c, c_1, c_2\} = \{0, 1, 2\}$, where c, c_1, c_2 are in the same cyclic order as 0, 1, 2, $\mathcal{S}^c$ is a directed graph in G^*, containing the dual edges of all c_1 and all c_2 edges of G, oriented as follows:*
- *The c labeled dual edge of a c_1 edge e is oriented from the triangle that e is directed opposite to the positive cyclic orientation (its right triangle) to the triangle that e is directed according to the positive cyclic orientation (its left triangle).*
- *The c labeled dual edge of a c_2 edge is oriented from its left triangle to its right triangle.*

Proof. Let $\mathcal{S}^c$ be the dual tree defined above. For the only child of the root both ii and iii are trivially true. We then prove these properties inductively also for every left and every right child. Without loss of generality we show the inductive step in the case of $\mathcal{S}^2$.



Let $p \in \mathcal{T}$ be an inner triangle satisfying the inductive hypothesis and let $l, r \in \mathcal{T}$ its left and right child respectively. Then, because of the fixed cyclic ordering of the Schnyder labeling, the edge in G corresponding to the dual edge connecting p and l is incident to the 1 and 2 corners inside p. Since it corresponds to a dual edge inside $\mathcal{S}^2$ it cannot be labeled by 2, so it is labeled by 1 and is oriented like iii dictates. Similarly, the edge corresponding to the dual edge connecting p and r is labeled by 0 and is also oriented as in iii. Moreover, there is exactly one admissible Schnyder labeling of l and r and in this labeling the corners in l and r opposite from p are indeed both colored by 2 as in ii.

For the inverse statement ii⇒i, notice that the dual spanning tree visits each triangle once and each triangle has exactly one corner labeled c, so the description in ii determines fully $\mathcal{S}^c$. Similarly, for iii⇒i, the edges of the c dual spanning tree are exactly the dual c_1 and c_2 edges, so iii also fully determines $\mathcal{S}^c$. □

This description means that every triangle has three different colored outgoing dual edges, appearing in the same positive cyclic order as the corner labels. It must therefore have three, not necessarily different colored, incoming edges, we can verify this in

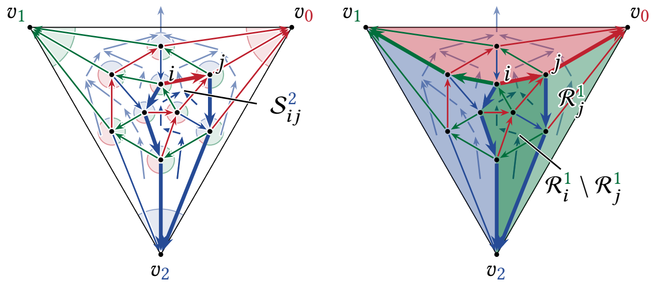


Fig. 7. The dual sub tree $\mathcal{S}^2_{ij}$ is enclosed by ij, $i {\longrightarrow}_2 v_2$ and $j {\longrightarrow}_2 v_2$ (left). As $\mathcal{S}^2_{ij}$ labeled 0, it contains all triangles of the region $\mathcal{R}^1_i \setminus \mathcal{R}^1_j$ (right)

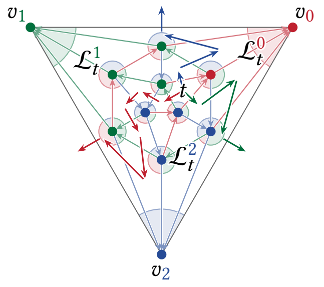


Fig. 8. The three dual paths from t to the outer face divide the vertices in three regions $\mathcal{L}^0$, $\mathcal{L}^1$ and $\mathcal{L}^2$.

Section 4.1. Now, we can see that the trees $\mathcal{S}^c$ have the following dual properties of T_c:

(1) The only sink of $\mathcal{S}^c$ is the outer face, since every interior triangle has a c-labeled incoming dual edge.
(2) There is a unique directed dual path inside $\mathcal{S}^c$ from any triangle to the outer face, since $\mathcal{S}^c$ is connected, has no cycles and no other sinks.
(3) Two dual paths in $\mathcal{S}^c$ of different color, starting in an interior triangle, do not meet in another triangle except the outer face. This follows from the fixed cycling ordering of the colored dual edges.

Intuitively speaking, every sub tree in $\mathcal{S}^c$ grows until hitting a c colored edge, so the sub tree of an inner triangle $t \in \mathcal{T}$ is enclosed by the edge $e = \{i, j\} \in E$ shared with its parent triangle and otherwise only by c colored edges. We call e the edge above the sub tree and we denote by $\mathcal{S}^c_e$ the sub tree below the edge e. As the vertices i and j have a c colored path to v_c, the sub tree below e is enclosed by e and the two c paths ${\longrightarrow}_c$ from i to v_c and j to v_j denoted with $i {\longrightarrow}_c v_c$ and $j {\longrightarrow}_c v_c$, respectively. In fact the sub tree contains all triangles within this undirected cycle, since $\mathcal{S}^c$ is a dual spanning tree and it cannot cross any c colored edge. This is depicted in Fig. 7 (left) with an example edge i, j. We use this property in Section 4.2 to show that we can construct two of the colored regions of i by the unification of such sub graphs (already shown on the right).

Similar to the primal graph, each inner triangle $t \in \mathcal{T}$ partitions the vertices into three disjoint dual regions by the three dual paths to the outer face t_{out}. We name the dual regions $\mathcal{L}^c_t \subseteq V$, where c is the color of the dual path not taking part in its boundary. It is also the color of the single boundary vertex v_c within this region. This is depicted in Fig. 8. The boundary of $\mathcal{L}_t^c$ is the set of all inner triangles on the c_1 and c_2-dual paths from t to the outer face. Equivalently, it is the set of all inner triangles $s \in \mathcal{T}$ with at least one vertex in $\mathcal{L}_t^c$ and at least one vertex outside of $\mathcal{L}_t^c$. The dual regions have similar properties to the primal regions. Here we prove the dual of Lemma 5.2 in Schnyder [Schnyder 1990], i.e. that if a triangle s touches the c-colored dual region of another triangle t, then $\mathcal{L}_s^c \subseteq \mathcal{L}_t^c$:

Lemma 4.2. *Let G be a maximal planar graph together with a Schnyder labeling and its corresponding Schnyder realization. Moreover, let $s, t \in \mathcal{T}$ be any two inner triangles such that $s \cap \mathcal{L}_t^c \neq \emptyset$ for some label $c \in \{0, 1, 2\}$. Then, it is true that $\mathcal{L}_s^c \subseteq \mathcal{L}_t^c$.*

Proof. Without loss of generality, we prove it for $c = 2$. Let $x \in \mathcal{T} \cup \{t_{\text{out}}\}$ be the first triangle on the 0-dual path from s to t_{out} that lies also in the boundary of $\mathcal{L}_t^2$. Because of the fixed cyclic ordering of the outgoing dual edges of x, the triangle x cannot be on the 1 dual path from t to t_{out}, so x is on the 0-dual path from t to t_{out} and also $x \neq t$. Since the dual paths of some color cannot form any undirected cycles, the 0-dual path from t to the outer face and from s to the outer face overlap from x on. Similarly, the first $y \in \mathcal{T} \cup \{t_{\text{out}}\}$ on the 1-dual path from s to t_{out} that lies in the boundary of $\mathcal{L}_t^2$ is also on the 1-dual path from t to the outer face and the two 1-dual paths from t and from s to the outer face overlap from y on. This means that every triangle on the boundary of $\mathcal{L}_s^c$ touches $\mathcal{L}_t^c$, which means that $\mathcal{L}_s^c \subseteq \mathcal{L}_t^c$. □

The dual regions and the primal regions are in fact associated through the following relation. All vertices i within the c colored dual region of the triangle t have t in their c colored primal region, or more formally:

Lemma 4.3. *Let G be a maximal planar graph together with a Schnyder labeling and its corresponding Schnyder realization. Moreover, let $t \in \mathcal{T}$ be any inner triangle and $i \in V$ any vertex. Then, $i \in \mathcal{L}_t^c \Leftrightarrow t \in \mathcal{R}_i^c$.*

Proof. Let c_1 and c_2 be the complementary colors of c. Then, given a vertex $i \in \mathcal{L}_t^c$, let $\{u, v\} \in E$ be the first edge on the c_1-path $i {\rightarrow}_{c_1} v_{c_1}$ such that $u \in \mathcal{L}_t^c$ and $v \notin \mathcal{L}_t^c$. Since the c_1 dual path can not cross a c_1 edge, the dual of $\{u, v\}$ must be labeled by c_2. The same holds for the c_2 path $i {\rightarrow}_{c_2} v_{c_2}$ and the c_1 dual path. This means that the c_1 path $i {\rightarrow}_{c_1} v_{c_1}$ and the c_2 path $i {\rightarrow}_{c_2} v_{c_2}$ separate the triangle t from v_c which is equivalent to $t \in \mathcal{R}_i^c$. □

Now that we introduced all necessary data structures, we can describe the mapping M from barycentric space to weight space and its inverse.

4.2 Forward - From weights to barycentric coordinates

Let $\mathbf{w} \in \mathcal{W}$ be some weight vector and let us fix a Schnyder labeling for a maximal planar graph $G = (V, E)$. To calculate the barycentric coordinate vector $\mathbf{b} = M\mathbf{w} \in \mathcal{B}$, we first traverse the dual binary tree of one key color c. Then in the second step we traverse the primal trees of the complementary colors of c. In the following we assume 2 is the chosen key color.

Given an edge $e = \{i, j\} \in E$, that divides a parent and child triangle of $\mathcal{S}^2$, this edge is either 0 or 1, as 2 edges can not divide a child and a parent in $\mathcal{S}^2$. As already explained, the sub tree $\mathcal{S}_e^2$ below e contains all triangles that are enclosed by the undirected cycle e, $j {\rightarrow}_2 v_2$, $v_2 {\leftarrow}_2 i$. Recall that the oriented boundary edges $v_0 v_2$ and $v_1 v_2$ are also considered 2 paths. W.l.o.g. we assume that the edge e is labeled 0 and points from i to j. The 1 region $\mathcal{R}_j^1$ of the vertex j is enclosed by the unoriented cycle $j {\rightarrow}_0 v_0$, $\{v_0, v_2\}$, $v_2 {\leftarrow}_2 j$. As j is in the 0 path $i {\rightarrow}_0 v_0$, the 1 region $\mathcal{R}_i^1$ of the vertex i is enclosed by the unoriented cycle e, $j {\rightarrow}_0 v_0$, $\{v_0, v_2\}$, $v_2 {\leftarrow}_2 i$. This means $\mathcal{R}_i^1 \setminus \mathcal{R}_j^1$ is enclosed by the cycle e, $j {\rightarrow}_2 v_2$, $v_2 {\leftarrow}_2 i$, which is exactly the sub tree $\mathcal{S}_e^2$, i.e. the sub tree below e. This example is shown on the right in Fig. 7. Formally, for every edge $e = \{i, j\} \in E$ colored by $c_1 \in \{0, 1\}$ and oriented from i to j we write

$$\mathcal{S}_e^2 = \mathcal{R}_i^{c_2} \setminus \mathcal{R}_j^{c_2}, \tag{1}$$

where c_1, c_2 are the complementary colors of 2. Inductively, for any given inner vertex $i \in V$ and color $c_1 \in \{0, 1\}$, the union of the sub trees below every c_1 edge along the path $i {\rightarrow}_{c_1} v_{c_1}$ is the c_2 region of i, where c_2 is the third color. Formally $\mathcal{R}_i^{c_2} = \bigcup_{e \in i \rightarrow_{c_1} v_{c_1}} \mathcal{S}_e^2$.

This insight allows for a simple algorithm calculating the barycentric coordinates for all interior vertices at once. First we need to calculate the summed weights s_e of all triangles within the sub tree $\mathcal{S}_e^2$ for all sub trees:

$$s_e = \sum_{t \in \mathcal{S}_e^2} w_t. \tag{2}$$

For $e = \{i, j\} \in E$, let $t = \{i, j, k\} \in \mathcal{T}$ be the root of $\mathcal{S}_e^2$. Then $\mathcal{S}_e^2 = \{t\} \cup \mathcal{S}_{\{j,k\}}^2 \cup \mathcal{S}_{\{k,i\}}^2$ and it follows that

$$s_e = w_t + s_{\{j,k\}} + s_{\{k,i\}}, \tag{3}$$

as $\mathcal{S}_{\{j,k\}}^2$ and $\mathcal{S}_{\{k,i\}}^2$ are disjoint and do not contain t. For this computation, we set $s_e = 0$ for every edge e labeled by 2, or equivalently we can think such an edge as being above an empty sub tree. This can be done in linear time by traversing the 2 dual binary tree in post-order. For our visualizations we store s_e in the edge e.

Note that we can calculate the 1 coordinate of i given the 1 coordinate of its 0 parent j, by adding to it the weight stored in the edge $\{i, j\}$. So, starting with $b_{v_0}^1 = 0$, we traverse T_0 in pre-order and for every oriented edge ij, we compute

$$b_i^1 = b_j^1 + s_{\{i,j\}}. \tag{4}$$

Then, we repeat the same for the 0 coordinates with the 1 primal tree. Lastly, we compute $N = \sum_{t \in \mathcal{T}} w_t$ to be the last entry of $\mathcal{B}$, since, in our notation, $b_i^2 = \sum_{t \in \mathcal{T}} w_t - b_i^0 - b_i^1$ for every $i \in V$. This algorithm is listed in Algorithm 1 and an example of such an embedding is shown in Fig. 9.

4.3 Backward - From barycentric coordinates to weights

Barrera-Cruz et al. [2014] pointed out, that the backward step is possible by solving a linear system of equations. Felsner and Zickfeld [2008] proved that the LSE has full rank. We present a faster algorithm that is even simpler than inverting the steps of the algorithm mentioned in Section 4.2.

ALGORITHM 1: Weights to Barycentric Coordinates

```
Input  : w ∈ $\mathcal{W}$
Output : b ∈ $\mathcal{B}$
sum_weights({$v_0$, $v_1$})      // Traverse $\mathcal{S}^2_{v_0 v_1}$ in post-order.
calc_bary($v_0$, $0$, 0)    // Traverse the 0 tree in pre-order.
calc_bary($v_1$, $1$, 0)    // Traverse the 1 tree in pre-order.
$b^2_{v_2} = \sum_{t \in \mathcal{T}} w_t$

Function sum_weights($e \in E$)
    if $e$ is 2 then
        $s_e := 0$
        return $s_e$
    end
    $\{i, j\} := e$
    $t := \{i, j, k\} \in \mathcal{S}^2_e$          // Get the root of $\mathcal{S}^2_e$.
    $s_e := w_t +$ sum_weights($\{j, k\}$) + sum_weights($\{k, i\}$)
    return $s_e$
end

Function calc_bary($j \in V, c \in \{0, 1\}, b \in \mathbb{Z}$)
    $\{c_o\} := \{0, 1\} \setminus \{c\}$
    $b^{c_o}_j := b$
    for $ij \in T_c$ do          // Iterate over children of $j$.
        calc_bary($i, c, b^{c_o}_j + s_{\{i,j\}}$)
    end
end
```

ALGORITHM 2: Barycentric Coordinates to Weights

```
Input  : b ∈ $\mathcal{B}$
Output : w ∈ $\mathcal{W}$
for $t \in \mathcal{T}$ do
    $w_t := 0$
    /* Iterate over all undirected edges $\{i, j\}$ in $t$. We
       assume $i$ and $j$ appear in positive cyclic order
       within $t$.                                          */
    for $\{i, j\} \in t$ do
        if $\{i, j\}$ is 0 then
            $w_t$ += $(b^1_j - b^1_i)$
        else if $\{i, j\}$ is 1 then
            $w_t$ += $(b^0_i - b^0_j)$
        end
    end
end
```

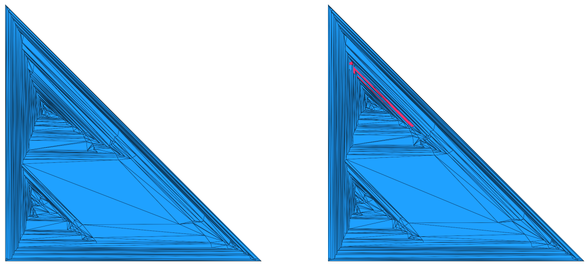


Fig. 9. Two embeddings generated by the forward transformation. The left was generated with all weights set to 1. The right has one weight set to -50. Note that multiple triangles have negative area (red) in the right example.

Let G be a maximal planar graph, together with a Schnyder labelling and its corresponding Schnyder realization. For any barycentric coordinates $\mathbf{b} \in \mathcal{B}$, we compute the corresponding weight assignment $\mathbf{w} = M^{-1}\mathbf{b}$ as follows: Let c_1, c_2 be the complementary labels of 2 and let $e = \{i, j\} \in E$ be any edge corresponding to a dual edge in $\mathcal{S}^2$, labeled by c_1 and oriented as ij in the Schnyder realization. Then, as $b^{c_2}_i = b^{c_2}_j + s_e$, we can easily reconstruct the weight s_e stored in e above the 2 dual binary sub tree $\mathcal{S}^2_e$:

$$s_e = \begin{cases} b^1_i - b^1_j & \text{If } e \text{ is } 0, \\ b^0_i - b^0_j & \text{if } e \text{ is } 1, \\ 0 & \text{if } e \text{ is } 2. \end{cases} \tag{5}$$

Given a triangle $t = \{i, j, k\} \in \mathcal{T}$ with the 2 colored corner at the vertex k, we know that $t \in \mathcal{S}^2_{\{i,j\}}$, by Lemma 4.1ii. Thus, using Eq. (3) we directly get

$$w_t = s_{\{i,j\}} - s_{\{j,k\}} - s_{\{k,i\}}. \tag{6}$$

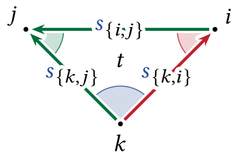
That can be directly calculated in a loop over all triangles, since for each triangle $t \in \mathcal{T}$ we only have to access the Schnyder labeling, in order to find out which of the three edges is the one that contributes positive and how every edge is oriented.

We can simplify the way this computation is written, using Lemma 4.1iii. Let $e = \{i, j\}$ be a 0 labeled edge, oriented as ij. Then its dual edge inside $\mathcal{S}^2$ is oriented from e's right triangle $r = \{i, j, k_1\}$ to e's left triangle $p = \{i, j, k_2\}$. This means that the positive cyclic ordering of r is jik_1, whereas the positive cyclic ordering of p is ijk_2. Also, s_e contributes positive in p and negative in r, because p is the parent of r. So, we get $w_r = b^1_i - b^1_j \pm s_{\{i,k_1\}} \pm s_{\{j,k_1\}}$ and $w_p = b^1_j - b^1_i \pm s_{\{i,k_2\}} \pm s_{\{j,k_2\}}$. So, in general, for some triangle $t = \{a, b, c\}$ with positive cyclic orientation abc, if $\{a, b\}$ is labeled 0, its contribution in w_t is $b^1_b - b^1_a$. Completely symmetrically, it is true that if $\{a, b\}$ is labeled 1, its contribution in w_t is $b^0_a - b^0_b$. This algorithm is listed in Algorithm 2.

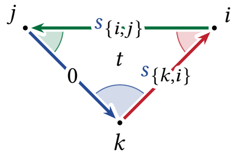
Four of the eight different edge labelings possible and their respective equation for the weights are depicted in Fig. 10. The last four can be calculated respectively, by replacing the top 1 edge $\{i, j\}$ (here oriented as ij) with a 0 edge, oriented in the opposite way ji.

It is therefore also easy to translate the barycentric coordinates of a vertex in weight space, as a change in barycentric coordinates of i only influences the incident triangles of i. In fact it only effects the (at most 6) triangles incident to the out going edges of i. If given a triangle incident to i contains two edges pointing towards i, the barycentric coordinates of i, that would influence the weight of the triangle, cancel out.

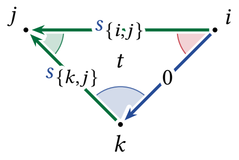
We can use this backwards algorithm to bring any planar triangulation, even those with flipped triangles, into weight space. There we could naively set every negative weight to 1, as only positive weights will lead to a valid embedding. This leads to embeddings that look similar to Schnyder embeddings with weights equal to one as we can see in Fig. 11. We will now present a method that introduces less distortions to the input mesh.



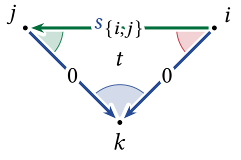
(a) $w_t = b^0_i - b^0_k + b^1_i - b^1_k = b^2_k - b^2_i$



(b) $w_t = b^0_i - b^0_j + b^1_i - b^1_k$



(c) $w_t = b^0_i - b^0_k$



(d) $w_t = b^0_i - b^0_j$

Fig. 10. Four of the eight possible Schnyder realizations for a triangle. To get the other four we just have to flip the orientation of the top edge from ij to ji and change its color to 0. The weight can be retrieved using two or four barycentric coordinates.

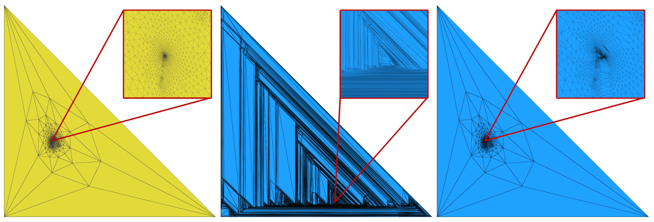


Fig. 11. Tutte embedding fails to embed the swirl mesh #40261 from Thingi10K [Zhou and Jacobson 2016] (left) in doubles. We fixed it with the naive approach, transforming it to weight space and set all negative weights to 1. The resulting embedding (mid) hardly resembles the input. Applying the unflip strategy results in less distored results (right).

5 GENERATING EMBEDDINGS

With the algorithms just described, any maximal planar graph G can be embedded in the plane by choosing arbitrary positive weights $\mathbf{w} > 0$ and sending them through the forward transformation. Conversely, we can take any planar realization of a maximal planar and represent it using weights $\mathbf{w}$. The weights are not necessarily positive even if the planar triangulation is embedded.

The latter transformation is useful for starting with a given planar realization, detecting flipped triangles, and then modifying the weights so that all triangles are correctly oriented. As we show below, this can be done efficiently. A practical prerequisite is, however, that the planar triangulation is represented on a discrete grid. If this is not the case the positions need to be quantized. This may cause additional degeneracy, but this is inconsequential. We discuss the issue of resolution of the discrete grid in Section 6.

Orientation of a triangle. Given a maximal planar graph G, together with a fixed global orientation of G and a mapping of G on the plane, we say that a triangle $t \in G$ has *positive area*, if the positive orientation of t is the same cyclic ordering of its vertices as their CCW ordering in the plane. Given that the boundary of G is a triangle, it cannot be part of any overlapping triangles. If all inner triangles $t \in \mathcal{T}$ have positive area, G is embedded. For any vector of barycentric coordinates $\mathbf{b} \in \mathcal{B}$ this is equivalent to $\det(\mathbf{b}_i, \mathbf{b}_j, \mathbf{b}_k) > 0$ for every $t = \{i, j, k\} \in \mathcal{T}$ with positive cyclic orientation ijk. Let $N = b^0_i + b^1_i + b^2_i = b^0_j + b^1_j + b^2_j = b^0_k + b^1_k + b^2_k$, then it is true that

$$\det(\mathbf{b}_i, \mathbf{b}_j, \mathbf{b}_k) = N \det \begin{bmatrix} b^0_i - b^0_k & b^0_j - b^0_k \\ b^1_i - b^1_k & b^1_j - b^1_k \end{bmatrix} \tag{7}$$

and so, we can always check this smaller determinant.

Increasing weights to create embeddings. It seems that there is no apparent connection between flipped triangles and negative weights. It is easy to see that flipped triangles can have positive weights and negatively weighted triangles can have positive area. All we know is that if all weights are positive then we have a barycentric representation [Felsner and Zickfeld 2008], all triangles have positive area [Felsner and Zickfeld 2008], therefore a valid embedding. This immediately leads to a fast and straightforward but rather naive approach: set all non-positive weights of the planar realization to a positive integer (e.g. 1). This algorithm runs in linear time. The problem with this strategy is that given a 'random' triangulation, half of the weights are negative. Empirically, embedded triangulations or triangulations with only few flipped triangles still have a large number of negative weights. This means a lot of weights change, resulting in a triangulation very different to the input one, e.g. see Fig. 11.

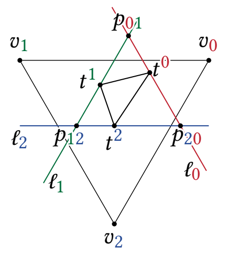
We present a strategy that iteratively increases weights until all triangles have positive area, *without necessarily making them positive*. Several technical results are necessary to show that this will lead to a terminating algorithm.

Fully extended triangles. Let $t \in \mathcal{T}$ be any inner triangle in the triangulation of G. Moreover, let t^0, t^1, t^2 be its three vertices, such that the corner in t at t^c is colored c in the Schnyder labeling. Since the three labels always appear in the cyclic order $0, 1, 2$ in positive orientation of t, we deduce that $t^0 t^1 t^2$ is the cyclic positive orientation of t. Note that the three dual paths starting from t to the outer face separate each vertex of t into a different dual region and moreover, $t^c \in \mathcal{R}^c_t$. Given barycentric coordinates $\mathbf{b} \in \mathcal{B}$, we will call an inner triangle $t \in \mathcal{T}$ *fully extended* with respect to $\mathbf{b}$, if for every $c \in \{0, 1, 2\}$, t^c is the vertex closest to v_c among t^0, t^1, t^2 or, more formally, if the following inequalities are true:

$$\begin{aligned} b^0_{t^0} &> \max\{b^0_{t^1}, b^0_{t^2}\}, \\ b^1_{t^1} &> \max\{b^1_{t^2}, b^1_{t^0}\}, \\ b^2_{t^2} &> \max\{b^2_{t^0}, b^2_{t^1}\}. \end{aligned} \tag{8}$$

Lemma 5.1. *Let G be a maximal planar graph, equipped with a fixed Schnyder labeling and $\mathcal{T}$ be the set of all inner triangles. Moreover, let $\mathbf{b} \in \mathcal{B}$ be a collection of barycentric coordinates and $t \in \mathcal{T}$ be any fully extended inner triangle with respect to $\mathbf{b}$. Then t has positive area.*

Proof. Since $t^0 t^1 t^2$ is the positive cyclic orientation of t, it suffices to show that the points $\mathbf{b}_{t^0}, \mathbf{b}_{t^1}, \mathbf{b}_{t^2}$ in the plane appear in CCW cyclic ordering.



We now divide the plane $\{(x, y, z) \in \mathbb{Z}^3 : x + y + z = 1\}$ using these three lines $\ell_0 : x = b^0_{t^0}$, $\ell_1 : y = b^1_{t^1}$ and $\ell_2 : z = b^2_{t^2}$. These lines are parallel to the boundary edges $\{v_1, v_2\}$, $\{v_2, v_0\}$ and $\{v_0, v_1\}$ respectively, so, in particular, no two of them are parallel to each other. Let p_{01}, p_{12}, p_{20} be the three intersection points. Because of the fixed cyclic ordering of v_0, v_1, v_2, the points p_{01}, p_{12}, p_{20} appear also CCW.

Since t is fully extended, the inequalities in ineqs. (8) imply that for every $c \in \{0, 1, 2\}$, t^c lies on the line ℓ_c, between the two points p_{cc_1} and p_{c_2c}, where c_1, c_2 are the complementary labels of c. This means that t_0, t_1, t_2 also appear in this CCW order. □

Increasing a single weight. For the rest of this section, we will investigate how the barycentric coordinates of any vertex $i \in V$ change if we increase the weight of one triangle $t \in \mathcal{T}$. Given some original $\mathbf{w} \in \mathcal{W}$, let $\mathbf{b} = M\mathbf{w} \in \mathcal{B}$ be the corresponding barycentric coordinates and $\delta \in \mathbb{Z}_{>0}$ be any positive integer. Let now $\mathbf{w}' = \mathbf{w} + \delta e_t$ be the new weight assignment, leaving every weight the same, except $w'_t = w_t + \delta$ and denote by $\mathbf{b}' = M\mathbf{w}' = M(\mathbf{w} + \delta e_t)$ the new barycentric coordinates. Choose $c \in \{0, 1, 2\}$ to be that label, for which $i \in \mathcal{L}^c_t$. Then, the new barycentric coordinates of i are

$$b'^c_i = b^c_i + \delta, \quad b'^{c_1}_i = b^{c_1}_i, \quad b'^{c_2}_i = b^{c_2}_i, \tag{9}$$

where c_1, c_2 are the complementary labels of c. This can easily be seen using the duality $i \in \mathcal{L}^c_t \Leftrightarrow t \in \mathcal{R}^c_i$ and the definition of M. Indeed, the weight change of t only affects the cumulative weight of $\mathcal{R}^c_i$ and thus the value of b'^c_i, leaving the other two coordinates unaffected. Notice that this change also affects the sum of all weights, as $\sum_{t\in\mathcal{T}} w'_t = \sum_{t\in\mathcal{T}} w_t + \delta$.

Proposition 5.2. *Let G be a maximal planar graph, equipped with a fixed Schnyder labeling and $\mathcal{T}$ be the set of all inner triangles. Moreover, let $\mathbf{w} = (w_t)_{t\in\mathcal{T}} \in \mathcal{W}$ be an assignment of weights and $t \in \mathcal{T}$ be any inner triangle. Then, there exists some $\delta_0 \in \mathbb{Z}_{>0}$ such that for every $\delta \in \mathbb{Z}_{>0}$ with $\delta \geq \delta_0$, t is fully extended w.r.t. the new barycentric coordinates $M(\mathbf{w} + \delta e_t) \in \mathcal{B}$.*

Proof. Let $\mathbf{w}' = \mathbf{w} + \delta e_t$, $\mathbf{b} = M\mathbf{w}$ and $\mathbf{b}' = M\mathbf{w}' = M(\mathbf{w} + \delta e_t)$. Since $t^c \in \mathcal{L}^c_t$, it is true that $b'^c_{t^c} = b^c_{t^c} + \delta$, for every $c \in \{0, 1, 2\}$, because of Eq. (9). Also, the sum of all weights is increased by δ as well. This means that:

$$\lim_{\delta\to\infty} \begin{bmatrix} \frac{\mathbf{b}'^T_{t^0}}{\sum_{t\in\mathcal{T}} w'_t} \\ \frac{\mathbf{b}'^T_{t^1}}{\sum_{t\in\mathcal{T}} w'_t} \\ \frac{\mathbf{b}'^T_{t^2}}{\sum_{t\in\mathcal{T}} w'_t} \end{bmatrix} = \lim_{\delta\to\infty} \begin{bmatrix} \frac{b^0_{t^0}+\delta}{\delta+\sum_{t\in\mathcal{T}} w_t} & \frac{b^1_{t^0}}{\delta+\sum_{t\in\mathcal{T}} w_t} & \frac{b^2_{t^0}}{\delta+\sum_{t\in\mathcal{T}} w_t} \\ \frac{b^0_{t^1}}{\delta+\sum_{t\in\mathcal{T}} w_t} & \frac{b^1_{t^1}+\delta}{\delta+\sum_{t\in\mathcal{T}} w_t} & \frac{b^2_{t^1}}{\delta+\sum_{t\in\mathcal{T}} w_t} \\ \frac{b^0_{t^2}}{\delta+\sum_{t\in\mathcal{T}} w_t} & \frac{b^1_{t^2}}{\delta+\sum_{t\in\mathcal{T}} w_t} & \frac{b^2_{t^2}+\delta}{\delta+\sum_{t\in\mathcal{T}} w_t} \end{bmatrix}$$
$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \lim_{\delta\to\infty} \begin{bmatrix} \frac{\mathbf{b}'^T_{v_0}}{\sum_{t\in\mathcal{T}} w'_t} \\ \frac{\mathbf{b}'^T_{v_1}}{\sum_{t\in\mathcal{T}} w'_t} \\ \frac{\mathbf{b}'^T_{v_2}}{\sum_{t\in\mathcal{T}} w'_t} \end{bmatrix}. \tag{10}$$

In other words, by increasing the weight of a triangle we deform the triangle towards the shape of the boundary triangle. In particular, there exists some δ_0, such that for all $\delta \geq \delta_0$, the following inequalities hold simultaneously:

$$\frac{b^0_{t^0} + \delta}{\delta + \sum_{t\in\mathcal{T}} w_t} > \frac{1}{2}, \quad \frac{b^1_{t^1} + \delta}{\delta + \sum_{t\in\mathcal{T}} w_t} > \frac{1}{2}, \quad \frac{b^2_{t^2} + \delta}{\delta + \sum_{t\in\mathcal{T}} w_t} > \frac{1}{2}. \tag{11}$$

Since $b'^0_i + b'^1_i + b'^2_i = \delta + \sum_{t\in\mathcal{T}} w_t = \sum_{t\in\mathcal{T}} w'_t$ for every $i \in \{t^0, t^1, t^2\}$, the ineqs. (11) imply that $b'^c_{t^c} > \frac{1}{2}\sum_{t\in\mathcal{T}} w'_t$ and $b'^{c_1}_{t^c} + b'^{c_2}_{t^c} < \frac{1}{2}\sum_{t\in\mathcal{T}} w'_t$, where c_1, c_2 are the two complementary labels of c. This proves that for every $\delta \geq \delta_0$, t is fully extended with respect to $\mathbf{b}'$. □

In particular, using Lemma 5.1, this means that by continuously increasing the weight of a triangle, this triangle eventually stays unflipped.

Proposition 5.3. *Let G be a maximal planar graph, equipped with a fixed Schnyder labeling and $\mathcal{T}$ be the set of all inner triangles. Moreover, let $\mathbf{w} = (w_t)_{t\in\mathcal{T}} \in \mathcal{W}$ be an assignment of weights, $t \in \mathcal{T}$ be any inner triangle fully extended with respect to $\mathbf{b} = M\mathbf{w} \in \mathcal{B}$ and $s \in \mathcal{T}$ be any other triangle. Then, for every $\delta \in \mathbb{Z}_{>0}$, t remains fully extended with respect to $\mathbf{b}' = M(\mathbf{w} + \delta e_s) \in \mathcal{B}$.*

Proof. We make a case distinction based on how many different dual regions of s contain some vertex of t. We have three cases, as t can touch just one, two of them or all three of them. Let t^0, t^1, t^2 be the usual names of the vertices of t.

(1) In case all vertices of t are in the same dual region of s, they get translated by the same amount which does not change the shape of t. Formally, let $t^0, t^1, t^2 \in \mathcal{L}^c_s$ for some $c \in \{0, 1, 2\}$. Then, according to Eq. (9) only $b'^c_{t^0}, b'^c_{t^1}, b'^c_{t^2}$ change. So, two of the three inequalities in ineqs. (8) remain true in the new coordinates and for the last one we have $b^c_{t^c} + \delta > \max\{b^{c}_{t^{c_1}} + \delta, b^c_{t^{c_2}} + \delta\}$, where c_1, c_2 are the complementary labels of c. This proves that t remains fully extended.
(2) In case t lies in only two dual regions of s, there exists exactly one $c \in \{0, 1, 2\}$ such that $\mathcal{L}^c_s \cap t = \emptyset$. Let c_1, c_2 be the complementary labels of c. Notice that t is on the c dual path from s to the outer face. This means that $\mathcal{L}^{c_1}_t \subseteq \mathcal{L}^{c_1}_s$ and $\mathcal{L}^{c_2}_t \subseteq \mathcal{L}^{c_2}_s$, as proven in Lemma 4.2. Since $t^{c_1} \in \mathcal{L}^{c_1}_t$ and $t^{c_2} \in \mathcal{L}^{c_2}_t$ by definition, we also have that $t^{c_1} \in \mathcal{L}^{c_1}_s$ and $t^{c_2} \in \mathcal{L}^{c_2}_s$. This means that the Ineq. in 8 involving $b'^c_{t^c}$ remains true as is and for the other two we have $b^{c_1}_{t^{c_1}} + \delta > \max\{b^{c_1}_{t^c} + \delta, b^{c_1}_{t^{c_2}}\}$ and $b^{c_2}_{t^{c_2}} + \delta > \max\{b^{c_2}_{t^c} + \delta, b^{c_2}_{t^{c_2}}\}$. Note that only one of $b^{c_1}_{t^c}, b^{c_2}_{t^c}$ is increased by δ, making one of these two inequalities even stronger.
(3) In case every vertex is in a different dual region, we have $s = t$ and for every $c \in \{0, 1, 2\}$, $t^c \in \mathcal{L}^c_s$ by definition. Then, according to Eq. (9), only $b'^0_{t^0}, b'^1_{t^1}, b'^2_{t^2}$ change. So, the three inequalities in ineqs. (8) give us $b^c_{t^c} + \delta > b^c_{t^c} > \max\{b^c_{t^{c_1}}, b^c_{t^{c_2}}\}$, where c_1, c_2 are always the labels complementary to c. This, again, proves that t remains fully extended. □

Newly flipped triangles. Notice that in the case that $t \subseteq \mathcal{L}^c_s$ for some label $c \in \{0, 1, 2\}$, the sign of the area of t does not change, no matter if it is fully extended or not. Intuitively, if t is fully inside the c labeled dual area, the transformation just shifts it towards v_c, without affecting its area. Rigorously, this can be proven, by noticing that such a transformation adds a constant δ to every entry of the c-row of the matrix $(\mathbf{b}_i, \mathbf{b}_j, \mathbf{b}_k)$ and this new matrix has the same determinant as the original. Thus, it is true that newly flipped triangles can only be found on the borders of the three dual areas, i.e. on the three dual paths from s to the outer face.

Fixing mapping into embedding. We now have the necessary insights to see that increasing the weights of a triangle $t \in \mathcal{T}$ will eventually fully extend (Proposition 5.2) and so t is going to eventually have positive area (Lemma 5.1). Once this state is achieved, t will stay fully extended, even if we increase the weights of other triangles (Proposition 5.3). This suggest different strategies for adjusting the weights in order to 'fix' a planar triangulation with flipped triangles. We consider the following main strategy: increase the weights of all flipped triangles to unflip them, then recompute the coordinates and find the new flipped ones. The algorithm is listed in Algorithm 3. For this, we implemented two strategy of increasing the weights. In the first, we fully extend a flipped triangle so that no further check of its area is necessary and in the second we increase the weight of a flipped triangle by the smallest amount needed to unflip this triangle.

Fully Extend Strategy. As we know that once $t \in \mathcal{T}$ is fully extended, t has positive area, even if weights of other triangles are increased later. Therefore we can choose δ for a flipped triangle t to be

$$\delta = 1 + \max \begin{Bmatrix} \max\{b^0_{t^1}, b^0_{t^2}\} - b^0_{t^0}, \\ \max\{b^1_{t^0}, b^1_{t^2}\} - b^1_{t^1}, \\ \max\{b^2_{t^0}, b^2_{t^1}\} - b^2_{t^2} \end{Bmatrix}. \tag{12}$$

This means we only have to adjust the weight of each triangle at most once.

Unflip Strategy. By using a step δ as in Eq. (12), we are "over-correcting". Whilst it is the case that fully extended triangles have positive area, being fully extended is not a necessary condition. This means we could in fact choose a more conservative step that still guarantees that t gets unflipped in the new coordinates, but does not guarantee that the area of t will stay positive if we increase the weight of other triangles afterwards. We can use Eq. (7) to calculate the smallest possible $\delta > 0$ that makes the area of t positive, by solving the following inequality:

$$\det \begin{bmatrix} b^0_{t^0} - b^0_{t^2} + \delta & b^0_{t^1} - b^0_{t^2} \\ b^1_{t^0} - b^1_{t^2} & b^1_{t^1} - b^1_{t^2} + \delta \end{bmatrix} > 0. \tag{13}$$

This gives $\delta > \frac{1}{2}(\sqrt{\operatorname{tr}^2 A - 4 \det A} - \operatorname{tr} A)$, where A is the same 2×2 matrix, before applying the transformation (so for $\delta = 0$). So, we can choose

$$\delta = \left\lfloor \frac{1}{2}\left(\sqrt{\operatorname{tr}^2 A - 4 \det A} - \operatorname{tr} A\right) + 1 \right\rfloor. \tag{14}$$

ALGORITHM 3: Fixing mapping into embedding

```
Input  : $\mathbf{b} \in \mathcal{B}$
Output : $\mathbf{b}' \in \mathcal{B}$
$\mathbf{w} := M^{-1}\mathbf{b}$                       // Compute weights.
$b' := b$
$Q :=$ find_all_flipped_triangles()
while $Q \neq \emptyset$ do
    for $t \in Q$ do
        $\mathbf{w} = \mathbf{w} +$ compute_step($t$)$e_t$
    end
    $\mathbf{b}' = M\mathbf{w}$                    // Update coordinates.
    $Q =$ find_all_flipped_triangles()
end

Function find_all_flipped_triangles()
    $Q := \emptyset$
    for $t \in \mathcal{T}$ do
        /* We assume that the vertices of $t$ are saved in
           the positive cyclic orientation                 */
        $\{i, j, k\} := t$
        if $\det(\mathbf{b}_i, \mathbf{b}_j, \mathbf{b}_k) \leq 0$ then     // If $t$ is flipped.
            $Q = Q \cup \{t\}$
        end
    end
    return $Q$
end

Function compute_step_fully_extend($t \in \mathcal{T}$)
    return $\delta = 1 + \max \begin{Bmatrix} \max\{b^0_{t^1}, b^0_{t^2}\} - b^0_{t^0}, \\ \max\{b^1_{t^0}, b^1_{t^2}\} - b^1_{t^1}, \\ \max\{b^2_{t^0}, b^2_{t^1}\} - b^2_{t^2} \end{Bmatrix}$
end

Function compute_step_unflip($t \in \mathcal{T}$)
    return $\left\lfloor \frac{1}{2}\left(\sqrt{\operatorname{tr}^2 A - 4 \det A} - \operatorname{tr} A\right) + 1 \right\rfloor$
end
```

6 IMPLEMENTATION DETAILS

We implemented E^{3}A in C++ using Eigen [Guennebaud et al. 2010] and Libigl [Jacobson et al. 2018]. The resulting tools have two restrictions: (1) the graph of the triangulations needs to be not only planar, but maximal, meaning the outer face is a triangle with exactly three boundary vertices; (2) if we want to start from a given planar triangulation, the coordinates have to be integers. In the following we explain how we address these restrictions; in addition, we explain the generation of the Schnyder labeling.

6.1 Boundary with more than three vertices

First note that any triangulated topological sphere satisfies the combinatorial requirement if we remove any triangle, i.e., identify it as the boundary. In case we want to use Schnyder's method on any mesh with disk-topology, we need to enclose the boundary with a triangle and then connect the triangle to the boundary. This can be done by connecting each inserted vertex with $\frac{1}{3}$ of the boundary (see Section 7.1 for an experiment with this approach). If we want to

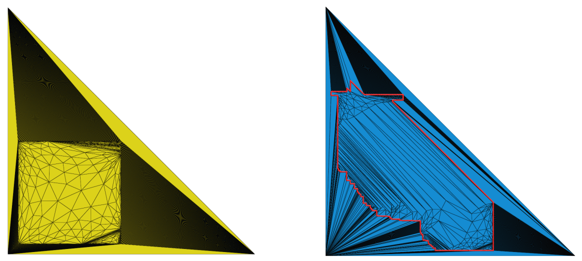


Fig. 12. The Hele-Shaw mesh [Shen et al. 2019] embedded in a square boundary using Tutte's method and then padded using Triangle [Shewchuk 1996] (left). The flipped triangles are fixed using E^{3}A. The original square boundary is distorted in the resulting mesh as our method cannot constrain vertex positions (right).

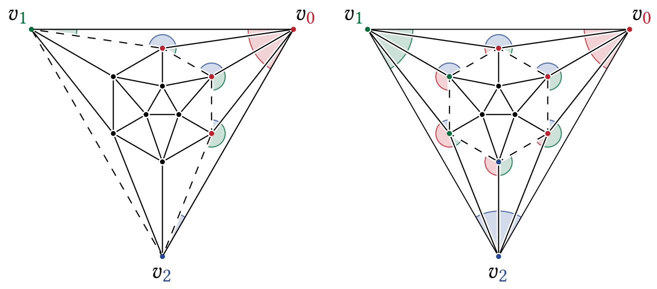


Fig. 13. The labeling algorithm is initialized with the three vertices at the boundary, giving them their respective color. They form a 'frontier' (dashed line). In each step one of the vertices on the frontier gets processed. This process involves labeling incident unlabeled triangles. Every neighboring vertex that has not yet been visited gets added to the boundary and inherits the color of the processed vertex. Left image shows the process after one step, right image after three steps.

use E^{3}A for fixing an already given planar triangulations and a more complex boundary shape is given, the region between the outer triangle and the given boundary can be triangulated [Shewchuk 1996]. As our method is not able to fix any interior vertex, the boundary in the resulting mesh cannot be prescribed, similar to methods that determine the shape of the boundary as part of generating a parameterization [Lévy et al. 2002; Liu et al. 2008; Smith and Schaefer 2015]. It is worth noting that these methods do this intentionally to minimize distortion, the artifact observed in the proposed method is merely a byproduct. Using a naive padding strategy, the resulting boundary geometry may appear undesirable, see Fig. 12.

6.2 Using integers for the coordinates

For turning a given planar triangulation into a set of weights, the coordinates have to be integers. We first normalize coordinates so that they are all contained in $[0,1]^2$. Then each coordinate is multiplied with a fixed quantization factor 2^r and rounded to the nearest integer. We call the integer r the *resolution* and compare among different choices in applications that depend on the resolution.

For possibly transferring the results of our tools to downstream applications we likely need to convert back to floating point numbers. We ensure no rounding errors by setting the resulting $x = b^0$ and $y = b^1$ and then divide them by the smallest power of 2 that is larger than $b^0 + b_1 + b_2$. This ensures that that all vertices are embedded within $[0,1]^2$. The IEEE754 double format has 53 Bits for storing the mantissa, implying that an embedding can be transformed losslessly as long as the sum of weights is below 2^{53} (we could further gain some space by also exploiting the sign).

6.3 Generating a Schnyder labeling

The presented method works with any Schnyder labeling, and we could use any of the published algorithms [Brehm 2000; Schnyder 1990] to generate a labeling in linear time. In our implementation we adopted the method of Brehm [2000]. Instead of constructing the labels starting from one corner of the outer triangle, we start from all three corners at the same time. Context for the following brief explanation may be found in Brehm [2000, Sec. 4.2].

The algorithm of Brehm [2000] selects a key-label, e.g. 0, and forms a 'frontier' initialized with the three boundary vertices. The algorithm then, one by one, processes the vertices on the 'frontier' by removing them from that boundary, adding every non-processed neighbouring vertex to that boundary, while labeling all non-processed triangles such that each corner incident to the current vertex gets labeled with the key label. To prove that this algorithm terminates and results in a valid Schnyder labeling, one needs to show that neither the two non-key color boundary vertices $\{v_1, v_2\}$ get processed nor any vertex that currently has more than two neighbours on the 'frontier'.

Although Brehm [2000] showed that this algorithm is able to generate every possible Schnyder labeling, in practice we have observed that the resulting primary trees are unbalanced, no matter what heuristic is being used to select the next vertex. We therefore extend this method by processing v_1 and v_2 at the same time. Instead of working with a key label, we label vertices on the 'frontier', initializing each boundary vertex $\{v_0, v_1, v_2\}$ with their respective label. When we process a vertex, we label the unlabeled triangles such that the corners incident to the current vertex get the label of the current vertex and color each unlabeled vertex, which is then a new vertex on the 'frontier', with the label of the current vertex. This mimics the extension with this label as key label of the algorithm of Brehm [2000]. We have depicted a possible first and third step of the algorithm in Fig. 13.

As the algorithm processes vertices from all directions using different labels for the extension, some vertices could get processed further using two different labels. Our algorithm implicitly picks a random one (the label that reached that vertex first). This means it is possible that a label vanishes from the 'frontier'. In this case we relabel a vertex on the 'frontier' that can get processed using two different labels. We believe this algorithm can be proved to generate a valid Schnyder labeling, however, leave this for future work. In practice we did not encounter a single failure case in our experiments and could, in any case, fall back to the algorithm by Brehm [2000] if problems were ever encountered.

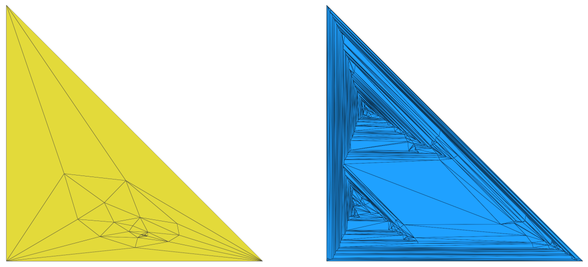


Fig. 14. Comparison of the the two resulting embeddings of the mesh #40261 from Thingi10K [Zhou and Jacobson 2016]. Tutte embedding (left) Schnyder embedding (right).

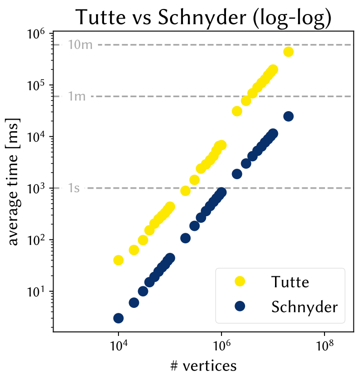


Fig. 15. Comparison of the times of Schnyder and Tutte embeddings in a log-log plot. Each marked time is the average of three runs on an Intel(R) Core(TM) i7-4770K. Our method is an order of magnitude faster.

7 RESULTS AND EVALUATION

In the following we evaluate the E^{3}A tools and compare them against alternatives. First we use Tutte embeddings as well as Schnyder realizations as a starting point for the minimization of symmetric Dirichlet energy and compare the outcomes. For fixing planar realizations with flipped triangles we first evaluate different strategies for our method, followed by a comparison to both Schnyder's method and Progressive Embedding. If not stated otherwise we ran our experiments on an Intel® Core™ i9-12900K with 32 GB of RAM.

7.1 Comparison to Tutte embeddings

Using just the connectivity of the mesh, we can create a Schnyder labeling and choose an arbitrary weight vector **w** to create an embedding in linear time. As Tutte embeddings also just need the adjacency matrix we compare these two methods for creating an arbitrary embedding. Note that these embeddings look completely different as can be seen in Fig. 14. In the Tutte embedding case the majority of vertices cluster in a very small region. Schnyder embeddings usually results in triangles with small shortest-to-largest edge ratios.

Runtime. We compared the runtime of Tutte's and Schnyder's method on different meshes sizes. As input we sampled random points within $[0, 1]^2$ and calculated the Delaunay triangulation. We then enclosed them with a triangle as described above. and ran both methods on the resulting maximal planar graph.

Tutte's method needs to solve a sparse linear system of equations, which we implemented using CHOLMOD [Chen et al. 2008]. This leads to nearly linear runtimes. Nevertheless, our method is roughly 10 times faster. This is illustrated in the log-log plot in Fig. 15.

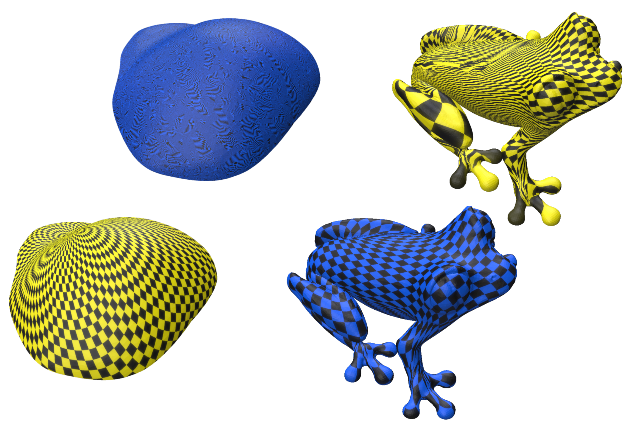


Fig. 16. Comparison of the parametrizations resulting from optimizing the symmetric Dirichlet energy starting from Tutte (yellow) or Schyder (blue) embeddings. The right case is more common, where Schnyder embedding does better than Tutte embedding.

Input for symmetric Dirichlet optimization. Tutte embeddings are often used as input for parametrization optimization methods, as they are fast and simple to compute. The classical Schnyder embeddings are faster to compute, so it might be worth considering Schnyder embeddings as input for further optimization. To test this, we selected 2244 closed meshes with genus 0 from Thingi10K [Zhou and Jacobson 2016], making sure that all faces had non-zero area. Then we generated boundaries by either selecting a random triangle in the mesh, or using a cut-graph as described by Bommes et al. [2009] and implemented in libigl [Jacobson et al. 2018]. The boundaries were laid out on circle for Tutte's method or combinatorially connected to a triangle for boundaries with more than 3 vertices for Schnyder's method. In 243 + 18 of these meshes, Tutte's method failed to produce a usable mesh. Among them are the reported [Shen et al. 2019] 80 meshes with flipped triangles. We then optimized all usable meshes based on symmetric Dirichlet energy with an implementation using TinyAD [Schmidt et al. 2022]. We stopped the optimization after convergence or 1000 iterations. Schnyder's method converged in over 92% of the cases, while Tutte's method failed to converge more often (85% / 65%, triangle / arbitrary boundaries). Although the cost for the embedding step is negligible compared to the optimization, Schnyder embedding was faster or as fast as Tutte embedding in all except for three cases. After stopping, the initial Schnyder embedding produced a lower symmetric Dirichlet energy more often than Tutte's method. For the convergence speed, we took the higher of the two resulting energies and compared the time after each energy has fallen below, in milliseconds. In this terms

Table 1. Comparison between Schnyder embeddings and Tutte embeddings as input for symmetric Dirichlet parametrization optimization on Thingi10K [Zhou and Jacobson 2016] meshes. The numbers tell how often Schnyder's or Tutte's method perform better, for the two cases of using a single triangle of a closed mesh or the existing boundary. The rows provide information on how often the initial embedding *failed*, otherwise which method was *faster*; and for symmetric Dirichlet energy if minimization *converged*, and which method was *faster* and resulted in *smaller energy*.

	Triangle bdy			Arbitrary bdy		
	Schnyder	Same	Tutte	Schnyder	Same	Tutte
Failed	0	0	243	0	0	18
Faster emb.	1165	836	0	1185	1039	3
Converged	177	1673	37	756	1402	11
Faster opt.	1163	370	468	1442	523	262
Lower en.	740	646	615	1137	813	277

Schnyder's method was faster on the majority of meshes. Numbers are listed in Table 1. Two examples, for which one of the resulting parametrizations is worse than the other is depicted in Fig. 16. On the whole, Schnyder's method is more robust than Tutte's method and worth considering as initial embedding.

7.2 Different Strategies for fixing embeddings

Our presentation of multiple strategies for fixing embeddings leads to several possible combinations: Single and batch wise updates can be combined with the unflip or fully extend update rule. These four combinations can be run on different initial resolutions. For our experiments we used three different resolutions 2^r, namely 2^{15}, 2^{30} and 2^{50}.

Runtime. For the single and the batch wise strategy, we used $r = 15, 30, 50$ to compare the running times for both fully-extend and unflip update rule. The results are presented in Fig. 17. These two graphs indicate that the batch wise update strategy is usually an order of magnitude faster than the single update strategy. Also, for the batch-wise update strategy the resolution seems to affect the running time, whereas for the single update strategy to a lesser degree. Lastly, the fully-extend update rule seems to be slightly faster than the unflip update rule.

Single vs. batch. For the energies, we fixed $r = 50$ and the unflip update rule. The results in Fig. 18 indicate that the single update strategy usually results in a shape with better symmetric Dirichlet and Tutte's spring energy. This comes at the cost of run-time.

Fully extended vs. unflip. To compare between the unflip and the fully-extend update rule, we fixed $r = 30$ and batch wise updates. The results regarding the running times and the energies are presented in Fig. 19 and Fig. 20 respectively. The results indicate that, in general, the fully-extend update rule is faster and its Dirichlet energy is better than the unflip update rule for higher resolutions. On the other hand, the unflip update rule tends to have better Tutte's spring energy, especially for high resolutions.

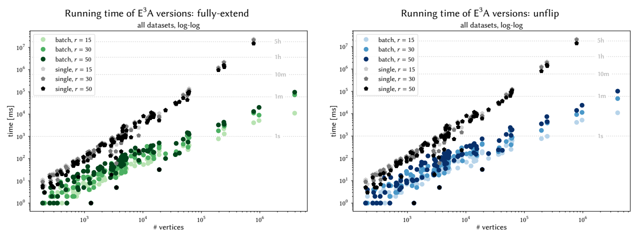


Fig. 17. Run-time difference of single vs. batch-wise updates using the fully-extend (left) and unflip strategy (right) with resolutions $r = 15, 30, 50$. The batch-wise approach is an order of magnitude faster and the fully-extend strategy is just slightly faster than the unflip strategy.

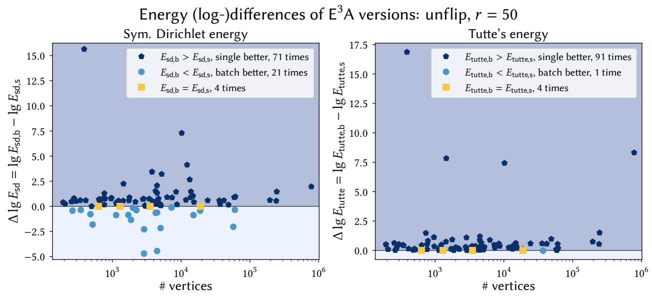


Fig. 18. Comparison between the single and batch-wise update strategy using the unflip update strategy and a resolution of $r = 50$ of the symmetric Dirichlet energy (left) and Tutte's spring energy (right). The single update seems to have a slight advantage.

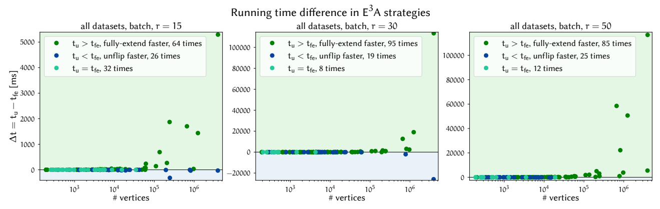


Fig. 19. Run-time comparison of the unflip strategy and fully-extend strategy using batch-wise updates using different resolutions $r = 15$ (left), $r = 30$ (mid) and $r = 50$ (right). The fully-extend strategy is usually faster

7.3 Comparison to Progressive Embeddings

We compared our method for fixing broken Tutte embeddings against Progressive Embedding [Shen et al. 2019]. In their work, the authors identify 80 meshes from Thingi10k of genus 0 for which Tutte's method failed to produce a valid embedding. Besides counting flipped faces, the authors used an additional energy as a measure of face quality. The energy is defined as the symmetric Dirichlet energy wrt. an equilateral of mean face area. They also define a triangle as invalid if it is either flipped or its symmetric Dirichlet is above the threshold of 10^{20}. To provide a fair comparison between our methods, we adopted the use of the symmetric Dirichlet as they define it. In addition, we used a normalized version of Tutte's spring

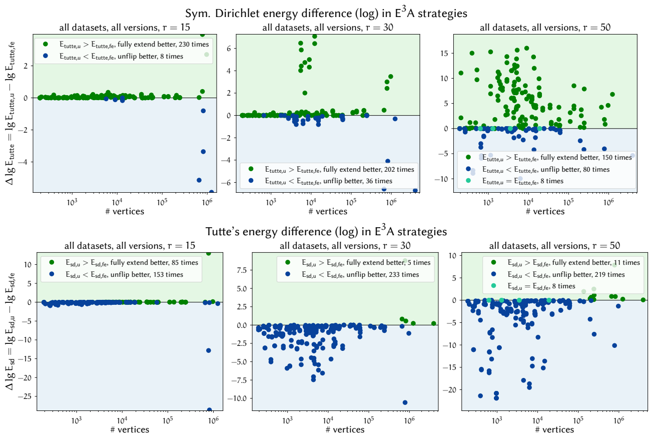


Fig. 20. Energy comparison of the unflip strategy and fully-extend strategy using batch-wise updates using different resolutions $r = 15$ (left), $r = 30$ (mid) and $r = 50$ (right). The fully extend strategy seems to perform slightly better for the Symmetric Dirichlet energy (top) while the unflip energy has usually a lower Tutte's spring energy (bottom).

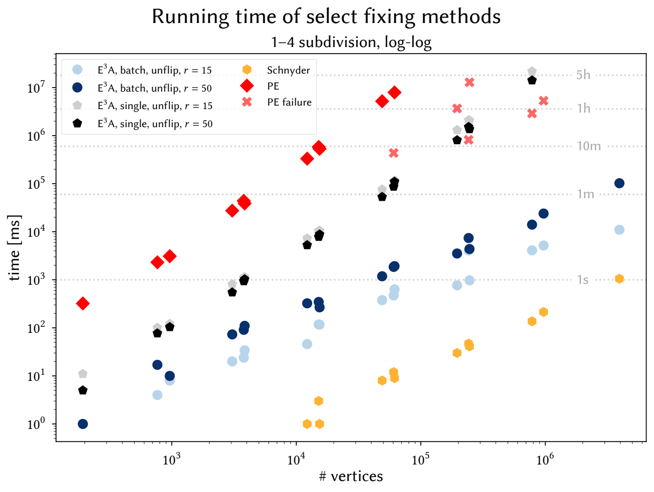


Fig. 21. Runtime comparison between E^{3}A unflip strategy with resolutions 15 and 50 and Progressive Embedding for the different face refinement iterations with up to 3.9 million vertices. Progressive Embedding finished all but the largest case, but did not provide a fixed embedding (shown with an X) starting from around 60 thousand vertices. It might be possible that this is one of the cases discussed by the authors where a solution cannot be guaranteed by their algorithm. In contrast, E^{3}A produced a fixed embedding in all cases with the worst running times staying in the order of minutes.

energy, defined as the summed squared norm of the mean distance of a vertex to its neighbors, divided by the total edge length.

We ran our method against all 80 meshes and compare for each of them whether a fixed embedding was produced, their running time, their maximum symmetric Dirichlet out of all faces for each embedding and their normalized Tutte's spring energy; the results can be seen in Fig. 22. Both Progressive Embedding and E^{3}A managed to produce a fixed embedding for all 80 meshes, but E^{3}A showed a

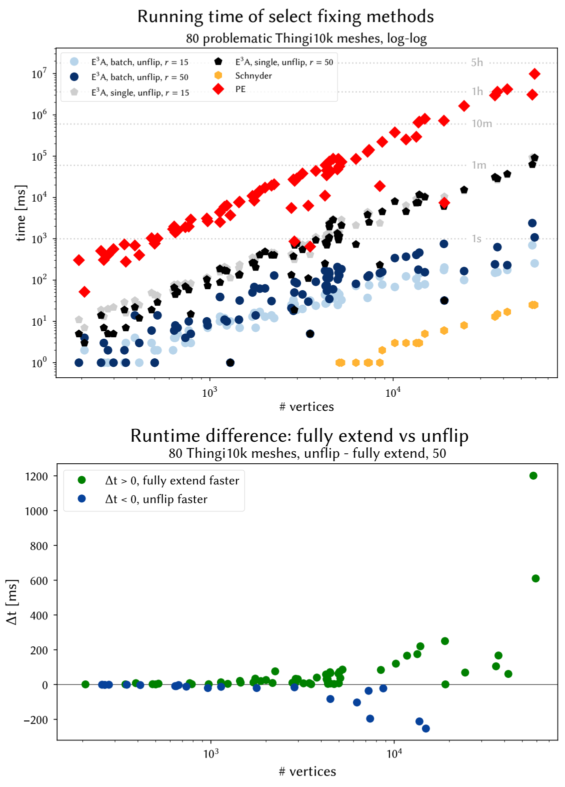


Fig. 22. Runtime comparison between Progressive Embedding [Shen et al. 2019] and E^{3}A unflip strategy with resolutions 15 and 50 for all 80 Thingi10k [Zhou and Jacobson 2016] meshes of genus 0 that break Tutte's method (top) and runtime difference between unflip and fully extend strategy where $\Delta t = t_{\mathrm{u}} - t_{\mathrm{fe}}$ (bottom).

speedup of up to 1000× for both strategies. For the defined energies, we noticed that different discretization resolutions yielded vastly different symmetric Dirichlet energies, with lower resolutions resulting in lower values, around those of Progressive Embedding; and higher resolutions resulting higher values, above Progressive Embedding's threshold. This likely comes from the fact that our strategy only increases the weights by 1 above the minimal needed weight to flip an triangle. This increase of 1 has less impact in higher resolutions, which leads to triangles with small area and therefore acute angles. Nevertheless, we argue that Tutte's spring energy normalized better represents the distortion of each method with respect to an ideally fixed Tutte embedding. In this case, we noticed that an E^{3}A resolution of 50 was able to produce better results that Progressive Embedding in most cases. The resulting energies can be seen in Fig. 23.

To test our method against larger meshes, we took a subset of three small meshes with 193, 963, and 3785 vertices respectively and subdivided their faces with a 1-4 ratio to produce larger meshes which would ideally also break resulting Tutte embeddings. Indeed, we found that the number of flipped triangles produced by Tutte's algorithm roughly corresponded with the scaled total number of faces. This way, we created meshes of up to 3.9 million vertices and

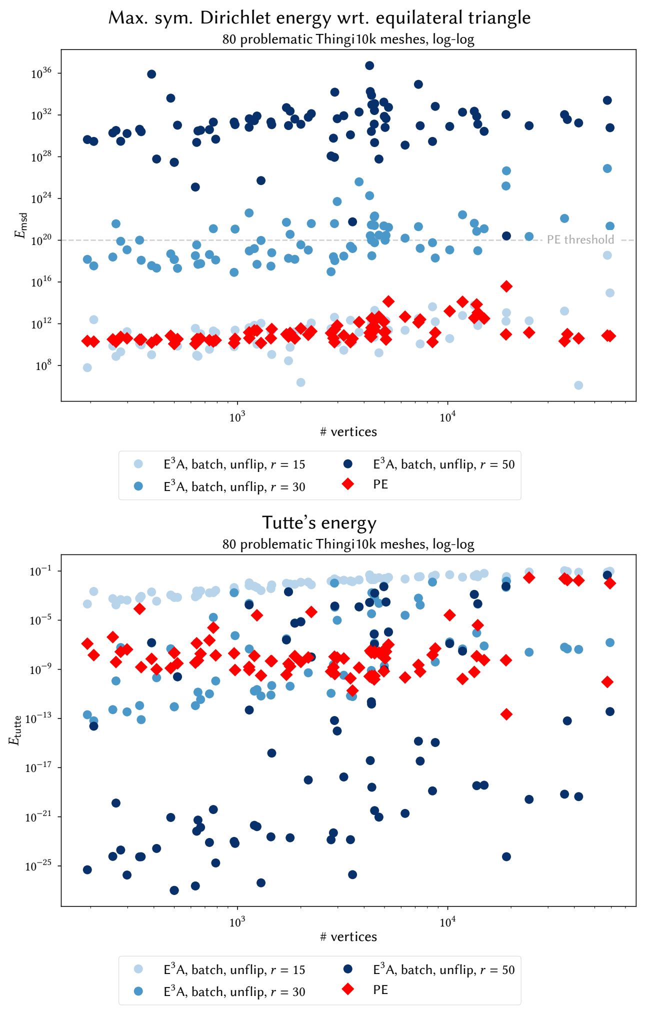


Fig. 23. Comparison of E^{3}A with different starting resolutions and Progressive Embedding [Shen et al. 2019]. On the top we compare their measure of energy (the threshold is marked with a grey line). On the bottom we show Tutte's spring energy. Depending on the starting resolution we can get similar or better results.

ran Progressive Embedding and E^{3}A to fix them. The results can be seen in Fig. 21. Our method was able to fix all of them, taking minutes to fix the most demanding mesh, whereas Progressive Embedding failed to produce an embedding after more than 8 hours. More generally, we found that Progressive Embedding failed to produce a fixed embedding starting at sizes of tens of thousands of vertices.

8 DISCUSSION

We develop a set of tools based on Schnyder labeling and realizers, exploiting the use of weights per triangle. All algorithms run in linear time, are efficient, and can be implemented in integer arithmetic. The particular progress over existing methods based on Schnyder [1990] lies in algorithms that start with existing realizations and modify them to remove flipped triangles. This introduces a small overhead compared to just embedding the triangulation, it allows preserving existing geometry and is still orders of magnitude faster than existing approaches that are also guaranteed to be exact.

While one may argue that any algorithm will in some way be limited by the number of bits used for the representation of coordinates, our tools degrade more gracefully: the size of the representation may affect the shape of the triangles, but it has no effect on the guarantees provided by the algorithms and effects time complexity only by the unavoidable constant factor. This is in contrast to all algorithms working in double we are aware of, which degrade by failing to provide an embedding.

Limitations. The single most significant limitation of our current implementation is the restriction to a triangular boundary. While we can embed more complex boundaries into the triangle, we cannot guarantee that the positions of the vertices on the boundary are preserved. Depending on the application this may or may not be acceptable. One may argue that fixing arbitrary vertices in an embedding may not always be possible without changing the combinatorics. It would still be desirable to provide solutions in case an embedding is possible. We consider further investigation into point constraints important future work.

A more natural limitation, related to the size of integers, follows from the fixing strategy. As the weights are increased, the sum of weights increases, and it cannot be guaranteed that the sum fits into a desired output format, such as the 53 bits offered by IEEE 754 double format.

Possible optimizations. Although our method is already quite fast, there are several optimizations in the data structures that are still possible. For example, it would be possible to avoid recomputing the area of triangles that have been fixed and are known not to be affected by changing the weights of others based on the tree structures.

We also did not make use of the space afforded by different weights for the initial embedding or the different Schnyder labelings. Both have significant effect on the generated embedding.

9 ACKNOWLEDGEMENTS

Early results of this work were presented at the Workshop on Robust Geometric Algorithms for Computational Fabrication hosted and supported by the Fields Institute. The ensuing discussions, in particular with Ilya Baran and Daniele Panozzo, have inspired the further development reported here. We also greatfully acknowledge the many helpful comments of the anonymous reviewers. Funded, in part, by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy–The Berlin Mathematics Research Center MATH+ (EXC-2046/1, project ID: 390685689); and by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101055448, ERC Advanced Grand EMERGE).

REFERENCES

Martin Aigner and Gnter M. Ziegler. 2009. *Proofs from THE BOOK* (4th ed.). Springer Publishing Company, Incorporated.

Luca Castelli Aleardi, Éric Fusy, and Thomas Lewiner. 2009. Schnyder Woods for Higher Genus Triangulated Surfaces, with Applications to Encoding. *Discrete & Computational Geometry* 42, 3 (01 Oct 2009), 489–516. https://doi.org/10.1007/s00454-009-9169-z

Fidel Barrera-Cruz, Penny Haxell, and Anna Lubiw. 2014. Morphing Schnyder Drawings of Planar Triangulations. In *Graph Drawing*, Christian Duncan and Antonios Symvonis (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 294–305.

Fidel Barrera-Cruz, Penny Haxell, and Anna Lubiw. 2019. Morphing Schnyder Drawings of Planar Triangulations. *Discrete & Computational Geometry* 61, 1 (01 Jan 2019), 161–184. https://doi.org/10.1007/s00454-018-0018-9

Giuseppe Di Battista, Roberto Tamassia, and Luca Vismara. 1999. Output-Sensitive Reporting of Disjoint Paths. *Algorithmica* 23, 4 (01 Apr 1999), 302–340. https://doi.org/10.1007/PL00009264

David Bommes, Henrik Zimmer, and Leif Kobbelt. 2009. Mixed-Integer Quadrangulation. *ACM Trans. Graph.* 28, 3, Article 77 (jul 2009), 10 pages. https://doi.org/10.1145/1531326.1531383

Nicolas Bonichon, Stefan Felsner, and Mohamed Mosbah. 2007. Convex Drawings of 3-Connected Plane Graphs. *Algorithmica* 47, 4 (01 Apr 2007), 399–420. https://doi.org/10.1007/s00453-006-0177-6

Enno Brehm. 2000. *3-orientations and Schnyder 3-tree-decompositions.* Master's thesis. FB Mathematik und Informatik, Freie Universität Berlin.

Hervé Brönnimann, Andreas Fabri, Geert-Jan Giezeman, Susan Hert, Michael Hoffmann, Lutz Kettner, Sylvain Pion, and Stefan Schirra. 2023. 2D and 3D Linear Geometry Kernel. In *CGAL User and Reference Manual* (5.5.2 ed.). CGAL Editorial Board. https://doc.cgal.org/5.5.2/Manual/packages.html#PkgKernel23

Marcel Campen, Cláudio T. Silva, and Denis Zorin. 2016. Bijective Maps from Simplicial Foliations. *ACM Trans. Graph.* 35, 4, Article 74 (jul 2016), 15 pages. https://doi.org/10.1145/2897824.2925890

Luca Castelli Aleardi. 2019. Balanced Schnyder Woods for Planar Triangulations: An Experimental Study with Applications to Graph Drawing and Graph Separators. In *Graph Drawing and Network Visualization*, Daniel Archambault and Csaba D. Tóth (Eds.). Springer International Publishing, Cham, 114–121.

Yanqing Chen, Timothy A. Davis, William W. Hager, and Sivasankaran Rajamanickam. 2008. Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate. *ACM Trans. Math. Softw.* 35, 3, Article 22 (oct 2008), 14 pages. https://doi.org/10.1145/1391989.1391995

Marek Chrobak and Goos Kant. 1997. Convex Grid Drawings of 3-Connected Planar Graphs. *Int. J. Comput. Geom. Appl.* 7 (1997), 211–223.

Marek Chrobak and Thomas H. Payne. 1995. A linear-time algorithm for drawing a planar graph on a grid. *Inform. Process. Lett.* 54, 4 (1995), 241–246. https://doi.org/10.1016/0020-0190(95)00020-D

Hubert De Fraysseix, János Pach, and Richard Pollack. 1990. How to draw a planar graph on a grid. *Combinatorica* 10, 1 (01 Mar 1990), 41–51. https://doi.org/10.1007/BF02122694

Raghavan Dhandapani. 2010. Greedy Drawings of Triangulations. *Discrete Comput. Geom.* 43, 2 (mar 2010), 375–392.

Xingyi Du, Noam Aigerman, Qingnan Zhou, Shahar Z. Kovalsky, Yajie Yan, Danny M. Kaufman, and Tao Ju. 2020. Lifting Simplices to Find Injectivity. *ACM Trans. Graph.* 39, 4, Article 120 (aug 2020), 17 pages. https://doi.org/10.1145/3386569.3392484

Xingyi Du, Danny M. Kaufman, Qingnan Zhou, Shahar Kovalsky, Yajie Yan, Noam Aigerman, and Tao Ju. 2022. Isometric Energies for Recovering Injectivity in Constrained Mapping. In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Republic of Korea) *(SA '22)*. Association for Computing Machinery, New York, NY, USA, Article 36, 9 pages. https://doi.org/10.1145/3550469.3555419

Stefan Felsner. 2001. Convex Drawings of Planar Graphs and the Order Dimension of 3-Polytopes. *Order* 18, 1 (01 Mar 2001), 19–37. https://doi.org/10.1023/A:1010604726900

Stefan Felsner and Florian Zickfeld. 2008. Schnyder Woods and Orthogonal Surfaces. *Discrete & Computational Geometry* 40, 1 (01 Jul 2008), 103–126. https://doi.org/10.1007/s00454-007-9027-9

Michael S. Floater and Kai Hormann. 2005. Surface Parameterization: a Tutorial and Survey. In *Advances in Multiresolution for Geometric Modelling*, Neil A. Dodgson, Michael S. Floater, and Malcolm A. Sabin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 157–186.

Vladimir Garanzha, Igor Kaporin, Liudmila Kudryavtseva, François Protais, Nicolas Ray, and Dmitry Sokolov. 2021. Foldover-Free Maps in 50 Lines of Code. *ACM Trans. Graph.* 40, 4, Article 102 (jul 2021), 16 pages. https://doi.org/10.1145/3450626.3459847

Torbjörn Granlund et al. 2020. *GNU MP: The GNU Multiple Precision Arithmetic Library* (6.2.1 ed.). https://gmplib.org/

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

Xin He and Huaming Zhang. 2010. Schnyder Greedy Routing Algorithm. In *Theory and Applications of Models of Computation*, Jan Kratochvíl, Angsheng Li, Jivrí Fiala, and Petr Kolman (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 271–283.

Christoph Martin Hoffmann. 1989. The problems of accuracy and robustness in geometric computation. *Computer* 22, 3 (1989), 31–39. https://doi.org/10.1109/2.16223

Fáry István. 1948. On straight-line representation of planar graphs. *Acta scientiarum mathematicarum* 11, 229-233 (1948), 2.

Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. https://libigl.github.io/.

Goos Kant. 1996. Drawing planar graphs using the canonical ordering. *Algorithmica* 16, 1 (01 Jul 1996), 4–32. https://doi.org/10.1007/BF02086606

Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. 2002. Least Squares Conformal Maps for Automatic Texture Atlas Generation. *ACM Trans. Graph.* 21, 3 (jul 2002), 362–371. https://doi.org/10.1145/566654.566590

Yaron Lipman. 2012. Bounded Distortion Mapping Spaces for Triangular Meshes. *ACM Trans. Graph.* 31, 4, Article 108 (jul 2012), 13 pages. https://doi.org/10.1145/2185520.2185604

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. 2008. A Local/Global Approach to Mesh Parameterization. In *Proceedings of the Symposium on Geometry Processing* (Copenhagen, Denmark) *(SGP '08)*. Eurographics Association, Goslar, DEU, 1495–1504.

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable Locally Injective Mappings. *ACM Trans. Graph.* 36, 4, Article 37a (jul 2017), 16 pages. https://doi.org/10.1145/3072959.2983621

Rohan Sawhney and Keenan Crane. 2017. Boundary First Flattening. *ACM Trans. Graph.* 37, 1, Article 5 (dec 2017), 14 pages. https://doi.org/10.1145/3132705

Stefan Schirra. 2000. Robustness and Precision Issues in Geometric Computation. In *Handbook of Computational Geometry*, Jörg-Rüdiger Sack and Jorge Urrutia (Eds.). Elsevier, Amsterdam, The Netherlands, 597–632.

Patrick Schmidt, Janis Born, David Bommes, Marcel Campen, and Leif Kobbelt. 2022. TinyAD: Automatic Differentiation in Geometry Processing Made Simple. *Computer Graphics Forum* 41, 5 (2022), 113–124. https://doi.org/10.1111/cgf.14607

Walter Schnyder. 1989. Planar graphs and poset dimension. *Order* 5, 4 (01 Dec 1989), 323–343. https://doi.org/10.1007/BF00353652

Walter Schnyder. 1990. Embedding Planar Graphs on the Grid. In *Proceedings of the First Annual ACM-SIAM Symposium on Discrete Algorithms* (San Francisco, California, USA) *(SODA '90)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 138–148.

Walter Schnyder and William Thomas Trotter. 1992. Convex embeddings of 3-connected plane graphs. *Abstracts of the AMS* 13 (1992), 502.

Alla Sheffer, Emil Praun, and Kenneth Rose. 2007. Mesh Parameterization Methods and Their Applications. *Foundations and Trends® in Computer Graphics and Vision* 2, 2 (2007), 105–171. https://doi.org/10.1561/0600000011

Hanxiao Shen, Zhongshi Jiang, Denis Zorin, and Daniele Panozzo. 2019. Progressive Embedding. *ACM Trans. Graph.* 38, 4, Article 32 (jul 2019), 13 pages. https://doi.org/10.1145/3306346.3323012

Jonathan Richard Shewchuk. 1996. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In *Applied Computational Geometry Towards Geometric Engineering*, Ming C. Lin and Dinesh Manocha (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 203–222.

Jonathan Richard Shewchuk. 1997. Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates. *Discrete & Computational Geometry* 18, 3 (1997), 305–363. https://doi.org/10.1007/PL00009321

Hang Si. 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator. *ACM Trans. Math. Softw.* 41, 2, Article 11 (feb 2015), 36 pages. https://doi.org/10.1145/2629697

Jason Smith and Scott Schaefer. 2015. Bijective Parameterization with Free Boundaries. *ACM Trans. Graph.* 34, 4, Article 70 (jul 2015), 9 pages. https://doi.org/10.1145/2766947

Sherman Kopald Stein. 1951. Convex Maps. *Proc. Amer. Math. Soc.* 2, 3 (1951), 464–466.

Jian-Ping Su, Chunyang Ye, Ligang Liu, and Xiao-Ming Fu. 2020. Efficient Bijective Parameterizations. *ACM Trans. Graph.* 39, 4, Article 111 (aug 2020), 8 pages. https://doi.org/10.1145/3386569.3392435

William Thomas Tutte. 1960. Convex Representations of Graphs. *Proceedings of the London Mathematical Society* s3-10, 1 (1960), 304–320. https://doi.org/10.1112/plms/s3-10.1.304

William Thomas Tutte. 1963. How to Draw a Graph. *Proceedings of the London Mathematical Society* s3-13, 1 (1963), 743–767. https://doi.org/10.1112/plms/s3-13.1.743

Klaus Wagner. 1936. Bemerkungen zum vierfarbenproblem. *Jahresbericht der Deutschen Mathematiker-Vereinigung* 46 (1936), 26–32.

Chee-Keng Yap. 1997. Towards exact geometric computation. *Computational Geometry* 7, 1 (1997), 3–23. https://doi.org/10.1016/0925-7721(95)00040-2

Qingnan Zhou and Alec Jacobson. 2016. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv preprint arXiv:1605.04797* (2016).