Computers & Graphics (2021)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

A multigrid approach for generating harmonic measured foliations

Shaodong Wang[a,b,*], Shuai Ma[a,b], Hui Zhao[a,b], Wencheng Wang[a,b,**]

[a] *State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences*
[b] *University of Chinese Academy of Sciences*

ARTICLE INFO

Article history:

Keywords: Geometry Processing, Harmonic Measured Foliation, Multigrid Solver

ABSTRACT

Harmonic measured foliation has demonstrated its usefulness for many geometric problems, including conformal parameterization and mesh quadrangulation. Due to the non-linearity and hard constraints for its computation, existing iterative solvers converge very slowly, and so impractical for large meshes. Though the multigrid approach is well-known for speeding up iterative solvers, a general multigrid solver cannot be applied here in a plug-and-play fashion, because the constraints for computing the harmonic measured foliations would be broken. In this article, we design a novel multigrid solver for this problem, where we propose specific multi-resolution mesh hierarchies and interpolation schemes to fulfill the requirements of the harmonic measured foliation. Experimental results show that our multigrid solver converges much faster than the original algorithm on meshes ranging from a few thousands to over one million edges, even by over a hundred times. This would benefit scalable geometry processing using harmonic measured foliations.

1. Introduction

The *foliation* is a geometric structure which decomposes a manifold into lower-dimensional submanifolds called its *leaves*. Due to its structural properties, the foliation has drawn growing attentions in geometry processing and shape analysis in recent years, including surface mapping [1], volume mapping [2], and computational fabrication [3].

Based on the profound theoretical relationship with complex analytic functions, a special class of foliation, called the *harmonic measured foliation*, can be used to compute global conformal parameterizations of surfaces, which is very useful for conformal texture mappings [4], generating quadrilateral meshes with regular structures [5], and computing correspondence between surfaces [6]. To this end, various methods [7, 4, 5] have been proposed to compute the harmonic measured foliation on surfaces. Among them, the method proposed by Palmer [7] is very effective by representing the discrete measured foliations of meshes as an edge-valued function, similar to a discrete 1-form [8], as the measured foliation is related to differential 1-form. Afterwards, the method is further improved in [4], and it has the advantage of representing the conformal parameterization without changing the connectivity of the original mesh like the comparative method [5]. Unfortunately, due to the non-linear nature of the problem for generating harmonic measured foliations, the methods [7, 4] can only be solved in an iterative fashion and cannot scale well to large meshes.

Motivated by the effectiveness of the multigrid approach in accelerating iterative solvers, we propose a novel method to improve the efficiency of [7, 4]. The basic idea for the multigrid approach is to solve the original problem on a hierarchy of coarse-to-fine grids. For this, the simplest paradigm, called the *cascadic* multigrid, is to first solve the problem on the coarsest grid, and then interpolate the results of the coarser grids

*Corresponding author.
**Corresponding author.
This is the author's version. Published version at: https://www.sciencedirect.com/science/article/pii/S0097849321002144
e-mail: wangsd@ios.ac.cn (Shaodong Wang), whn@ios.ac.cn (Wencheng Wang)

iteratively to the finer grids (called *prolongation*) to speed up convergence. Thus, the core to the multigrid method is to properly construct a multigrid hierarchy and interpolate the function values between different levels of the hierarchy. In computer graphics, existing multigrid methods for triangular surface meshes [9, 10, 11] mainly construct the multigrid hierarchy by mesh simplification using edge collapsing, and interpolate the values using some locally-weighted-average measures. However, the existing multigrid methods cannot be used to generate harmonic measured foliations, as discussed below.

- First, these methods all focus on solving linear systems for scalar functions defined on mesh vertices, while the variables of harmonic measured foliations are defined on mesh edges.
- Second, the optimization of harmonic measured foliation must satisfy several hard constraints, which are strictly preserved by the iteration proposed in [7] (to be discussed in details in Section 3). However, when interpolating the values of the coarser grids to the finer grids in the multigrid approach, simply mimicking existing strategies, such as one-ring averaging, will invalidate the constraints and make the foliation incorrect at the next level.
- Third, the algorithm [7, 4] only guarantees to converge on Delaunay meshes due to its use of cotangent weights, but edge collapsing will break the Delaunay property of the input mesh, which may lead the algorithm to not converge on coarser grids.

Therefore, it is vital to carefully design the multigrid hierarchy as well as the interpolation strategy for correctly computing the harmonic measured foliation with the multigrid method. In this article, we propose a novel cascadic multigrid method for generating harmonic measured foliations [7, 4], consisting of two key ingredients. First, we build the multigrid hierarchy by performing the edge flipping algorithm [12] at each level after edge collapsing, for obtaining a Delaunay mesh at each level. Second, we present a set of constraint-preserving interpolation measures to correctly prolong the foliation values from coarser grids to finer grids, with respect to both edge collapsing and edge flipping. With extensive experiments, we demonstrate that our algorithm can achieve an acceleration of several orders of magnitudes on large meshes, as shown in the example Fig. 1. As a result, our method makes the harmonic measured foliation much more scalable to large meshes, which are always required in applications.

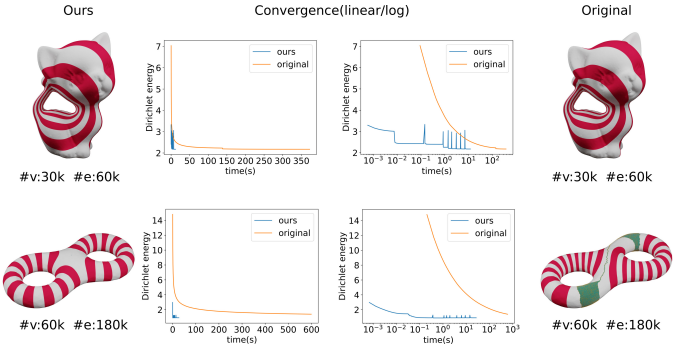


Fig. 1. Comparing our method with the original algorithm [7, 4]. On the first row, both methods generate similar results, but our multigrid solver converges much faster than the original algorithm. On the second row, our approach produces a harmonic foliation in less than 10 seconds, but the original algorithm is far from convergence even after 10 minutes. In this article, the foliation is visualized as red-white stripes following the method in [4], and the colored spheres and triangles illustrate the singularities.

2. Related work

2.1. Discrete foliations

Foliations arise in many fields of mathematics including dynamic systems and complex analysis. Below we focus on discrete foliations in geometry processing, especially the discrete harmonic measured foliations.

As mentioned in Section 1, the foliation on a manifold decomposes it into lower dimensional submanifolds (its leaves). In computer graphics, most works study 2D (surfaces) and 3D (volumes) manifolds, so the leaves are often 1D curves or 2D surfaces. Till now, many works have been done for facilitating many tasks., e.g., computing the one-dimensional simplicial foliations to build bijective maps between surfaces and volumes [1], using the two-dimensional generalized foliations to build maps between volume meshes [2], and computing the geodesic foliations on surfaces for benefiting the design of weaving structures [3].

To our knowledge, the measured foliation has received much attention as it has many rich structures [13] and equivalent to the horizontal trajectories of a holomorphic quadratic differential [14], by which the measured foliations can be used to compute conformal parameterizations on high-genus meshes, since the holomorphic quadratic differential is equivalent to a conformal flat metric with cones. There are two major lines of algorithms to compute harmonic measured foliations on surface meshes, as discussed below.

Lei et al. [15, 16, 5] propose to compute a harmonic map between the cylinder graph of the surface to the surface itself using the non-linear heat flow algorithm [17]. Then the surface is segmented into several cylinders, and the harmonic measured foliation is treated as harmonic functions in each cylinder. This algorithm has been applied to compute quadrilateral and hexahedral meshes [15, 16, 5] as well as surface correspondence [6]. But this algorithm has its downside that the mesh has to be sliced into cylinders so that its original connectivity may be changed.

Besides the algorithms mentioned in the above paragraph, Palmer [7] proposes to directly represent the discrete measured foliations as scalar values defined on mesh edges, and develops an algorithm for generating harmonic measured foliations. This is based on the definition of the measured foliation [13], stating that the measured foliation is locally equivalent to a differential 1-form. However, the discrete harmonic measured foliation cannot be computed efficiently with a linear system like the 1-forms [18]. Instead, Palmer [7] formulates the problem as a non-linear optimization and proposes an iterative algorithm to optimize it. First, an initial foliation is prescribed by a set of disjoint, non-trivial, and non-freely-homotopic triangle loops,

where the common edges between consecutive triangles of a loop are assigned values. Then, the values on the one-ring edges of a vertex are updated by a so-called discrete Whitehead move, such that the energy is minimized while maintaining the constraints of the measured foliation. This algorithm is further improved by Zhao et al. [4] for its robustness by constraining the order of vertex traversal to avoid creating extra singularities. Since the triangle loops generally cover a very small portion of the whole mesh, it would take a long time to obtain the harmonic measured foliation over the mesh. With the mesh becoming larger, the convergence of this algorithm will become much slower. So this algorithm does not scale well to large meshes which are commonly seen in real world applications. In this article, we try to speed up the computation of [7, 4] for well handling large meshes.

2.2. Multigrid methods

The multigrid approach does not refer to a single numerical method, but a whole spectrum of techniques based on the multi-resolution strategy. In the following, we mainly discuss the design and application of multigrid methods in computer graphics, based on the domains of their grids, including the geometric multigrid (structured and unstructured) and algebraic multigrid. For a more comprehensive understanding, we refer readers to the book [19].

Structured grids like pixels and voxels are the simplest domains. They admit a natural multigrid hierarchy thanks to the consistent local neighborhoods, and the values are often interpolated simply by bilinear or trilinear interpolations. They have been widely applied to efficient image processing [20], fluid simulation [21, 22], and surface reconstruction [23, 24, 25]. However, these grids are too simple to represent complex domains like surfaces.

Unstructured grids like triangular and tetrahedral meshes are often used to represent more complex shapes. For Euclidean domains like planar surfaces [26] and volumes [27], it is common to first fix the boundary mesh elements for each level, and then triangulate the interior. Then the function values are generally interpolated using barycentric coordinates. For curved surfaces, which is also the main focus of this paper, the multigrid hierarchy is often built by mesh simplification using edge collapsing [28]. However, since the simplified mesh no longer lies on the same surface as the finer mesh, simple barycentric interpolation [9, 29] or one-ring averaging [10] may induce very large errors. Recently, Liu et al. [11] propose an intrinsic multigrid method to jointly parameterize the local neighborhood before and after an edge collapsing operation to a common domain, by which the values can be interpolated with respect to the intrinsic geometry, so the iterations can converge much faster. But these methods cannot be applied to compute the harmonic measured foliation, as discussed in Section 1.

On the other hand, there are algebraic multigrid methods that build the hierarchy by exploring the sparsity of the matrix in the linear system, and saving the necessity of dealing with the actual geometric grids like the above methods. This has the advantage of totally avoiding the complexity of the unstructured multigrid methods, and has been applied in shape deformation [30] and cloth simulation [31]. However, existing algebraic multigrid methods also cannot be directly applied to generate the harmonic measured foliations, because it is not clear how to maintain the constraints during the interpolation in the algebraic multigrid methods.

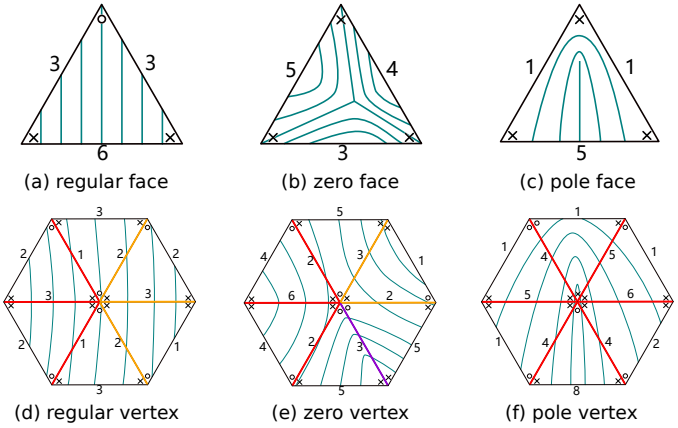


Fig. 2. Illustration of the discrete measured foliations and related structures. The leaves of the foliation are illustrated by green curves. The numbers on the edges show the possible foliation values. The closed corners are marked by ◯, and non-closed corners by ×. Colored edges in (d-f) are used to indicate different sectors. (This figure is taken from [4](Fig. 2).)

3. Discrete harmonic measured foliations

For the sake of self-completeness, we review the key results of [7, 4], which paves the way to our multigrid method.

3.1. Notations

In this paper, we represent a closed manifold surface $\mathcal{M}$ as a triangular mesh $M = (V, E, T)$, where V, E, T are the set of vertices, edges, and triangles of the mesh. A vertex is denoted as $v_i \in V$; An edge between v_i and v_j is denoted as $e_{ij} \in E$; A triangle formed by v_i, v_j, v_k is denoted as $t_{ijk} \in T$; A corner between e_{ij} and e_{jk} in t_{ijk} is denoted as $\angle ijk$.

3.2. Definitions

Basic definitions. The *discrete measured foliation* is defined as a map $F : E \to \mathbb{R}_{\geq 0}$ [7], where a non-negative scalar value is assigned to each edge of the mesh, called the *foliation value* F_{ij} on edge e_{ij}, as illustrated in Fig. 2. The foliation value represents the path integral of the smooth foliation on this edge by analogy with the discrete differential 1-form. A discrete measured foliation is *harmonic* if it is both *closed* and *coclosed* [7], as discussed below.

Closedness. In [7], the foliation is required to be *closed*, meaning that it is locally integrable. As studied in [7], the foliation can be *regular* or *singular* at a face or vertex, and a singularity can be further classified as a *zero* or a *pole*, as shown in Fig. 2. A foliation is closed if it does not contain poles [7]. The zeros and poles of the foliation can be determined on faces and vertices as explained below.

A corner $\angle ijk$ is said to be closed for a triangle t_{ijk} if $F_{ij} + F_{jk} = F_{ki}$, and marked with a ◯ symbol as shown in

Fig. 2, otherwise it is not closed and marked with a × symbol [7]. The *index* $i_t(F)$ is used to indicate whether a triangle contains a singularity, which is computed as,

$$i_t(F) = 2 - X(t) - Z(t) \tag{1}$$

where $X(t)$ denotes the number of non-closed corners in t and $Z(t)$ denotes the number of edges having zero foliation values in t [7]. If $i_t(F) = 0$, then the foliation is regular in that face, as shown in Fig. 2(a), otherwise it contains a singularity. If the following triangle inequalities hold,

$$F_{ij} < F_{jk} + F_{ki}, \quad F_{jk} < F_{ki} + F_{ij}, \quad F_{ki} < F_{ij} + F_{jk} \tag{2}$$

then the triangle contains a zero of the foliation, as shown in Fig. 2(b). And if not, the face contains a pole of the foliation, as shown in Fig. 2(c). In a harmonic foliation, poles are not permitted.

Similarly, the index of the foliation on a vertex $i_v(F)$ is computed as

$$i_v(F) = 2 - O(v) + Z(v) \tag{3}$$

where $O(v)$ denotes the number of closed corners around v, and $Z(v)$ denotes the number of one-ring edges having zero foliation values around v [7]. According to $i_v(F) = 0, < 0, > 0$, the vertex v can be determined as a regular point, zero, or pole of the foliation, as shown in Fig. 2(d-f).

Coclosedness. In [7], the one-ring edges $\mathcal{E}_v$ around a vertex v are separated into different *sectors* by the closed corners adjacent to v, illustrated by different colors in Fig. 2(d-f). Let us denote the k-th sector of vertex v as S_v^k, then the foliation is said to be *coclosed* at v if Eq. (4) holds for each sector S_v^k,

$$\sum_{e_{ij} \in S_v^k} \alpha_{ij} F_{ij} \leq \frac{1}{2} \sum_{e_{ij} \in \mathcal{E}_v} \alpha_{ij} F_{ij} \tag{4}$$

where α is cotangent weight [32] of the mesh. The foliation is said to be coclosed if it is coclosed on all vertices.

3.3. Algorithm

Based on the definition of harmonic measured foliation stated in the last subsection, Palmer [7] proposes to iteratively optimize an initial closed foliation to be coclosed in order to generate a harmonic measured foliation. Particularly, their algorithm can maintain the *Whitehead* class [13] of the foliation, which ensures that the algorithm always converges to the same result given compatible inputs (detailed below). Its workflow is illustrated in Fig. 3.

Initialization. The initialization to the algorithm is a closed foliation. It is initialized by a set of disjoint, non-trivial, and non-freely-homotopic triangle loops $L = \{l_i\}$. For each loop $l_i \in L$, the same value w_i is assigned to the common edges between consecutive triangles in l_i as their foliation values. For the other edges, their foliation values are all zeros. As a result, we obtain a closed foliation whose leaves are contained within the input loops, as shown in Fig. 3(a). According to [4], these loops can be easily drawn by a user interactively, or computed using non-trivial loops on surfaces such as the homology basis [33, 34].

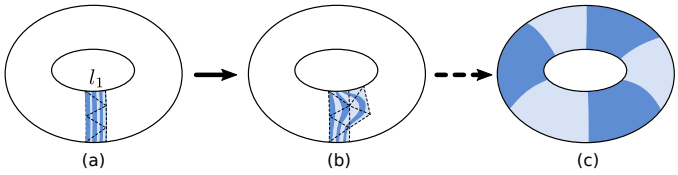


Fig. 3. The framework of algorithm [7]. (a) An input loop is used to initialize a closed foliation. (b) With a discrete Whitehead move, the foliation values around a vertex is updated for the foliation to be locally harmonic here. (c) After enough iterations, a harmonic measured foliation that is Whitehead equivalent to the initial foliation is generated.

Optimization. This is for generating the measured foliations on the mesh by the initialization in loops. This problem is formulated as minimizing the Dirichlet energy of the foliation [7],

$$D(F) = \sum_{e_{ij} \in E} \alpha_{ij} F_{ij}^2 \tag{5}$$

whose minimizer realizes the coclosed condition. In order to maintain the Whitehead class of the foliation, Palmer [7] proposes to use a particular line search direction to minimize the energy, called a *Whitehead move*. In a Whitehead move, the foliation values on $\mathcal{E}_v$ for a vertex v are updated by subtracting a value μ from the foliation values of edges in a selected sector of v, and adding μ to the foliation values of edges in its other sectors, as shown in Fig. 3(b). Here, the selected sector should have its Dirichlet energy larger than the summed Dirichlet energy of other sectors, and μ is computed by the weighted difference between these two Dirichlet energies. For the details, please refer to [7]. With the Whitehead moves applied iteratively, the algorithm will converge to achieve a harmonic measured foliation, as shown Fig. 3(c).

This algorithm requires two conditions: α is positive on all edges, and no poles are created during the optimization [7]. The first condition can be satisfied by remeshing the input mesh to be Delaunay using algorithms like [12]. But the second condition is not always fulfilled by [7], which is solved in [4] by restricting the order of vertex traversal in the optimization to totally avoid creating poles in the process. In this article, we use the improved algorithm of [4] for generating harmonic measured foliations.

4. The cascadic multigrid solver

Here, we introduce our multigrid solver for generating harmonic measured foliations. Among the solving schemes like the V-cycle and F-cycle [19], we choose the simple scheme called the cascadic multigrid (CMG) method. In the CMG method, a solution is first computed on the coarsest level, and then the solution of a coarser level is iteratively taken as the initialization for the next finer level by interpolation to achieve the solution on that level, until the final solution is obtained on the finest level. The CMG solver is simple to implement, since it only involves interpolation from the coarser grid to the finer grid, and the optimization is solved at each level only once. Besides, the CMG method is efficient when the coarser grid solution provides a good approximation to the finer grid solution

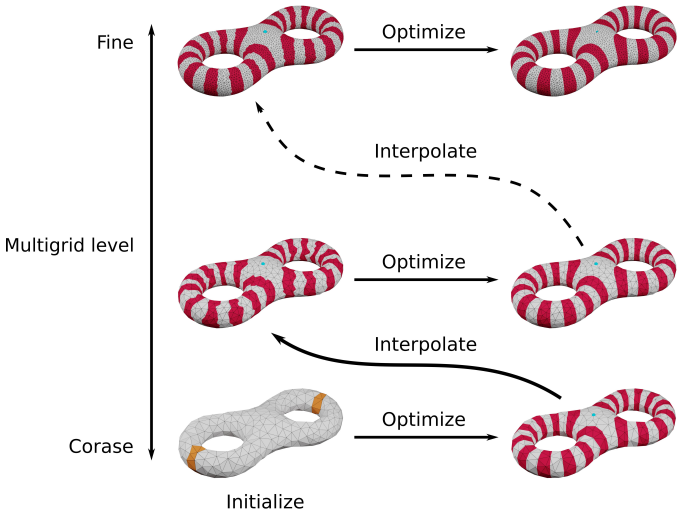


Fig. 4. The workflow of our method. First, we construct a multigrid hierarchy by simplifying the original mesh hierarchically. Then, we use triangle loops on the coarsest mesh to initialize the foliation. After that, we repeatedly run the algorithm [7, 4] in each level and interpolate the foliation of the coarser level to initialize the foliation of the next finer level, until we get the harmonic foliation on the original mesh.

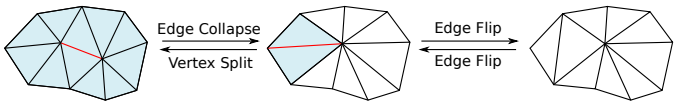
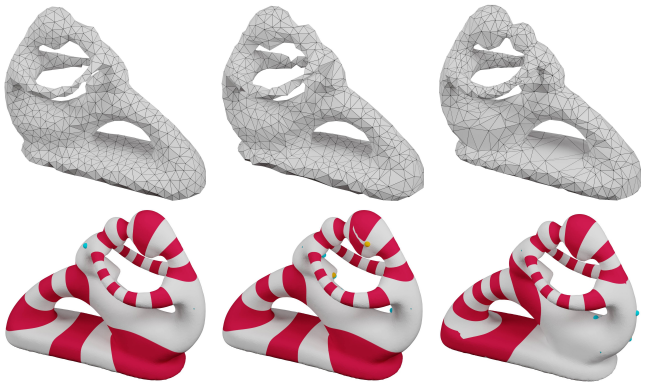


Fig. 5. Our multigrid hierarchy is built by a series of edge collapsing and edge flipping operations. The edge under operation is marked in red, and the region affected by the operation is marked in blue. These operations are recorded in the solver, and are rewound during interpolation to return from the coarse mesh to the original mesh.



(a) Midpoint (0) (b) Endpoint (3) (c) QSLIM (3)

Fig. 6. Different simplification strategies lead to different results. (a) Collapsing the shortest edge and place the new vertex at the midpoint of the edge. (b) Collapsing the shortest edge and place the new vertex at the endpoint of the edge. (c) Collapsing based on the QSLIM algorithm [35]. The numbers in the parentheses represent the numbers of additional poles generated, which should be avoided.

[9]. As discussed in the following, our interpolation scheme only changes the foliation values locally, so that the coarser grid solution would be close to the real solution in the finer grid, meaning the CMG scheme is suitable for our use.

The overall workflow of our method is illustrated in Fig. 4. First, we build a hierarchy of multi-resolution meshes. After that, we generate the foliation on the coarsest mesh by initialization in loops and optimization by the Whitehead moves of [7, 4]. With the generated foliation on the coarsest mesh, the CMG method is used to generate foliations on the finer levels until the harmonic foliation is obtained on the original mesh. Here, the foliation on a finer level is initialized by the foliation on the previous coarser level by interpolation, and optimized by Whitehead moves of [7, 4].

As mentioned in Section 1, there are two challenges we need to address. First, the algorithm [7, 4] only guarantees to converge on Delaunay meshes, but simplifying a mesh using edge collapsing may result in a non-Delaunay mesh. Thus, we need to build a hierarchy of Delaunay meshes (Section 4.1). Second, simple interpolation strategies like barycentric interpolation may break the closedness condition of the foliation and alter its Whitehead class. For this, we need to design specialized interpolation measures to maintain all the properties of the initial foliation (Section 4.2).

4.1. Constructing Delaunay multigrid hierarchy

Our multigrid hierarchy is constructed by alternating mesh simplification and Delaunay remeshing at each multigrid level, as demonstrated in Fig. 5. First, the finer grid is coarsened with some iterations of edge collapsing operations. Then, the simplified mesh is remeshed to be Delaunay using the edge flipping algorithm [12]. The sequence of collapsing and flipping operations are recorded, so that we can simply recover the original mesh by applying edge flipping and vertex splitting in the reverse order.

4.1.1. Edge collapsing

When doing mesh simplification, we need to choose the edge collapsing strategy, the number of levels, and the number of edges to reduce in each level.

The edge collapsing strategy refers to deciding which edge to collapse and where to place the remaining vertex after the collapse. The simplest way is by collapsing the shortest edge and placing the vertex at the midpoint or endpoint of the edge. Besides, there are also more sophisticated strategies that can preserve the shape of the surface better like the QSLIM method [35]. As shown in Fig. 6, the midpoint strategy tends to generate the most uniform meshes, while the other two strategies often create non-uniform meshes, which are more likely to cause unwanted poles. As a result, we use the midpoint strategy in this paper.

For the number of multigrid levels, it generally should not be set too big or too small, as shown in the example in Fig. 7. When creating too many levels, the mesh in the coarsest level will become too coarse. Although running the optimization on a coarser mesh is faster than on a finer mesh, it requires more time for interpolation and reconstructing the hierarchy, which would offset the time saved in optimization. On the other hand, if we only construct a few levels for a high-resolution mesh, then the mesh in the last level is still very large, and so requiring much more time for the optimization to converge. In general,

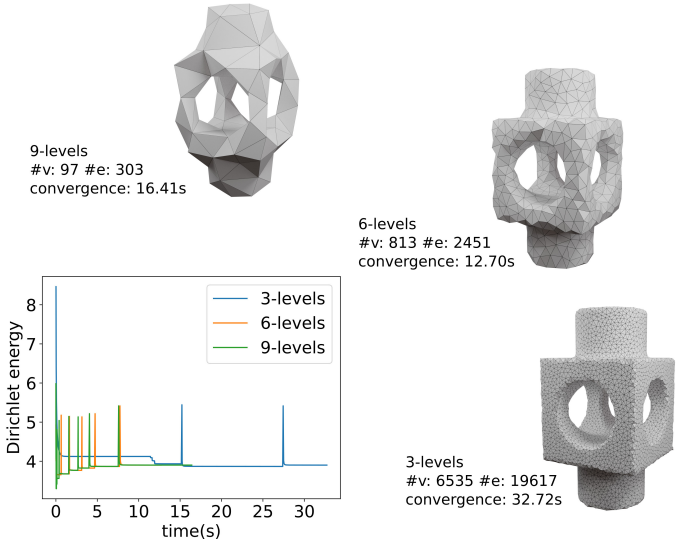
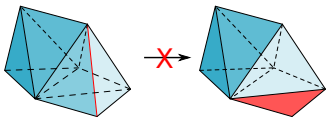


Fig. 7. Testing the performance of our method with different multigrid levels. In each case, we show the statistics of the mesh in the coarsest level.

the number of levels for achieving high efficiency is much related to the size of the mesh, so that we do not fix the number of levels in simplification. By our experiences, when the number of vertices of the coarsest mesh is roughly 1k (lowered down for very small meshes), the obtained number of levels is a good choice.

Lastly, it is also important to decide how many edges to collapse between neighboring levels in the multigrid hierarchy. As studied in previous works, the coarser mesh is often set to be proportional to the size in the previous level [9, 11], or constructed by removing the maximal independent set of the mesh [10]. In this paper, we also use a fixed ratio between two levels. By the experiments in Section 5, we set the number of edges in the coarser level to be 50% of the finer level, which tends to perform the best.

4.1.2. *Edge flipping*



We remesh the simplified mesh in each multigrid level to be Delaunay using the edge flipping algorithm proposed in [12], which greedily flips the edges that are not locally Delaunay (NLD). Unfortunately, as shown in the inset, some edges are topologically non-flippable since they will create overlapping faces. So strictly speaking, meshes in our multigrid hierarchy are nearly Delaunay and could contain a few NLD edges. Notice that Dyer et al. [12] also propose a geometry-preserving Delaunay remeshing algorithm which can handle the topologically non-flippable cases. However, as this algorithm involves vertex splitting and will potentially add a lot of vertices to the mesh, it is not suitable for coarsening. Fortunately, as we will discuss in Section 5, the optimization may also converge even there are some NLD edges (Table 2).

4.2. *Whitehead-class-preserving interpolation*

For a multigrid solver, the solution it produces should converge to the real solution computed directly on the original mesh. As introduced in Section 3, this means that as long as the input loops to the multigrid solver L and input loops to the original mesh L' have the same homotopy types as well as the same associated foliation values, then the initial foliations should converge to the same harmonic measured foliation in the same Whitehead class.

Accordingly, when interpolating the foliation on the coarser mesh to the finer mesh, we need to ensure that the Whitehead class of the foliation is not changed. As discussed in [7], two measured foliations are Whitehead equivalent if the path integrals along every class of homotopic curves on the surface are the same. This is equivalent to requiring that no poles are generated on the locally modified vertices and faces, as shown in Fig. 5. Since a foliation without poles is closed, the path integrals for homotopic curves are the same no matter whether the curve crosses the modified region or not due to the closedness. As a result, as long as the flipping and collapsing do not create poles, they will not affect the path integrals along any curve, and so the Whitehead class of the foliation is preserved. In the following, we will discuss how to interpolate the foliation when reverting the edge flipping and collapsing without poles introduced.

4.2.1. *Whitehead perturbation*

As discussed in Section 3, edges with zero foliation values will affect the corner closedness as well as the indices of vertices and faces. To simplify the discussion, we first perturb the foliation to eliminate such edges before interpolation.

Specifically, for an edge e_{ij} whose foliation value $F_{ij} = 0$, we conduct a Whitehead move at v_i or v_j, as shown in Fig. 8. Here, a small value ϵ is subtracted from an arbitrary sector S (red sector in this example) not containing e_{ij}, and added to the other edges. For preserving the properties of the foliation, the value ϵ should satisfy the following two conditions. First, when subtracting ϵ, the foliation values should remain positive, so $\epsilon < F_{ij}$ for $e_{ij} \in S$. Second, when adding ϵ to the other edges, we need to ensure that no pole faces are created (cf. Eq. (2)), so $\epsilon < (F_{ij} + F_{ki} - F_{jk})/2$ for $e_{ij}, e_{jk}, e_{ki} \in t_{ijk}$ and $e_{ij}, e_{ki} \in \mathcal{E} \setminus S$. As a result, we simply set ϵ to be half of the upper bound as computed above in our experiments.

Note that when v_i or v_j is a zero of the foliation, the Whitehead perturbation can move the zero from the vertex to an incident triangle, which does not affect the following computation.

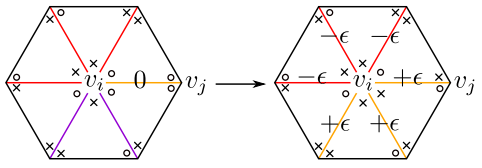


Fig. 8. Locally perturb the foliation by conducting a discrete Whitehead move around the vertex to eliminate edges whose values are zeros.

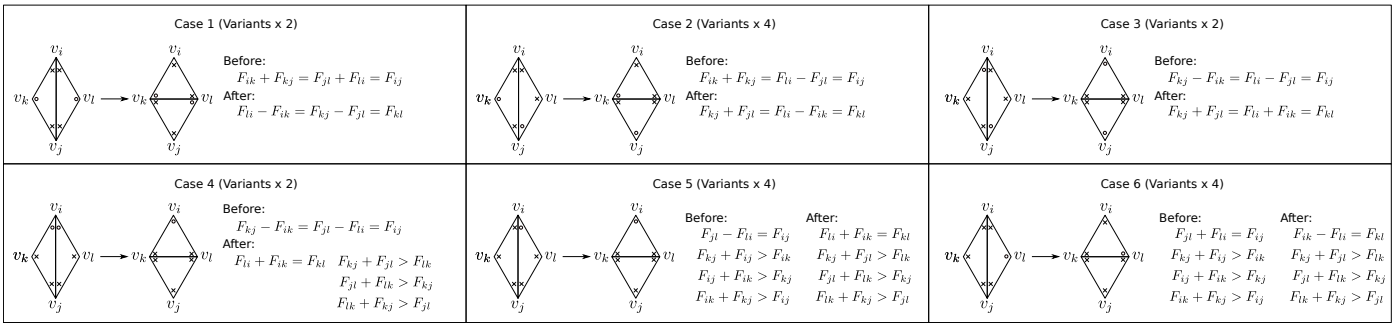


Fig. 9. The interpolation measures to revert an edge flipping when switching to the finer level in the multigrid hierarchy. They are classified into six cases based on the closedness of the corners in the two related triangles. Please refer to the text for details.

4.2.2. Interpolation for edge flipping

When reverting an edge flip, we only need to determine the foliation value for the flipped edge, e.g. F_{kl} for e_{kl} in Fig. 9. The key idea for setting F_{kl} is to make the number of closed corners around all vertices $O(v)$ and non-closed corners around all faces $X(t)$ unchanged. Then according to Eqs. (1) to (3), the indices for all the vertices and faces will remain the same and no poles are introduced.

The specific interpolation measures are classified into six cases based on the configuration of the closedness on the corners. There are several variants for each case due to symmetry, and we only demonstrate one variant for each case in Fig. 9, and discuss the other variants in the text. Case 1-4 describe the situations where the two incident triangles are regular, while case 5-6 describe the situations where one incident triangle is regular and the other is zero. As for the case where both incident triangles are zeros, we conduct a Whitehead perturbation on v_k or v_l to move one of the zero from the face to the vertex, making it into case 5 or 6. Now, we discuss these cases respectively.

Case 1. For regular triangles, one of its three corners is closed, and the other two are not. In this case, the two opposite corners of e_{ij} are closed. After flipping, F_{kl} is set as in Fig. 9 such that the corners $\angle ikl, \angle jlk$ are closed, so that $O(v)$ and $X(t)$ are not changed, which ensures no poles are created as mentioned above. There is another variant for this case when the closed corners after flipping are $\angle kli, \angle lkj$, then $F_{kl} = F_{ik} - F_{li} = F_{jl} - F_{kj}$. Here, which variant to be used depends on the relative values of F_{li}, F_{ik} and F_{kj}, F_{jl}.

Case 2. In this case, one closed corner is opposite to e_{ij}, and the other closed corner is adjacent to e_{ij}. There are in total four variants for this case considering symmetry, namely (1) $\angle ikj, \angle ijl$; (2) $\angle ikj, \angle lij$; (3) $\angle jli, \angle kji$; (4) $\angle jli, \angle jik$. For variant (1) and (3) we have $F_{kl} = F_{kj} + F_{jl}$ as shown in Fig. 9, while for (2) and (4) we have $F_{kl} = F_{li} + F_{ik}$.

Case 3. In this case, the two closed corners are both adjacent to e_{ij}, but not incident to the same vertex. So they can either be (1) $\angle jik, \angle ijl$; or (2) $\angle kji, \angle lij$. Here F_{kl} for both cases are the same as listed in Fig. 9, since they result in the same configuration after the flip.

Case 4. In this case, the two closed corners are both adjacent to e_{ij} and incident to the same vertex. As we can see, there is no way to keep $O(v)$ and $X(t)$ unchanged after the flipping, so the indices for the vertices and faces have to change. For the variant shown in Fig. 9, we set $F_{kl} = F_{li} + F_{ik}$. We can easily verify that t_{kjl} satisfies the triangle inequalities (Eq. (2)) as follows,

Proof.

$$\begin{aligned} F_{kj} + F_{jl} &= F_{kj} + F_{ij} + F_{li} \\ &= F_{kj} + F_{ij} + F_{kl} - F_{ik} \\ &= 2F_{kj} + F_{kl} \\ &> F_{kl} \end{aligned}$$

$$\begin{aligned} F_{jl} + F_{lk} &= F_{jl} + F_{li} + F_{ik} \\ &= F_{li} + F_{ij} + F_{li} + F_{ik} \\ &= 2F_{li} + F_{kj} \\ &> F_{kj} \end{aligned}$$

$$\begin{aligned} F_{lk} + F_{kj} &= F_{li} + F_{ik} + F_{ij} + F_{ik} \\ &= 2F_{ik} + F_{jl} \\ &> F_{jl} \end{aligned}$$

□

As a result, t_{ijk} becomes a zero face. Meanwhile, $O(v_i)$ is reduced by 1, which in turn causes the index of v_i to increase by 1 (Eq. (3)). Thus, if v_i is a zero before flipping, it will become a zero with a smaller index or a regular vertex after flipping, which is acceptable. But when v_i is regular, it will become a pole after flipping, which could be problematic. However, this situation rarely happens in practice, because closed corners tends to distribute uniformly around a vertex. If it does happen, v_i is often not coclosed because one sector of v_i contains much more edges than its other sectors (Eq. (4)). So this issue can often be fixed by running a few iterations of the optimization, by which the corner structure changes to another case that is acceptable. Unfortunately, this is not guaranteed, and the interpolation may create a pair of pole and zero locally. More experimental results for this case can be found in Section 5 (Table 2).

Case 5. In this case, one of the triangle is zero, and the closed corner in the regular triangle is adjacent to the flipping edge e_{ij}. For the variant shown in Fig. 9, the zero face moves from t_{ikj} to t_{kjl}, which can be verified similar to case 4. With regard to the indices of the other vertices and faces, they remain the same. In this case, there are four variants based on which corner adjacent

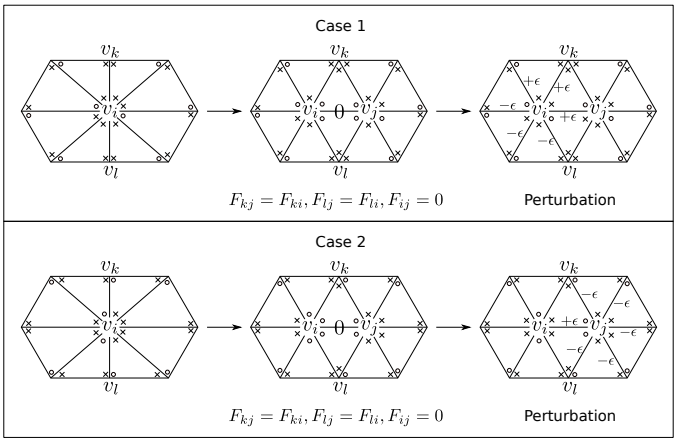


Fig. 10. The interpolation measures to revert an edge collapsing when switching to the finer level in the multigrid hierarchy. They are classified into two cases based on the closedness of the corners in the related triangles. Please refer to the text for details.

to e_{ij} is closed, where F_{ij} can be naturally deduced from the closed corner.

Case 6. This case is similar to case 5, except that the closed corner is opposite to e_{ij}. There are also four variants to this case based on which corner adjacent to e_{kl} is closed after the flipping, where F_{ij} can be naturally deduced from the closed corner.

4.2.3. Interpolation for edge collapsing

A vertex splitting is the opposite operation to an edge collapsing. As demonstrated in Fig. 10, after splitting the vertex v_i, one vertex (v_j), three edges (e_{ij}, e_{jk}, e_{lj}), and two faces (t_{ijk}, t_{ilj}) are created. We need to set the foliation values for the three edges such that no poles are created. Here, we let $F_{ij} = 0, F_{jk} = F_{ki}, F_{lj} = F_{il}$. Then we do a Whitehead perturbation, as introduced in Section 4.2.1, to eliminate the zero foliation value for F_{ij}. Based on the configuration of the closed corners, there are two cases, as detailed below.

Case 1. In this case, both sides (identified as ccw circulation around v_i from v_l to v_k and then from v_k to v_l) contain at least one closed corner before the splitting. Thus, neither v_i nor v_j will become a pole after the splitting and interpolation. Then, we can arbitrarily select v_i or v_j to conduct a Whitehead perturbation to make $F_{ij} > 0$, as shown in Fig. 10(a).

Case 2. In this case, one side (identified as ccw circulation around v_i from v_l to v_k and then from v_k to v_l) does not contain any closed corner before the splitting. As a result, the index for v_i and v_j will change after the splitting, by which one vertex may be turned into a zero and the other into a pole. Here, we conduct a Whitehead perturbation on the pole vertex so that the pole and zero can cancel each other out, as shown in Fig. 10(b).

5. Experiments and analysis

We evaluated our algorithm in various aspects, and compared it to the original algorithm [7, 4]. In this paper, we implemented all the algorithms in C++17, and compiled using the gcc-11 compiler with the -O3 flag. All the experiments were conducted on a personal computer running the ArchLinux system with AMD 3950X 3.5GHz CPU and 64GB RAM, and the convergence threshold is set to be 10^{-6}.

Comparison with the original method [7, 4]. We tested the performance of our method on different meshes of different resolutions. The statistics are summarized in Table 1, and the resulting harmonic foliations are illustrated in Fig. 11. In this test, we use handle loops generated with [34] and set all associated weights uniformly to 1. For our method, we varied the simplification ratio for edges between consecutive levels in the hierarchy to be 25%, 50%, and 75% respectively.

As shown in Table 1, our multigrid method greatly improves the efficiency in all cases. The larger the mesh becomes, the more speed-up our method can achieve. This is because on large meshes, the input loops will only cover a very small portion of the mesh, so that only a few edges are armed with the initial values and the rest are all zeros. Thus, it will take a very long time to optimize this foliation to be harmonic. As listed in Table 1, the original method even failed to converge on some large meshes. On the other hand, since the coarsest mesh in our multigrid hierarchy is always small enough, so that the initialized foliation can quickly be optimized to be harmonic with the algorithm of [7, 4]. As the foliation on a coarser level provides a much better initialization for generating the foliation on the next finer level, providing foliation values for nearly all edges of the mesh for the finer level, our solver can converge much faster.

Using our method, the time used for building the multigrid hierarchy is very little compared to the time for solving the foliation. Meanwhile, our multigrid hierarchy can be reused with different input loops once it is constructed. Since the placement of the singularities for the foliation is fully determined by the optimization, one has to adjust the input loops as well as its associated weights and re-run the algorithm to see if the result meets one's requirements. Thus, using our method, one can quickly test the inputs on the coarser levels to see if the initialized foliation is desirable, which can save much time on adjustment. As for using the original algorithm of [7, 4], it would be very difficult for such interactive adjustments as it is very time-consuming.

Performance analysis. In Fig. 12, we measure the time spent for three major aspects of our algorithm, namely building the multigrid hierarchy, performing the Whitehead move optimization algorithm of [7, 4], and interpolating foliations between neighboring levels. From the statistics, the building phase takes up around 1%-7% of the total running time, since the collapsing and flipping are local operations and fast to compute. The interpolation of all levels takes up relatively the same amount of time as the building phase, since the number of edges to process is identical. The optimization by Whitehead moves of all levels consumes over 90% of the total running time.

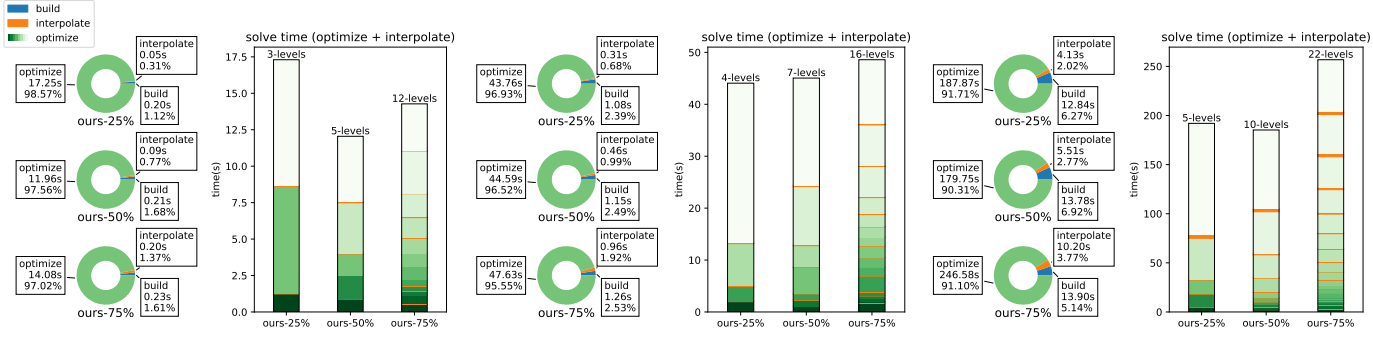
As discussed in Section 4, the number of edges to collapse between consecutive hierarchical levels has much influence on the efficiency. Here, we made a test by setting the simplification ratio for edges being 25%, 50%, and 75% to check their convergence efficiencies for some meshes. In Table 1, the statistics are listed, from which we know that 50% is generally the best.

Table 1. Time cost[1] of our multigrid solver compared to the original algorithm [7, 4].

mesh	#v	#e	original	ours-25%[2]		ours-50%		ours-75%	
				build[3]	solve[4]	build	solve	build	solve
block	26k	78k	103.55	0.19	17.30	0.21	**12.05 (9x)**	0.23	14.28
bob	5k	15k	15.47	0.03	4.12	0.03	**4.00 (4x)**	0.03	13.01
botijo	93k	280k	—[5]	1.08	35.49	1.11	**34.79**	1.27	36.73
bumpy-torus	12k	36k	73.56	0.08	7.67	0.09	**4.38 (17x)**	0.09	29.23
cad	74k	222k	3372.10	0.73	29.43	0.80	**28.87 (117x)**	0.84	32.11
dtorus	66k	197k	—	0.62	19.65	0.63	**15.82**	0.81	24.93
eight	100k	301k	—	1.25	26.06	1.63	**26.86**	1.35	32.54
fertility	30k	90k	940.53	0.23	25.28	0.24	25.24	0.28	**22.13 (43x)**
genus	104k	312k	—	1.16	51.82	1.22	**43.99**	1.43	60.13
holes3	96k	288k	2985.67	1.08	**44.38 (67x)**	1.15	45.05	1.26	48.58
joint	36k	108k	2024.21	0.27	23.34	0.29	**15.38 (132x)**	0.33	18.11
kitten	50k	150k	1610.22	0.41	23.65	0.56	**13.38 (120x)**	0.47	16.24
master-cylinder	542k	1625k	—	12.84	192.00	13.78	**185.25**	13.90	256.78
pegasus	20k	60k	395.29	0.13	**19.62 (x20)**	0.14	20.89	0.16	21.10
rocker-arm	1k	3k	1.06	0.00	1.00	0.00	1.02	0.00	**0.83 (1x)**
sculpt	58k	175k	570.58	0.48	**25.27 (23x)**	0.55	25.29	0.61	26.82
torus	3k	9k	7.83	0.02	3.43	0.03	2.61	0.03	**1.58 (5x)**
trefoil-knot	147k	442k	—	2.31	34.47	2.36	**23.36**	2.73	28.71
trimstar	246k	737k	—	4.30	67.20	4.58	**53.43**	4.99	61.62
tube	42k	126k	3216.15	0.33	43.86	0.35	15.90	0.41	**13.15 (245x)**

[1] Time measured in seconds. [2] (edges in coarser level) / (edges in finer level). [3] Time to build the multigrid hierarchy.
[4] Time to compute the foliation. [5] Not converged within a time limit (1h in all experiments).

Fig. 11. The harmonic measured foliations generated for meshes listed in Table 1 (showing the results of ours-50%).



(a) block (26k) (b) holes3 (96k) (c) master-cylinder (542k)

Fig. 12. Time cost for different aspects of our algorithm for three examples chosen from Table 1, including the time for building the multigrid hierarchy (build), optimizing the foliation using the Whitehead move algorithm (optimize), and interpolating foliation between multigrid levels (interpolate). Here, the sections of a bar represent the time cost for handling the levels of the hierarchy for the mesh, and they are arranged from bottom to top, corresponding to handling the levels from the coarsest to the finest.

Fig. 13. Our algorithm achieves similar converged results on the same shape in three very different resolutions.

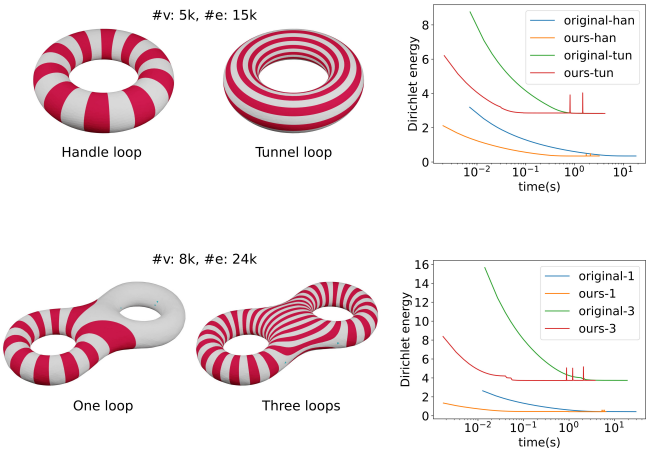


Fig. 14. Our algorithm converges faster than the original method [7, 4] no matter what types and numbers of input loops are given.

As a result, we choose 50% in all the other experiments.

With regard to the influence of our interpolation on the convergence efficiency, we notice from the time-energy diagrams shown in Figs. 1, 7 and 14 that there are some spikes in the curves of our method. These spikes are caused by our interpolation because it only preserves the closedness of the foliation, but breaks the coclosedness. Fortunately, these spikes can be quickly smoothed in a just few iterations, and do not incur too much overhead on our algorithm. This is because the interpolation only alters the foliation locally. Therefore, the interpolated foliation is not very far from the harmonic solution, as seen in Fig. 4 where the initial foliation in each level (left) is similar to the harmonic one (right).

Testing on the same shape in different resolutions. As seen in Table 1, the running time of the algorithm does not solely depend on the resolution of the mesh. In general, the geometry of the surface has much influence on the time cost, e.g., the algorithm tends to converge more slowly on a complex, curvy surface than on a simple, smooth surface. For example, the "pegasus" mesh is smaller than the "block" mesh, but the algorithm runs more slowly on the first mesh for both the original method and our multigrid method.

To see the performance of our method independent of the shape geometry, we tested the algorithm on several meshes in different resolutions representing the same surface in Fig. 13. The results show that our multigrid solver always converges to similar solutions, independent of the mesh resolutions.

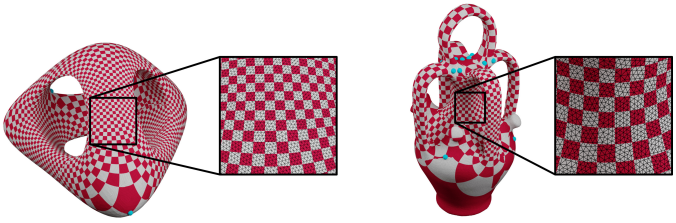


Fig. 15. With our multigrid solver, we can compute global conformal parameterizations on very large meshes, which cannot be achieved by [4].

Fig. 16. The foliations with poles collected from Table 2. Left: pegasus; Right: master-cylinder. Poles are shown in yellow while zeros are shown in blue.

Testing different input loops. As discussed in [7], changing the homotopy types of the input loops will change the Whitehead class of the foliation, as well as the convergence time of the algorithm. Nevertheless, our multigrid method can accelerate the convergence of the optimization in face of different input loops, as illustrated in Fig. 14.

Application: scalable conformal parameterization on high-genus meshes. Zhao et al. [4] propose to compute a conformal parameterization on high-genus meshes using the harmonic measured foliation. Due to the inefficiency of the original harmonic measured foliation algorithm, Zhao et al. [4] only report results on meshes less than 30,000 vertices. But armed with our multigrid solver, we can compute global conformal parameterization using [4] on extremely large meshes, e.g. over one million edges, as shown in Fig. 15.

Limitations. As mentioned in Section 4, our algorithm may create poles due to 1) non-Delaunay edges that are not flippable when constructing the mesh hierarchy (NLD), and 2) a rare circumstance in case 4 during the interpolation for edge flipping (BF). Fortunately, the NLD cases are very rare and only happens on complex shapes, as reported in Table 2. Even if the NLD edges appear, the foliation algorithm may also be free of poles in practice, e.g., the "pegasus" mesh using ours-25% method. The BF cases are also scarce compared to the mesh size, and many of them can be fixed by running a few more iterations of the Whitehead move algorithm as stated in Section 4.2. In our current implementation, we set a time limit of 1s for the fixing procedure to prevent it from getting stuck, and it can potentially reduce more BF cases if it is allowed to run longer on some complex shapes. Also note that for the "genus" mesh using ours-25% method, the BF case is not fixed, but no poles are generated because the pole and zero cancel each other out in the later iterations. Even so, our algorithm may produce poles as shown in Fig. 16. This requires further studies for solving the problem.

6. Conclusion and future work

Table 2. Corner cases encountered for the experiments in Table 1.

mesh	ours-25%			ours-50%			ours-75%		
	NLD[1]	BF[2]	P[3]	NLD	BF	P	NLD	BF	P
block	0	0/0	0	0	0/0	0	0	0/0	0
bob	0	0/0	0	0	0/0	0	0	0/0	0
botijo	0	0/0	0	0	2/0	0	0	1/0	0
bumpy-torus	0	0/0	0	0	0/0	0	0	0/0	0
cad	0	0/0	0	0	0/0	0	0	0/0	0
dtorus	0	0/0	0	0	0/0	0	0	0/0	0
eight	0	0/0	0	0	1/0	0	0	0/0	0
fertility	0	0/0	0	0	0/0	0	0	1/0	0
genus	0	1/1	0	0	0/0	0	0	0/0	0
holes3	0	0/0	0	0	0/0	0	0	1/0	0
joint	0	0/0	0	0	0/0	0	0	0/0	0
kitten	0	0/0	0	0	1/0	0	0	0/0	0
master-cylinder	0	0/0	0	0	5/2	6	0	1/0	0
pegasus	3	0/0	0	3	2/2	3	7	2/2	2
rocker-arm	0	0/0	0	0	0/0	0	0	0/0	0
sculpt	0	0/0	0	0	1/0	0	0	0/0	0
torus	0	0/0	0	0	0/0	0	0	0/0	0
trefoil-knot	0	0/0	0	0	0/0	0	0	1/0	0
trimstar	0	1/0	0	0	0/0	0	0	0/0	0
tube	0	1/0	0	0	1/0	0	0	0/0	0

[1] Non-locally Delaunay edges.
[2] Bad flips causing poles before/after fixing.
[3] Poles in the final result.
[4] Data in this table represents the number of occurrences of corner cases. In general, the probabilities of NLD, BF, and P are always very small in handling a mesh. For example, for the pegasus mesh using ours-50%, the probabilities of NLD, BF, and P are approximately 0.05‰, 0.03‰, and 0.05‰respectively against the edges of the mesh.

In this paper, we present a multigrid method for generating discrete harmonic measured foliations [7, 4], which involves non-linear optimization with hard constraints and is known hard to solve. The core of our multigrid solver consists of a tailored multi-resolution mesh hierarchy and a set of constraints-preserving interpolation measures to help the algorithm converge to the real solution. Experimental results show that our method achieves an acceleration rate up to 200+ times compared to the original algorithm, and can generate a solution for a mesh with over one million edges in just a few minutes. With our multigrid solver, the harmonic measured foliation algorithm can scale well to very large meshes.

There are many directions to explore in the future. First, it is interesting to see whether more sophisticated schemes, like the Full Approximation Scheme (FAS) [19], can help obtain much more acceleration. Second, since the discrete measured foliation is similar to a differential 1-form, it is interesting to extend our multigrid solver to solve the other problems formulated using 1-forms [8], which are more widely adopted in the literature.

Acknowledgements

The meshes used in our experiments are collected from the dataset provided by [36]. The work is partially supported by the National Natural Science Foundation of China under Grant 62072446.

References

[1] Campen, M, Silva, CT, Zorin, D. Bijective maps from simplicial foliations. ACM Transactions on Graphics 2016;35(4):1–15. doi:10.1145/2897824.2925890.
[2] Cohen, D, Ben-Chen, M. Generalized volumetric foliation from inverted viscous flow. Computers & Graphics 2019;82:152–162. doi:10.1016/j.cag.2019.05.015.
[3] Vekhter, J, Zhuo, J, Fandino, LFG, Huang, Q, Vouga, E. Weaving geodesic foliations. ACM Transactions on Graphics 2019;38(4):1–22.
[4] Zhao, H, Wang, S, Wang, W. Global conformal parameterization via an implementation of holomorphic quadratic differentials. IEEE Transactions on Visualization and Computer Graphics, early access 2020;doi:10.1109/TVCG.2020.3016574.
[5] Lei, N, Zheng, X, Si, H, Luo, Z, Gu, X. Generalized regular quadrilateral mesh generation based on surface foliation. Procedia Engineering 2017;203:336–348. doi:10.1016/j.proeng.2017.09.818.
[6] Zheng, X, Wen, C, Lei, N, Ma, M, Gu, X. Surface registration via foliation. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). 2017, p. 938–947. doi:10.1109/ICCV.2017.107.
[7] Palmer, DR. Toward computing extremal quasiconformal maps via discrete harmonic measured foliations. Bachelor's thesis; Harvard College; 2016.
[8] Crane, K. Discrete differential geometry: An applied introduction. Notices of the AMS, Communication 2018;:1153–1159.
[9] Ray, N, Levy, B. Hierarchical least squares conformal map. In: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications. Canmore, Alta., Canada: IEEE Comput. Soc. ISBN 978-0-7695-2028-5; 2003, p. 263–270. doi:10.1109/PCCGA.2003.1238268.
[10] Aksoylu, B, Khodakovsky, A, Schröder, P. Multilevel solvers for unstructured surface meshes. SIAM Journal on Scientific Computing 2005;26(4):1146–1165. doi:10.1137/S1064827503430138.
[11] Liu, HTD, Zhang, JE, Ben-Chen, M, Jacobson, A. Surface multigrid via intrinsic prolongation. ACM Transactions on Graphics 2021;40(4):1–13. doi:10.1145/3450626.3459768.
[12] Dyer, R, Zhang, H, Möller, T. Delaunay mesh construction. In: Proceedings of the Fifth Eurographics Symposium on Geometry Processing. SGP '07; Aire-la-Ville, Switzerland, Switzerland: Eurographics Association. ISBN 978-3-905673-46-3; 2007, p. 273–282.
[13] Fathi, A, Laudenbach, F, Poénaru, V. Thurston's Work on Surfaces (MN-48). Princeton University Press; 2012. ISBN 978-0-691-14735-2.
[14] Strebel, K. Quadratic Differentials. Ergebnisse Der Mathematik Und Ihrer Grenzgebiete. 3. Folge / A Series of Modern Surveys in Mathematics; Berlin Heidelberg: Springer-Verlag; 1984. ISBN 978-3-540-13035-2. doi:10.1007/978-3-662-02414-0.
[15] Lei, N, Zheng, X, Jiang, J, Lin, YY, Gu, DX. Quadrilateral and hexahedral mesh generation based on surface foliation theory. Computer Methods in Applied Mechanics and Engineering 2017;316:758–781. doi:10.1016/j.cma.2016.09.044.
[16] Lei, N, Zheng, X, Jiang, J, Lin, YY, Gu, DX. Quadrilateral and hexahedral mesh generation based on surface foliation theory II. Computer Methods in Applied Mechanics and Engineering 2017;321:406–426. doi:10.1016/j.cma.2017.04.012.
[17] Shi, R, Zeng, W, Su, Z, Damasio, H, Lu, Z, Wang, Y, et al. Hyperbolic harmonic mapping for constrained brain surface registration. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013, p. 2531–2538. doi:10.1109/CVPR.2013.327.
[18] Gu, X, Yau, ST. Global conformal surface parameterization. In: Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing. SGP '03; Goslar, DEU: Eurographics Association. ISBN 978-1-58113-687-6; 2003, p. 127–137.

[19] Briggs, WL, Henson, VE, McCormick, SF. A Multigrid Tutorial. Other Titles in Applied Mathematics; Society for Industrial and Applied Mathematics; 2000. ISBN 978-0-89871-462-3. doi:10.1137/1.9780898719505.

[20] Kazhdan, M, Hoppe, H. Streaming multigrid for gradient-domain operations on large images. ACM Transactions on Graphics 2008;27(3):1–10.

[21] McAdams, A, Sifakis, E, Teran, J. A parallel multigrid poisson solver for fluids simulation on large grids. In: Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. 2010, p. 65–73.

[22] Aanjaneya, M, Han, C, Goldade, R, Batty, C. An efficient geometric multigrid solver for viscous liquids. Proceedings of the ACM on Computer Graphics and Interactive Techniques 2019;2(2):1–21.

[23] Kazhdan, M, Bolitho, M, Hoppe, H. Poisson surface reconstruction. In: Proceedings of the Fourth Eurographics Symposium on Geometry Processing. SGP '06; Cagliari, Sardinia, Italy: Eurographics Association. ISBN 978-3-905673-36-4; 2006, p. 61–70.

[24] Kazhdan, M, Hoppe, H. Screened poisson surface reconstruction. ACM Transactions on Graphics 2013;32(3):1–13.

[25] Kazhdan, M, Hoppe, H. An adaptive multigrid solver for applications in computer graphics. Computer Graphics Forum 2019;38(1):138–150. doi:10.1111/cgf.13449.

[26] Wang, Z, Wu, L, Fratarcangeli, M, Tang, M, Wang, H. Parallel multigrid for nonlinear cloth simulation. Computer Graphics Forum 2018;37(7):131–141. doi:10.1111/cgf.13554.

[27] Otaduy, MA, Germann, D, Redon, S, Gross, M. Adaptive deformations with fast tight bounds. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. 2007, p. 181–190.

[28] Botsch, M, Kobbelt, L. A remeshing approach to multiresolution modeling. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing. ACM; 2004, p. 185–192.

[29] Ni, X, Garland, M, Hart, JC. Fair Morse functions for extracting the topological structure of a surface mesh. ACM Transactions on Graphics 2004;23(3):613–622. doi:10.1145/1015706.1015769.

[30] Shi, L, Yu, Y, Bell, N, Feng, WW. A fast multigrid algorithm for mesh deformation. ACM Transactions on Graphics 2006;25(3):1108–1117.

[31] Tamstorf, R, Jones, T, McCormick, SF. Smoothed aggregation multigrid for cloth simulation. ACM Transactions on Graphics 2015;34(6):1–13.

[32] Meyer, M, Desbrun, M, Schröder, P, Barr, AH. Discrete differential-geometry operators for triangulated 2-manifolds. In: Visualization and Mathematics III. Springer; 2003, p. 35–57.

[33] Erickson, J, Whittlesey, K. Greedy optimal homotopy and homology generators. In: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms. SODA '05; USA: Society for Industrial and Applied Mathematics. ISBN 978-0-89871-585-9; 2005, p. 1038–1046.

[34] Dey, TK, Fan, F, Wang, Y. An efficient computation of handle and tunnel loops via Reeb graphs. ACM Transactions on Graphics 2013;32(4):1–10. doi:10.1145/2461912.2462017.

[35] Garland, M, Heckbert, PS. Surface simplification using quadric error metrics. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co.; 1997, p. 209–216.

[36] Myles, A, Pietroni, N, Zorin, D. Robust field-aligned global parametrization. ACM Transactions on Graphics 2014;33(4):1–14. doi:10.1145/2601097.2601154.