Dihedral Angle-based Maps of Tetrahedral Meshes

Gilles-Philippe Paillé[1] Nicolas Ray[2] Pierre Poulin[1] Alla Sheffer[3] Bruno Lévy[2]
[1]Université de Montréal [2]INRIA Nancy - Grand Est [3]University of British Columbia

Abstract

We present a geometric representation of a tetrahedral mesh that is solely based on dihedral angles. We first show that the shape of a tetrahedral mesh is completely defined by its dihedral angles. This proof leads to a set of angular constraints that must be satisfied for an immersion to exist in $\mathbb{R}^3$. This formulation lets us easily specify conditions to avoid inverted tetrahedra and multiply-covered vertices, thus leading to locally injective maps. We then present a constrained optimization method that modifies input angles when they do not satisfy constraints. Additionally, we develop a fast spectral reconstruction method to robustly recover positions from dihedral angles. We demonstrate the applicability of our representation with examples of volume parameterization, shape interpolation, mesh optimization, connectivity shapes, and mesh compression.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

Keywords: Volume parameterization, interpolation, compression, optimization

1 Introduction

Geometry represention plays an important role in how we think about a problem and how we solve it. A careful choice of variables can simplify the expression of a solution and lead to a straightforward implementation. Such changes of variables have significantly improved efficiency and robustness of surface maps, such as methods based on angles [Sheffer and de Sturler 2001], on metric [Springborn et al. 2008], and on curvature [Crane et al. 2011; Crane et al. 2013]. Computer animation has also benefited from represention such as barycentric coordinates [Ju et al. 2005] and modal analysis [Pentland and Williams 1989] to reduce computational time. Based on these experiences for surfaces, it is only natural to look for similar approaches for volumes. While some deformation techniques using barycentric coordinates and modal analysis trivially generalize to volumes, the body of work in this field remains limited.

In this paper, we study *dihedral angles*, a geometric quantity well-known to the volume mesh processing community. A dihedral angle is defined as the angle between two planes; each edge of a tetrahedron has a dihedral angle determined by its two incident triangles. It is commonly used to assess the quality of tetrahedral meshes [Labelle and Shewchuk 2007], and therefore, it seems natural to express an objective function that defines the quality of a mesh directly in terms of these angles, and to find a numerical solution mechanism to optimize them. We take a step in this direction by showing that the shape of a tetrahedral mesh is completely specified by its dihedral angles, up to a global rotation and isotropic scale. Basically, this implies that manipulating these angles indeed changes the geometry of the mesh. Our goal is to study the properties and the structure of this representation, and to gain some understanding about the intricate relations between vertex positions and dihedral angles.

Our first contribution is a proof that this change of variables is equivalent to an immersion of a mesh in $\mathbb{R}^3$, where an immersion is a locally injective simplicial map [Cervone 1996], which guarantees that there are no inverted tetrahedra and no multiply-covered vertices. This proof leads to the construction of a set of variables along with a complete set of constraints required for the representation to be an immersion. Our second contribution is a numerical algorithm that optimizes arbitrary input dihedral angles in such a way that they satisfy the constraints that we have exhibited. We call this process *flattening* as a parallel to surface metric flattening techniques. Our third contribution is a robust spectral reconstruction that recovers vertex positions from dihedral angles.

Related Work Our work is best interpreted as a volume equivalent of a series of angle-based surface parameterization techniques that were popularized over the last decades. The first angle-based technique was conceived by Di Battista and Vismara [1993] to draw graphs on 2D planes. This method has then been introduced to the computer graphics community for conformal parameterization of surfaces [Sheffer and de Sturler 2001]. A considerable amount of work has been done toward improving its efficiency, including by Sheffer et al. [2005] who exploit the structure of constraints to significantly decrease solving time. Convergence rate has also been improved by reformulating nonlinear constraints [Zayer et al. 2005] and linearizing the problem [Zayer et al. 2007]. More relevant related work is given along its respective applications.

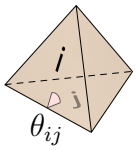
The success of these methods inspired us to search for an equivalent representation for tetrahedral meshes, which lead us to dihedral angles. It is known that the dihedral angles of a single tetrahedron determine its shape [Luo 1997]. We generalize this observation to tetrahedral meshes and to build a complete framework to work with dihedral angles. While this generalization is nontrivial, we show that the same flattening-reconstruction workflow can be applied in a simple and intuitive way.

Overview Our goal is to prove that dihedral angles alone determine the shape of a tetrahedral mesh, and to show how to recover vertex positions from the sole dihedral angles (plus a global similarity transformation). We first proceed to show that the vertex positions of a tetrahedral mesh can be uniquely reconstructed from the sole dihedral angles, up to a global similarity transformation (Section 2). To prove this, we first demonstrate equivalence for a simple elementary mesh. Then, we consider a general mesh as a sequence of elementary combinatorial operations applied to a simple one, and prove the property by structural induction. In some application contexts, it may be desired that the angles match some prescribed values that do not necessarily satisfy the constraints. For these applications, we propose a constrained optimization problem that looks for the nearest set of angles that satisfy all constraints. We solve this problem using a nonlinear least-squares method (Section 3). Finally, we develop a fast spectral reconstruction method to robustly recover

positions from dihedral angles (Section 4). We show some applications (Section 5) expressed with dihedral angles that range from volume parameterization, shape interpolation, mesh optimization, connectivity shapes, and mesh compression.

2 Dihedral Angle Variables

We now prove that dihedral angles completely determine the shape of a tetrahedral mesh, and exhibit the set of constraints that these dihedral angles need to satisfy. We suppose that the mesh is a topological ball. We can think of this change of representation as a change of variables. The key is to ensure that our new variables have the same representation power as position variables.



Our variables are dihedral angles of all tetrahedra. Each tetrahedron has six dihedral angles $\theta_{ij} \in [0, \pi]$, i.e., one for each pair of facets (i, j) (see inset). To simplify the expression of further equations, we say that $\theta_{ij} = \theta_{ji}$ and $\theta_{ii} = \pi$. We note that an angle-based representation is unaware of global position (3 DOFs), rotation (3 DOFs), and scale (1 DOF) of the shape. Therefore, we must add 7 transformation variables to our representation for completeness. Note that for applications in this paper, this transformation is usually computed after reconstruction.

We rely on minimal geometric constructions to prove equivalence. For each construction, we use simple geometric arguments to discover constraints. We gain additional certification of the property by a counting argument, showing that the number of variables in the vertex-based representation coincides with the number of degrees of freedom in the angle-based representation, i.e., the number of variables minus the number of equality constraints. We stress that counting arguments are used only as additional validation and do not actually constitute part of the proof. We first show that the dihedral angles of a single tetrahedron are enough to describe its shape, but that they are not independent. The same conclusions are drawn for two tetrahedra sharing a triangle, and for tetrahedra surrounding an edge. During this process, we show that three types of *structural* constraints and some inequalities are needed to ensure representation equivalence. The conclusion is that these structural constraints are necessary and sufficient to guarantee that the mesh can be built in $\mathbb{R}^3$.

Cell Constraints The simplest construction is composed of a single tetrahedron. Luo [1997] gives necessary and sufficient conditions on dihedral angles to recover vertex positions, for which we give a simple proof here. It is sufficient to show that for a set of dihedral angles satisfying some constraints, there exists, up to a rotation, a unique set of unit normals spanning three dimensions and not lying in the same half-space. Such normals uniquely determine vertex positions of a tetrahedron, up to an isotropic scale and a translation, using plane intersections.

We start from the identity $\sum_i \mathbf{n}_i a_i = \mathbf{0}$, where $\mathbf{n}_i$ and a_i are the unit normal and area of the ith triangle of tetrahedron t. Taking the dot product with each of the four triangle normals $\mathbf{n}_j$ keeps the identity and lets us write $-\sum_i \cos(\theta_{ij}) a_i = 0$. We rewrite these four identities using Gram matrix $\mathbf{G}_t$ of triangle normals as

$$\underbrace{\begin{bmatrix} 1 & -\cos\theta_{01} & -\cos\theta_{02} & -\cos\theta_{03} \\ -\cos\theta_{01} & 1 & -\cos\theta_{12} & -\cos\theta_{13} \\ -\cos\theta_{02} & -\cos\theta_{12} & 1 & -\cos\theta_{23} \\ -\cos\theta_{03} & -\cos\theta_{13} & -\cos\theta_{23} & 1 \end{bmatrix}}_{\mathbf{G}_t} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

This shows how dihedral angles are determined from normals. We now show how to uniquely recover normals from $\mathbf{G}_t$. First, we note that $\mathbf{G}_t$ is a Gram matrix of normals spanning three dimensions. Thus, $\mathbf{G}_t$ is rank deficient, which implies

$$|\mathbf{G}_t| = 0. \tag{1}$$

Additionally, $\mathbf{G}_t$ is constructed with normals that do not lie in the same half-space. Recalling that $\sum_i \mathbf{n}_i a_i = \mathbf{0}$ and that a_i are nonzero, this condition is satisfied if and only if a_i have the same sign, which we enforce as follows. Let $\mathbf{C}_t$ be the cofactor matrix of $\mathbf{G}_t$. It can be shown that entries $(\mathbf{C}_t)_{ij}$ are proportional to $a_i a_j$ [Barrett 1994]. Thus, we could either constrain $\mathbf{C}_t$ to be positive or negative. However, $\mathbf{G}_t$ is a rank 3 positive semidefinite matrix, which forces diagonal entries of $\mathbf{C}_t$ to be positive, giving

$$\mathbf{C}_t > 0, \tag{2}$$

where the inequality is applied element-wise, and for which we give the complete expression in Appendix A. We do not need to further enforce positive semidefiniteness since the smallest unconstrained principal submatrices have determinants 1 and $\sin^2\theta_{ij}$. Positive determinants of all principal submatrices ensure that $\mathbf{G}_t$ is positive semidefinite.

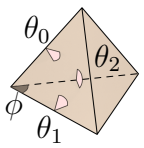
We now show that Constraints (1) and (2) are sufficient to recover normals. Since $\mathbf{G}_t$ is a rank 3 positive semidefinite matrix, there exists a unique matrix $\mathbf{N}_{3\times 4}$, up to a rotation, such that $\mathbf{G}_t = \mathbf{N}^T\mathbf{N}$. Columns of $\mathbf{N}$ are the desired normals. The unit diagonal and the rank of $\mathbf{G}_t$ ensure that normals have unit length and that they span three dimensions. The same sign property of the area eigenvector ensures that they do not lie in the same half-space.

As additional validation, we now count DOFs to ensure that they match the number of position variables. In terms of vertex positions, t has 12 DOFs (4 vertices $\times$ 3 coordinates). In terms of angles, t has 6 dihedral angle variables and 7 global transformation variables, for a total of 13 DOFs. Assuming that Constraints (1) and (2) are satisfied, there remain 12 DOFs, which supports the equivalence.

Each tetrahedron of the mesh should satisfy equality Constraint (1) and inequality Constraints (2). These constraints ensure the structural validity of each individual tetrahedron. However, the dihedral angles need to satisfy additional constraints that enforce the validity of the *assembly* of the tetrahedra, namely facet constraints and edge constraints detailed below.

Facet Constraints The next construction is the assembly of two neighbor tetrahedra t_0 and t_1 sharing a triangle. We assume that cell constraints are satisfied by both tetrahedra. At this point, each tetrahedron can vary its shape independently. However, the assembly of t_0 and t_1 requires that the two glued triangles have the same shape, for which a necessary and sufficient condition is that all three pairs of corresponding corner angles are equal. Since cell constraints are satisfied, we know that corner angles are in $[0, \pi]$. Thus, we can equivalently ensure that the cosine of corner angles are equal. Noting ϕ_i^0 and ϕ_i^1 the ith corner angles of glued triangles of t_0 and t_1, we have the three constraints

$$\cos\phi_i^0 = \cos\phi_i^1, \tag{3}$$



for which only two are independent by the identity $\phi_0^i + \phi_1^i + \phi_2^i = \pi$. The cosine of a corner angle can be computed from dihedral angles incident to the corner vertex. Following the notations in the inset, we have [Dittrich and Speziale 2008]

$$\cos\phi = \frac{\cos\theta_0 + \cos\theta_1\cos\theta_2}{\sin\theta_1\sin\theta_2}.$$

We now count DOFs to ensure that they match the number of position variables. In terms of vertex positions, this construction has 15 DOFs (5 vertices $\times$ 3 coordinates). In terms of angles, this construction has 12 dihedral angle variables and 7 global transformation variables, for a total of 19 DOFs. Assuming that cell and facet constraints are satisfied by both tetrahedra, there remain 15 DOFs, which supports the equivalence.

Each pair of neighbor tetrahedra introduces three constraints (Equation 3). Intuitively, they ensure that *strips* of tetrahedra glued along common faces can be reconstructed. For a general topological ball, we need to examine another configuration that corresponds to a *wheel* of tetrahedra sharing a common edge.

Edge Constraints The next construction is the assembly of tetrahedra looping around an edge. We assume that cell constraints are satisfied by all tetrahedra, and that facet constraints are satisfied by all pairs of neighbor tetrahedra. Even if all these constraints are satisfied, configurations exist where the wheel does not close. One can compare this phenomenon to angular defect around vertices of a curved surface. A necessary and sufficient condition for the wheel to close perfectly is to constrain the set Θ_e of dihedral angles around edge e with

$$\sum_{\theta \in \Theta_e} \theta = 2\pi, \tag{4}$$

which ensures that the edge is simply covered, and thus, that the local construction is injective.

In the 2D case (e.g., ABF), nothing constrains closing edges to be isometric, and thus, a wheel constraint is necessary. In our case, closing facets are similar due to facet constraints. However, these facets share an edge with other tetrahedra. The length of this edge being equal for all tetrahedra, the facets are not only similar, they are isometric. Thus, no wheel constraint is needed.

We now count DOFs to ensure that they match the number of position variables. We first consider the simple case of a wheel with only 3 tetrahedra. In terms of vertex positions, this construction has 15 DOFs (5 vertices $\times$ 3 coordinates). In terms of angles, this construction has 18 dihedral angle variables and 7 global transformation variables, for a total of 25 DOFs. Assuming that cell, facet, and edge constraints are satisfied for the 3 tetrahedra, 15 DOFs remain, which confirms the equivalence. For more general wheels, we proceed inductively. Adding a tetrahedron in the wheel adds one vertex. In terms of vertex positions, this adds 3 DOFs. In terms of angles, this adds 6 dihedral angle variables and 3 constraints, for a total of 3 DOFs. Thus, no more constraints are needed.

Each edge introduces one constraint (Equation 4), which lets us build topological balls. We now have a complete and valid representation of a volume immersed in $\mathbb{R}^3$ given by a global similarity transformation and dihedral angle variables such that cell, facet, and edge constraints are satisfied.

Vertex Constraints Local injectivity requires that each interior vertex is simply covered, i.e., that their solid angle is equal to 4π. However, the combined edge constraints around a vertex already enforce this equality [Yin et al. 2008], and thus, no vertex constraint is needed.

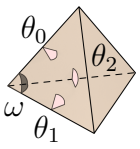
We also note that counting DOFs of a local construction containing a single interior vertex requires some care. Indeed, closing a 1-ring creates redundant constraints. This is comparable to facet constraints where only two out of the three constraints are independent. We stress the fact that they are redundant but not contradictory, and thus, do not invalidate our method.

3 Flattening

We now tackle the case where given dihedral angles do not satisfy all constraints. It occurs when geometric constraints are not satisfied by the data (Section 5.1), or when the user gives an arbitrary set of dihedral angles (Section 5.2). A failure to satisfy constraints reflect the impossibility to recover positions without distorting input angles. We control this distortion by solving the following optimization problem,

$$\begin{array}{lll} \min & f(\Theta) & \\ \text{s.t.} & |\mathbf{G}_t| = 0 \wedge \mathbf{C}_t > 0 \quad \forall t \in T & \text{Eqs. (1,2)} \\ & \cos\phi_c^0 = \cos\phi_c^1 \quad \forall c \in C & \text{Eq. (3)} \\ & \sum_{\theta \in \Theta_e} \theta = 2\pi \quad \forall e \in E & \text{Eq. (4)} \\ & \theta_b \leq \theta \leq \pi - \theta_b \quad \forall \theta \in \Theta, & \end{array} \tag{5}$$

where Θ is the vector of all dihedral angles, Θ_e is the set of dihedral angles of an edge e, T is the set of tetrahedra, C is the set of corners, E is the set of edges, ϕ_c^0 and ϕ_c^1 are corner angles of glued triangles t_0 and t_1 associated to c, θ_b is a positive parameter to bound dihedral angles, and $f(\Theta)$ is an objective function measuring distance to input dihedral angles.

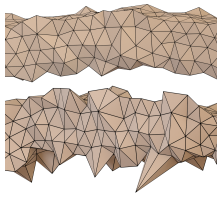


Objective Function We choose f to be a quadradic function of Θ that preserves the shape of tetrahedra defined by original dihedral angles Θ^*. We use a combination of two sets of geometric properties that are linearly obtained from dihedral angles. The first set of properties is composed of the dihedral angles themselves. The second set of properties is composed of solid angles at each vertex of all tetrahedra. A solid angle ω is a linear function of dihedral angles incident to a vertex in a tetrahedron. Using notations in the inset, we have

$$\omega = \theta_0 + \theta_1 + \theta_2 - \pi.$$

Noting Ω and Ω^* the vectors of solid angles computed from Θ and Θ^*, we define f as

$$f(\Theta) = \|\Theta - \Theta^*\|^2 + \|\Omega - \Omega^*\|^2.$$



The solid angle term tends to penalize elongated tetrahedra. The inset shows a connectivity shape (Section 5.3) flattened with (top) and without (bottom) the second term. Now that we have a complete formulation of the problem, we can solve it in practice in the following way.

Optimization To optimize the constrained nonlinear problem (5), we propose a multi-phase penalty method. Each phase minimizes an energy function that tries to balance between constraints satisfaction and objective function minimization in a nonlinear least-squares sense. After each phase, we set initial dihedral angles Θ^* to the current solution to let the next phase further satisfy the constraints. We repeat this procedure until the constraints are satisfied up to a given threshold. A solution is usually attained after two or three phases. Given initial dihedral angles Θ^*, equality constraints $\mathbf{c}(\Theta) = \{c_i(\Theta)\}$, and inequality constraints $\mathbf{d}(\Theta) = \{d_j(\Theta)\}$, we define the function to be minimized as

$$E(\Theta) = \alpha f(\Theta) + \|\mathbf{c}(\Theta)\|^2 + \|\min(\mathbf{d}(\Theta), 0)\|^2 \tag{6}$$

where constraints are expressed as $c_i(\Theta) = 0$ and $d_j(\Theta) > 0$, and α is a regularization factor. We set $\alpha = 10^{-6}$ for all our examples.

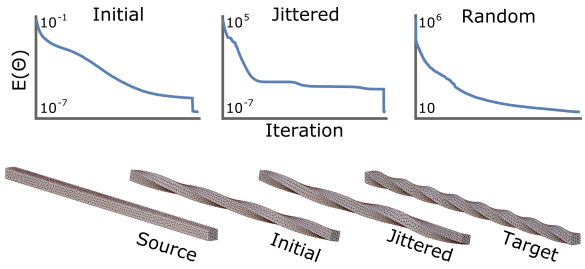


Figure 1: *Convergence plots visualized using log scale for shape interpolation of a twisted bar for different initial solutions: initial interpolated angles (left), jittered interpolated angles (center), and random angles (right). The random initial solution experiment converged to a local minimum and could not be reconstructed. Discontinuities in the plots are caused by a change of optimization phase.*

Constraint set $\mathbf{c}$ contains all constraints, including dependent ones, since we observed better convergence.

In this formulation, the objective function plays two roles. First, since the zero set of the constraint manifold contains every possible immersion of the mesh, it ensures that we do not stray too far from the input dihedral angles. Second, since some constraints are redundant, numerical errors due to the discretization can make simultaneous constraints impossible to satisfy exactly. In this case, the objective function regularizes the optimization and forces a compromise without ending up with degenerate solutions. The regularization factor is chosen to be very small so that the solver puts more efforts on satisfying the constraints.

We minimize E with a Quasi-Newton method called L-BFGS as implemented by Liu et al. [2009] using a M1QN3 preconditioner, and a memory of eight iterations. This solver only requires first derivatives of each constraint. We refer the reader to Appendix B for more details on computing constraint derivatives.

Initial Solution Since the optimization problem in Equation (5) is nonlinear and nonconvex, convergence to the global minimum is not guaranteed for arbitrary initial solutions using a local minimization method. We choose the initial solution to be the initial dihedral angles Θ^* specific to each application since experience shows that it tends to improve convergence. As shown in Figure 1, choosing a jittered or random initial solution tends to slow convergence or can lead to a local minimum. The jittered initial solution experiment started with a larger error and needed $5\times$ more iterations to converge.

4 Reconstruction

At this point, we have a set of dihedral angles satisfying all constraints. We can now proceed to recover positions. One could perform a greedy reconstruction of the entire mesh by laying out one tetrahedron at a time. However, this method is prone to numerical error accumulation and is only viable for small meshes (see Figure 2). Instead, we propose a robust linear spectral reconstruction method that distributes numerical errors uniformly across the mesh. To this end, we build a local construction for each tetrahedron. Then, we extract linear equations from pairs of neighboring tetrahedra relating their vertex positions using barycentric coordinates. We use these equations to build a linear system involving the entire mesh. Finally, we solve this linear system using eigendecomposition. The result is a set of vertex positions representing the immersion.

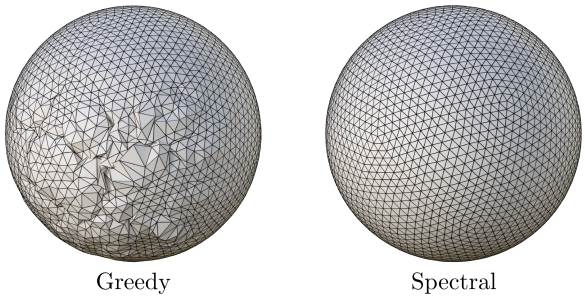
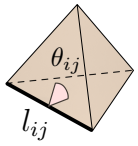


Figure 2: *Comparison between greedy and spectral reconstructions of a sphere mesh with jittered dihedral angles. Our spectral method adds robustness by distributing the error over the entire mesh.*

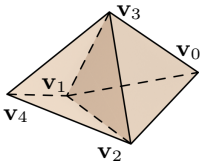
Local Constructions The first step is to lay out each tetrahedron independently using its dihedral angles. While angles are theoretically sufficient for reconstruction, we observed that recovering edge lengths of a tetrahedron beforehand, and laying out using edge-length geometry is more stable. This stability has also been observed in greedy reconstructions of triangle meshes [Springborn et al. 2008]. To this end, we use the following identity [Barrett 1994]

$$l_{ij} = b_t \frac{\partial |\mathbf{G}_t|}{\partial \theta_{ij}},$$

where b_t is a coefficient of proportionality, and l_{ij} is the length of the edge associated to θ_{ij} as shown in the inset. Even though scale is not important at this time, we choose b_t so that an arbitrary edge has unit length since it tends to make further computations more robust. Then, we perform a trilateration to recover local vertex positions as shown in Appendix C, which completes the construction.



Linear Relations Local constructions offer a localized view of the mesh. We now need to position them globally. We do this by expressing relative positions of neighbor tetrahedra using barycentric coordinates. For all pairs of tetrahedra (t_0, t_1) sharing a triangle, we glue the two local constructions t_0 and t_1 along their associated two triangles by transforming the vertex positions of t_0. We scale t_0 so that the two triangles are congruent. There only remains to find the position of the vertex v opposite to the associated triangle f in t_0. We compute local coordinates of v using two edges of f and its normal as basis. We compute a similar basis using the corresponding elements of t_1 and use it to obtain the transformed v.



Using notations in the inset, for two glued tetrahedra composed of five vertices having positions $\mathbf{v}_i$, where $(\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3)$ forms a tetrahedron, we express $\mathbf{v}_4$ by

$$\mathbf{v}_4 = a_0\mathbf{v}_0 + a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + a_3\mathbf{v}_3, \quad (7)$$

where a_i are barycentric coordinates computed by

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \mathbf{v}_0 & \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{v}_4 \\ 1 \end{bmatrix}.$$

We gather linear equations (7) from all pairs of tetrahedra into a linear system $\mathbf{Ax} = \mathbf{0}$, where $\mathbf{A}$ is a matrix composed of barycentric coordinates, and $\mathbf{x}$ are the unknown global positions of all vertices. We rewrite this overdetermined system into a least-squares formulation $\mathbf{A}^T\mathbf{Ax} = \mathbf{0}$ to which we must find the kernel.

Spectral Solver We cannot solve for $\mathbf{x}$ directly since the kernel is not empty. This is due to barycentric coordinates being invariant to affine transformations. Fixing four vertices would remove this invariance, but this would also introduce distortions near anchors [Mullen et al. 2008]. We instead use eigendecomposition to find the eigenvector associated to the smallest non-zero eigenvalue of $\mathbf{A}^T\mathbf{A}$. This is efficiently computed using an inverse power method as described in Algorithm 1, where we use $\mu = 10^{-8}$ and $\epsilon = 10^{-12}$. We solve the linear system using Cholesky decomposition, as implemented by CHOLMOD [Chen et al. 2008]. Since the matrix is constant throughout iterations, we factorize $\mathbf{B}$ beforehand, and then solve using back-substitution.

Algorithm 1 Spectral Solver

Input: a matrix $\mathbf{A}^T\mathbf{A}$ and an initial guess $\mathbf{x}$
Output: vertex positions $\mathbf{x}$
  $\mathbf{B} \leftarrow \mathbf{A}^T\mathbf{A} + \mu\mathbf{I}$
  $\mathbf{x} \leftarrow$ RemoveAffineTransformation($\mathbf{x}$)
  while $\|\mathbf{B}\mathbf{x} - (\mathbf{x}^T\mathbf{B}\mathbf{x})\,\mathbf{x}\|_\infty > \epsilon$ **do**
    Solve $\mathbf{B}\mathbf{y} = \mathbf{x}$
    $\mathbf{x} \leftarrow$ RemoveAffineTransformation($\mathbf{y}$)

To avoid degenerate solutions, we remove the global affine transformation contained in $\mathbf{x}$ at each iteration as follows. First, we center vertices around the origin. Then, we remove shearing by comparing current solution to local constructions. Let $\mathbf{J}_t$ be the matrix transforming tetrahedron t, as defined by the current solution, to its local construction. We compute average shearing $\mathbf{S}$ of all tetrahedra in log-space [Arsigny et al. 2006] as

$$\mathbf{S} = \exp\left(\frac{1}{2N}\sum_{t\in T}\ln \mathbf{J}_t^T\mathbf{J}_t\right),$$

where the logarithm of a symmetric matrix $\mathbf{M}$ is computed using its diagonalization $\ln\mathbf{M} = \ln(\mathbf{Q}\mathbf{D}\mathbf{Q}^T) = \mathbf{Q}\ln(\mathbf{D})\mathbf{Q}^T$, and the logarithm of a diagonal matrix is computed component-wise on diagonal elements only. We apply $\mathbf{S}$ to all vertices to remove shearing. Then, we remove rotation by aligning an arbitrary edge on the X-axis and one of its connected triangles on the XY-plane. Finally, we normalize $\mathbf{x}$ to remove global scale.

Solution $\mathbf{x}$ is the immersion in terms of vertex positions devoid of global scale, translation, and rotation. We choose the final rigid transformation to fulfill individual application needs.

5 Applications

We need to stress the fact that our motivation was mainly guided by scientific curiosity, and by the natural question of studying the structure of the space of admissible values for dihedral angles. We also studied potential applications based on this representation and on our constrained optimization procedure. We do not claim that our experiments perform better than the state of the art in volume parameterization, shape interpolation, and mesh compression in terms of computation time, but we think that besides our theoretical motivation, the experimental results below tend to confirm that our approach may have potential applications. In particular, we show that mesh optimization using our formulation manages to recover highly tangled meshes, which pose problems for many optimization methods. Moreover, we show that shape interpolation in terms of dihedral angles is well-behaved, even for large rotations.

We implemented the methods in C++ in the *Graphite* software [2015] and all tests were conducted on an Intel® Core™i7 930 processor with 12GB of memory and without any GPU acceleration. We took advantage of the multi-threading capacity of the multi-core architecture using OpenMP. Computation times are shown in Table 2.

From Positions to Dihedral Angles Some applications rely on the initial dihedral angles of the mesh. Thus, we must be able to convert vertex positions to dihedral angles. While the process is straightforward, doing it robustly requires some care. Most programming languages offer a two-argument inverse tangent function that is generally more robust than an inverse cosine function. For this reason, we use the following formula [Shewchuk 2013]

$$\theta_{ij} = \tan^{-1}\left(-\frac{3Vl_{ij}}{2\mathbf{n}_i\cdot\mathbf{n}_j}\right),$$

where θ_{ij} is a dihedral angle of tetrahedron t of volume V, l_{ij} is the length of the edge associated to θ_{ij}, and $\mathbf{n}_i$ is the area weighted normal of the ith triangle of t.

Geometric Constraints So far, we have looked at *structural* constraints that ensure that a mesh is well-formed. For some applications, we need to further constrain the shape of the mesh. For example, creating polycubes (Section 5.1) requires patches of the boundary of a mesh to be flat, and neighboring patches to be orthogonal. For mesh optimization (Section 5.4), we require the shape of the boundary of the mesh to remain unchanged. For both of these examples, we need *geometric* constraints to ensure that some geometric properties are satisfied by dihedral angles. We present two ways of constraining the shape of mesh M.

The first way is to reinterpret edge constraints (4) as follows. General immersions of volume geometries are described by their Riemann curvature tensor. An interesting property is that Riemann curvature is completely defined by sectional curvature. Regge [1961] shows that sectional curvature of a tetrahedral mesh is a discrete measure defined per edge e as $2\pi - \sum_{\theta\in\Theta_e}\theta$. To prescribe sectional curvature $\tilde{\theta}_e$ of e, we rewrite edge constraint (4) using the more general equation

$$\sum_{\theta\in\Theta_e}\theta = \tilde{\theta}_e,$$

where we sum over all dihedral angles incident to e. This curvature prescription lets us specify the flatness and orthogonality properties needed to create polycubes. The second way is to prescribe corner angles of boundary triangles of M. To do so, we add facet constraints as

$$\cos\phi_c = \cos\tilde{\phi}_c,$$

where ϕ_c is the angle of a corner c of M, and $\tilde{\phi}_i$ is its desired angle. This corner angle prescription lets us constrain the shape of the boundary needed for mesh optimization.

Note that these two methods are not mutually exclusive. One could use a combination of both if needed, but one must be careful to avoid contradictory constraints, in which case a solution will not exist. However, for the applications presented in this paper, it will be clear from the context that all constraints are compatible.

5.1 Volume Parameterization

Volume parameterization is the act of deforming the geometry of a volume so that it fits a simple regular domain. These simpler domains are quite popular for texturing and remeshing [Li et al. 2012; Livesu et al. 2013]. A dihedral angle-based representation fits perfectly for this application. In our context, the geometry of domains is specified by adding geometric constraints. We present two types of domains. First, we consider polycube domains with their planar boundaries, and then domains defined by singularity graphs adding interior singularities.

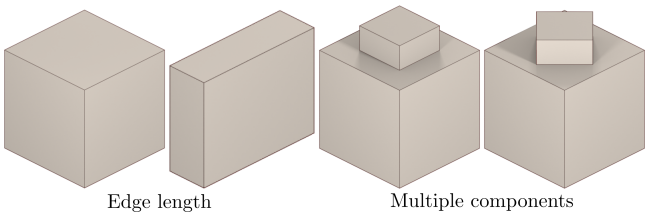


Figure 4: *Geometric ambiguities inherent to singularity graphs. We show two domains sharing the same singularity graph.*

For polycube creation, the input is a surface triangle mesh for which every triangle has a label specifying the axis that its normal must be aligned to. From this surface, we create a tetrahedral mesh using *TetGen* [Si 2007]. We compute initial dihedral angles for all tetrahedra using the initial vertex positions. Then, we flatten the mesh (Section 3) using boundary geometric constraints that force polycube faces to be flat, and polycube edges to have angle turns matching the change in normal orientation. Finally, we recover positions using spectral reconstruction (Section 4). Results are shown in Figure 3 (left, center).

For this particular application, it is possible to stop the optimization before convergence and let the reconstruction absorb remaining errors. Stopping the flattening process may produce a curved polycube. To fix this, we add the following equations to the linear system (7) during reconstruction to align each boundary triangle f_{ijk} to its assigned axis $\mathbf{n}$

$$(\mathbf{v}_j - \mathbf{v}_i) \cdot \mathbf{n} = 0$$
$$(\mathbf{v}_k - \mathbf{v}_j) \cdot \mathbf{n} = 0$$
$$(\mathbf{v}_i - \mathbf{v}_k) \cdot \mathbf{n} = 0.$$

In addition to geometric boundary constraints, users can define edge constraints that do not sum up to 2π for interior edges. Such constraints are useful for introducing interior singularities for hexahedral meshing [Li et al. 2012]. We define a *singularity graph* as the set of such interior edges along with boundary edges such that their edge constraint does not sum up to π. An example with a handcrafted graph is shown in Figure 3 (right).

Our framework offers some advantages over position-based techniques for volume parameterization using singularity graphs. First, since dihedral angles are independent of global positioning, each tetrahedron can be spatially disconnected from its neighbors. This implies that dihedral angles naturally encode nontrivial domains without changing the combinatorics of the mesh. Second, this shows that a singular graph alone is enough to perform parameterization. Thus, our work opens the door for new algorithms that could directly create singularity graphs, instead of being by-products of frame fields [Li et al. 2012].

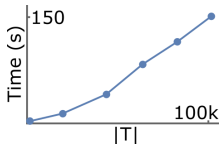
We note that two geometric informations are not encoded by singularity graphs (Figure 4). First, the length of each edge of the domain is not part of the formulation. Second, a singular graph with multiple components may have pairwise orientation ambiguities.

Combinatorial Considerations Given compatible geometric constraints, an immersion may not exist for a given mesh combinatorics. For example, two boundary triangles of a single tetrahedron cannot be coplanar without degenerating the tetrahedron. For the same reason, two edges of a single tetrahedron cannot be colinear. This observation has been made for polycube domains [Aigerman and Lipman 2013] and singularity graphs [Jiang et al. 2014]. We rewrite these two constraints here for completeness.

The first constraint is that there should be a sufficient number of tetrahedra around edge e for given curvature prescription $\tilde{\theta}_e$. Knowing that a single tetrahedron can support a dihedral angle up to π, we must have $\tilde{\theta}_e < |T_e|\pi$, where T_e is the set of tetrahedra incident to e. If an edge does not satisfy the condition, we select an arbitrary tetrahedron $t \in T_e$ and split the edge opposite to e in t. The second constraint is related to singularity graphs. The constraint is that a triangle f should not be incident to two singular edges. In this eventuality, we split the third edge of f. These corrections only *add* tetrahedra around edges, so they cannot break other constraints.

Note that these combinatorics constraints are not only inherent to our method, but to every method that deforms tetrahedral meshes.

Method Scaling We evaluate how our method scales with the number of tetrahedra by parameterizing a sphere to a cube for different resolutions. The inset log-log plot indicates that the computation time increases linearly with the number of tetrahedra. This behavior has been observed for most experiments. However, convergence rate also depends on geometric constraints complexity, hence the apparent nonlinear scaling between the Squirrel and the Buste models shown in Table 2.



5.2 Shape Interpolation

Shape interpolation is the creation of a smooth and natural transition between two different shapes. This application is inspired by the work of Chen et al. [2013], that proposes to interpolate planar shapes using squared edge lengths. The process of interpolating the metric, and then recovering positions using a robust conformal metric flattening method, creates a smooth deformation of bounded distortion. Unfortunately, current surface metric flattening methods do not trivially generalize to volumes, preventing direct application of this shape interpolation method to tetrahedral meshes. We propose to interpolate dihedral angles instead of squared edge lengths and to use our flattening-reconstruction procedure to recover the immersion.

The algorithm goes as follows. For two poses of a same tetrahedral mesh, we compute dihedral angles of each pose from vertex positions. Given a weight $w \in [0, 1]$, we linearly interpolate corresponding dihedral angles (θ_j^0, θ_j^1) as $\theta_j^* = (1 - w)\theta_j^0 + w\theta_j^1$, where θ_j^i is the jth dihedral angle of the ith mesh, and then we flatten resulting angles θ_j^* and recover vertex positions to obtain the interpolated mesh. One can also interpolate between n poses using n weights $w_i \geq 0$, $\sum_i w_i = 1$. Finally, we fix the global similarity transformation by interpolating the center of gravity, cancelling global rotation, and preserving shape volume. Results are shown in Figure 5.

Our method is most similar to the second-order interpolation method of Kircher and Garland [2008], which expresses changes in Jacobian matrices across the mesh. As ours, their formulation does not depend on global coordinates and thus, we achieve similar results. However, our method has the benefit of guaranteeing local injectivity (this guarantee comes at the cost of a longer computation time). We note that our method differs from techniques working in global coordinates, such as ARAP [Alexa et al. 2000], that may have problems with large rotations and require special care to minimize fold-overs [Levi and Gotsman 2015].

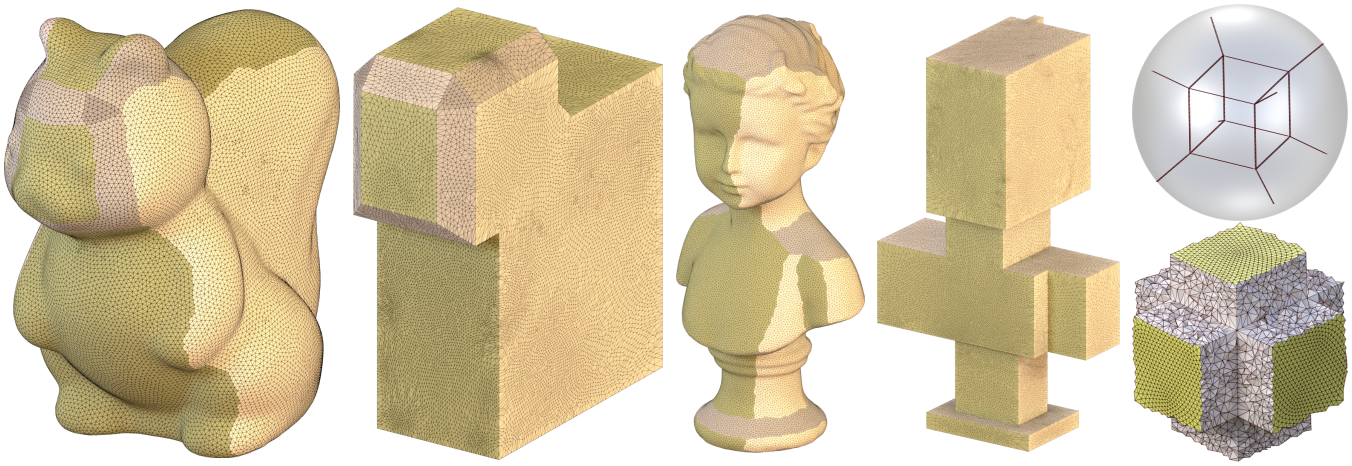


Figure 3: *Volume parameterization using polycube constraints (left and center) and a singularity graph (right). The boundary surface of all examples are constrained to be planar, but singularities are either on boundaries or inside the mesh.*

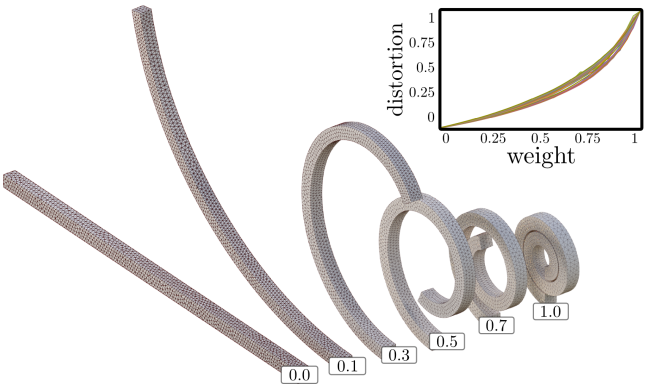


Figure 5: *We linearly interpolate dihedral angles from a straight bar (left) to a curled bar (right). We show the condition number, normalized in* $[0, 1]$*, of the map relative to the straight bar for the 100 most distorted tetrahedra.*

5.3 Connectivity Shapes

The connectivity shape concept, as introduced by Isenburg et al. [2001], shows that a lot of geometric information is encoded into its combinatorics. To prove their point, they start from a surface mesh, erase all vertex positions, and manage to recover an approximation of the original shape using an optimization procedure. This procedure simply tries to place vertices in space such that every edge has approximately the same length.

We extend this idea to volume mesh. Instead of using uniform edge length as the main criterion, we use uniform dihedral angles. We start from the combinatorics of a mesh, set dihedral angles of regular tetrahedra, i.e., $\cos^{-1}(1/3)$, and flatten the result without any geometric constraints. The result is a shape such that each tetrahedron is as close as possible to a regular tetrahedron, while maintaining coherency with its neighbors. A result is shown in Figure 6.

This concept is not an application on its own. Nevertheless, it demonstrates the power of our representation, i.e., we do not need an initial immersion of the mesh. This property becomes useful when serving a more practical purpose, such as mesh optimization, as shown in the next section.

Figure 6: *Connectivity shape illustrated with the foot model. The original immersion (left) has its vertex positions entirely removed. Using only the combinatorics, we can approximate them (right).*

5.4 Mesh Optimization

We apply our method to mesh optimization. We focus on optimizing the shape of each tetrahedron without modifying the connectivity of the mesh. For this process, we use the flattening procedure as a mean to start from regular tetrahedron dihedral angles, and optimize them so they form a reconstructible set of angles. We include geometric constraints for boundary edges and triangle corners to preserve the original boundary shape. We fix boundary vertices to their original positions during reconstruction. We note that constraining both boundary dihedral angles and corner angles during flattening ensures that there are no DOFs left for the boundary to vary its shape. For this reason, this application supports higher genus meshes.

This procedure ignores the original geometry of internal tetrahedra. This has the advantage of supporting tangled meshes having inverted and degenerate tetrahedra. By setting the initial angles to ideal angles, we ensure that the result will be as close as possible to regular tetrahedra. In Figure 7, we show initial meshes of a sphere and fertility model with jittered interior vertices with many inverted tetrahedra. The mesh optimization procedure improves the quality of initial meshes and remove all inverted or degenerated tetrahedra.

Our mesh optimization method resembles that of Aigerman et al. [2013], which aims to improve the quality of each tetrahedron by bounding distortions relative to a regular tetrahedron. The key difference is that they measure distortions using the ratio of the maximum and minimum transformation matrix eigenvalue while we measure

Model	IBDM (min/max DA)	Ours (min/max DA)
Duck	16° / 148°	17° / 146°
Elephant (v1)	16° / 148°	17° / 146°
Elephant (v2)	18° / 147°	19° / 146°
Hand	17° / 148°	18° / 146°
Horse	16° / 148°	18° / 146°
Max Plank	21° / 148°	22° / 146°
Rocker	16° / 148°	18° / 146°
Skull	14° / 153°	16° / 151°

Table 1: *Results of mesh optimization using models from Aigerman et al. [2013] (IBDM). We show the minimum and maximum dihedral angles (DA) of resulting meshes using IBDM and our method.*

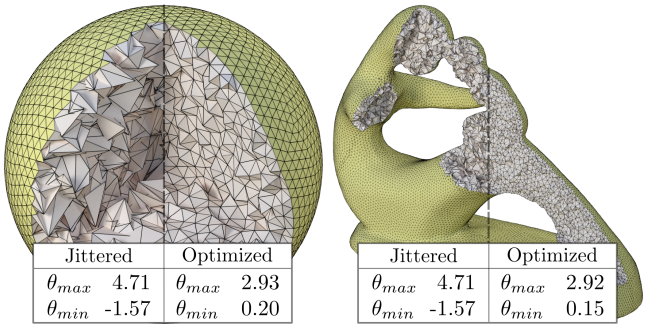


Figure 7: *Optimization of interior vertices of a sphere and fertility model. Interior vertices have been jittered (left) to introduce inversions. After optimization (right), the geometry of all tetrahedra was recovered without any inversion, as shown by the minimum and maximum dihedral angles.*

differences in dihedral angles. As shown in Table 1, our method is able to improve extremal dihedral angles for all experiments using their models as a starting point. As noted by the authors, bounding dihedral angles does not bound aspect-ratio and could create elongated elements. However, the solid angle term contained in our objective function tends to minimize this kind of distortions.

5.5 Mesh Compression

Dihedral angles can be used as a way to compress the geometry of a mesh. Our idea is to quantize angles using a coarse approximation, for example, using only 2 bits per angle, and recover vertex positions from this information alone. One could quantize angles on a uniform sampling of $[0, \pi]$. However, a good sampling should take advantage of the fact that the dihedral angles of a good quality mesh are concentrated around $\cos^{-1}(1/3)$. To this end, we propose to guide the quantization using the histogram of all dihedral angles.

First, we compute dihedral angles of the initial geometry using vertex positions. Then, given a budget of n bits per angle, we build the histogram by creating a single bin containing all dihedral angles and recursively split it around its mean value until we have 2^n bins. Our sampling is the mean value of each bin. Finally, we encode each angle using the index of its nearest sample. We also need to keep the table of mean values for decompression. Note, however, that the space used by this table is negligible for coarse approximations.

To recover the original geometry, we set initial dihedral angle θ_i^* to the mean value associated to its index. We flatten θ_i^*, and then we recover vertex positions using spectral reconstruction. Results are shown in Figures 8 and 9.

For all examples, we are able to reduce to 2 bits per angle before distortions start to be visually perceived. This high compression ratio is due to geometric redundancies in dihedral angle variables. Indeed, there is approximately $8\times$ more angle variables than position variables, which means that we encode an average mesh using 48 bits per vertex. We do not compete with state-of-the-art methods that achieve an average of 8 to 16 bits per vertex [Maglo et al. 2015]. However, exploiting redundancies in angle variables to achieve higher compression ratios is subject to future work. Moreover, one could choose to keep more geometric information on the boundary, and little to no information for the interior. Since there is generally more interior variables than boundary ones, this strategy should improve compression ratios.

Application	Model	Tets	θ_b	Iters	Time
Parameterization	Squirrel	119k	0.20	2×2k	493
	Buste	160k	0.30	3×3k	1502
	Sphere	37k	0.10	2×2k	129
	Torus	6k	0.10	2×2k	21
Interpolation	Bar	9k	0.10	2×2k	7
Optimization	Sphere	37k	0.20	2×2k	129
	Fertility	364k	0.15	2×2k	1550
Connectivity	Foot	158k	0.10	2×2k	684
Compression	Cube	47k	0.10	2×2k	50
	Bimba	124k	0.10	2×2k	156
	Buste	214k	0.10	2×2k	274

Table 2: *Statistics for all examples, where the number of iterations (iters) is the number of phases times the number of iterations per phase, and timings are in seconds.*

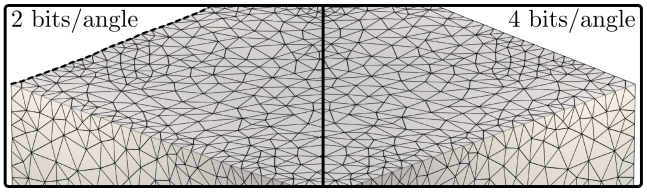


Figure 8: *We compress the geometry of a cube mesh using its dihedral angles at different bit rates. Keeping 4 bits per angle creates no visible artifact, while some irregularities can be seen on edges (dashed) when keeping only 2 bits.*

6 Discussion and Conclusion

We have shown that the shape of a tetrahedral mesh is completely determined by its dihedral angles, and have introduced constraints that must be satisfied to immerse the mesh in $\mathbb{R}^3$. This has lead us to an optimization procedure in the case where constraints are not readily satisfied. We then developed a robust spectral reconstruction method to recover positions. The result is an immersion of the mesh, which implies that there are no inverted tetrahedra and no multiply-covered vertices.

Furthermore, we demonstrated the applicability of our method in different contexts of volume geometry processing. In these applications, in terms of quality, our results are comparable to the state of the art, and we think that the additional robustness of locally injective maps guaranteed by our method may be a desirable property. However, this property comes at the expense of several limitations, which suggests future directions of research to overcome them.

Nonlinear Optimization Our formulation as a nonlinear, nonconvex optimization problem is reputably hard to solve numerically. As a consequence, the performance of our method does not match the

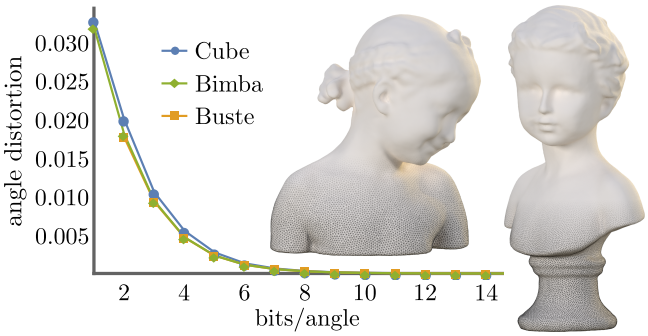
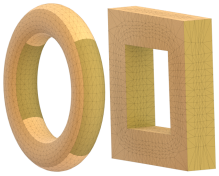


Figure 9: *Angle distortion of the compressed cube (Figure 8), Bimba (left), and Buste (right) meshes at different bit rates. We show reconstruction of compressed meshes using only 2 bits per angle.*

state of the art. In addition, while experience shows that our nonlinear least-squares formulation is well-behaved for most experiments, we have experienced some failure cases where the solver got stuck in local minima (see Figure 1). Convexifying the problem would improve convergence rate and ensure a single global minimum.



Genus > 0 We remind the reader that this method applies only to topological balls, with an exception being the mesh optimization application. Dealing with global constraints has always been a challenge for geometry algorithms working in local coordinates, and this work is no exception. Each additional handle needs global constraints to ensure that it closes perfectly. One way of solving this limitation is to let the spectral reconstruction deal with remaining errors introduced by the missing constraints. As shown in the inset, we successfully parameterized a torus to a polycube with a maximum difference of 0.4° between flattened and reconstructed dihedral angles. Note, however, that the spectral reconstruction absorbs errors in the least-squares sense, and thus, does not guarantee injectivity. To remedy this, one could instead temporarily fill all tunnels before flattening, and remove them after reconstruction. Another possibility would be to study the structure of the constraints and feasible set. We think that the co-homology basis plays a central role in this structure, and that each generator adds a set of constraints.

Global Injectivity While our method ensures local injectivity for the mesh interior, we do not guarantee, similar to 2D methods, that the boundary does not self-intersect. We could potentially achieve global injectivity using techniques similar to those suggested by Sheffer and de Sturler [2001], i.e., by iteratively reconstructing the shape while adjusting boundary constraints to avoid self-intersection.

Acknowledgements

We thank anonymous reviewers for their valuable comments. The work of the Canadian authors was supported by NSERC and GRAND NCE, and Bruno Lévy was supported by ANR BECASIM and ANR MORPHO. The Squirrel, Buste, Fertility, Foot, and Bimba models are provided courtesy of AIM@SHAPE Shape Repository.

References

AIGERMAN, N., AND LIPMAN, Y. 2013. Injective and bounded distortion mappings in 3D. *ACM Trans. Graph. 32*, 4, 106:1–14.

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-rigid-as-possible shape interpolation. In *Proc. of SIGGRAPH '00*, 157–164.

ARSIGNY, V., FILLARD, P., PENNEC, X., AND AYACHE, N. 2006. Log-Euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic Resonance in Medicine 56*, 2, 411–421.

BARRETT, J. W. 1994. First order Regge calculus. *Classical and Quantum Gravity 11*, 11, 2723–2730.

CERVONE, D. P. 1996. Tight immersions of simplicial surfaces in three space. *Topology 35*, 4, 863–873.

CHEN, Y., DAVIS, T. A., HAGER, W. W., AND RAJAMANICKAM, S. 2008. Algorithm 887: CHOLMOD, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw. 35*, 3, 22:1–14.

CHEN, R., WEBER, O., KEREN, D., AND BEN-CHEN, M. 2013. Planar shape interpolation with bounded distortion. *ACM Trans. Graph. 32*, 4, 108:1–12.

CRANE, K., PINKALL, U., AND SCHRÖDER, P. 2011. Spin transformations of discrete surfaces. *ACM Trans. Graph. 30*, 4, 104:1–10.

CRANE, K., PINKALL, U., AND SCHRÖDER, P. 2013. Robust fairing via conformal curvature flow. *ACM Trans. Graph. 32*, 4, 61:1–10.

DI BATTISTA, G., AND VISMARA, L. 1993. Angles of planar triangular graphs. In *Proc. ACM Symp. on Theory of Computing (STOC)*, 431–437.

DITTRICH, B., AND SPEZIALE, S. 2008. Area-angle variables for general relativity. *New Journal of Physics 10*, 8, 083006.

GRAPHITE, 2015, Jan. http://alice.loria.fr.

ISENBURG, M., GUMHOLD, S., AND GOTSMAN, C. 2001. Connectivity shapes. In *Proc. Conf. on Visualization (VIS)*, 135–142.

JIANG, T., HUANG, J., WANG, Y., TONG, Y., AND BAO, H. 2014. Frame field singularity correction for automatic hexahedralization. *IEEE Trans. Vis. Comput. Graphics 20*, 8, 1189–1199.

JU, T., SCHAEFER, S., AND WARREN, J. 2005. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. 24*, 3, 561–566.

KIRCHER, S., AND GARLAND, M. 2008. Free-form motion processing. *ACM Trans. Graph. 27*, 2, 12:1–13.

LABELLE, F., AND SHEWCHUK, J. R. 2007. Isosurface stuffing: Fast tetrahedral meshes with good dihedral angles. *ACM Trans. Graph. 26*, 3, 57:1–10.

LEVI, Z., AND GOTSMAN, C. 2015. Smooth rotation enhanced as-rigid-as-possible mesh animation. *IEEE Trans. Vis. Comput. Graphics 21*, 2, 264–277.

LI, Y., LIU, Y., XU, W., WANG, W., AND GUO, B. 2012. All-hex meshing using singularity-restricted field. *ACM Trans. Graph. 31*, 6, 177:1–11.

LIU, Y., WANG, W., LÉVY, B., SUN, F., YAN, D.-M., LU, L., AND YANG, C. 2009. On centroidal voronoi tessellation - energy smoothness and fast computation. *ACM Trans. Graph. 28*, 4, 101:1–17.

LIVESU, M., VINING, N., SHEFFER, A., GREGSON, J., AND SCATENI, R. 2013. Polycut: Monotone graph-cuts for polycube base-complex construction. *ACM Trans. Graph. 32*, 6, 171:1–12.

LUO, F. 1997. On a problem of Fenchel. *Geometriae Dedicata 64*, 3, 277–282.

MAGLO, A., LAVOUÉ, G., DUPONT, F., AND HUDELOT, C. 2015. 3D mesh compression: Survey, comparisons, and emerging trends. *ACM Comput. Surv. 47*, 3, 44:1–44:41.

MULLEN, P., TONG, Y., ALLIEZ, P., AND DESBRUN, M. 2008. Spectral conformal parameterization. *Comput. Graph. Forum 27*, 5, 1487–1494.

PENTLAND, A., AND WILLIAMS, J. R. 1989. Good vibrations: Modal dynamics for graphics and animation. *ACM Trans. Graph. 23*, 3, 207–214.

PETERSEN, K. B., AND PEDERSEN, M. S. 2012. *The Matrix Cookbook*. Technical University of Denmark.

REGGE, T. 1961. General relativity without coordinates. *Nuovo Cim. 19*, 3, 558–571.

SHEFFER, A., AND DE STURLER, E. 2001. Parameterization of faceted surfaces for meshing using angle-based flattening. *Engineering with Computers 17*, 3, 326–337.

SHEFFER, A., LÉVY, B., MOGILNITSKY, M., AND BOGOMYAKOV, A. 2005. ABF++: fast and robust angle based flattening. *ACM Trans. Graph. 24*, 2, 311–330.

SHEWCHUK, J. R., 2013. Lecture notes on geometric robustness. University of California at Berkeley.

SI, H., 2007. TetGen. A quality tetrahedral mesh generator and three-dimensional Delaunay triangulator. `http://tetgen.berlios.de`.

SPRINGBORN, B., SCHRÖDER, P., AND PINKALL, U. 2008. Conformal equivalence of triangle meshes. *ACM Trans. Graph. 27*, 3, 77:1–11.

YIN, X., JIN, M., LUO, F., AND GU, X. 2008. Discrete curvature flow for hyperbolic 3-manifolds with complete geodesic boundaries. In *International Symp. on Visual Computing (ISVC)*.

ZAYER, R., RÖSSL, C., AND SEIDEL, H.-P. 2005. Variations on angle based flattening. In *Advances in Multiresolution for Geometric Modelling*, 187–199.

ZAYER, R., LÉVY, B., AND SEIDEL, H.-P. 2007. Linear angle based parameterization. In *Proc. EG/ACM Symp. on Geometry Processing (SGP)*, 135–141.

A Cell Inequality Constraints

Cofactor matrix $\mathbf{C}$ of $\mathbf{G}$ is constrained to have positive entries. First, we note that $\mathbf{C}$ is symmetric, which eliminates six inequalities. Let (i, j, k, l) be a permutation of $(0, 1, 2, 3)$, $c_{ij} = \cos\theta_{ij}$ and $s_{ij} = \sin\theta_{ij}$. Off-diagonal entries of $\mathbf{C}$ can be written as

$$\mathbf{C}_{ij} = c_{ij}s_{kl}^2 + c_{il}(c_{jl} + c_{jk}c_{kl}) + c_{ik}(c_{jk} + c_{jl}c_{kl}).$$

Diagonal inequalities can be simplified using the following observations. The minor matrix $\mathbf{M}_{ii}$ of $\mathbf{G}$ is the Gram matrix of a triangle formed with corner angles $(\theta_{jk}, \theta_{kl}, \theta_{lj})$, and a triangle is spherical if and only if its Gram matrix is positive definite [Luo 1997]. However, inequality $\mathbf{C}_{ii} = |\mathbf{M}_{ii}| > 0$ and the fact that principal submatrices of $\mathbf{M}_{ii}$ have positive determinants ensures that $\mathbf{M}_{ii}$ is positive definite. Thus, we can replace $\mathbf{C}_{ii} > 0$ with

$$\theta_{jk} + \theta_{kl} + \theta_{lj} > \pi.$$

B Constraint Derivatives

Minimizing energy function (6) requires derivatives of constraints. Let (i, j, k, l) be a permutation of $(0, 1, 2, 3)$, $\partial_{ij} = \partial/\partial\theta_{ij}$, $\partial_i = \partial/\partial\theta_i$, $c_{ij} = \cos\theta_{ij}$, and $s_{ij} = \sin\theta_{ij}$. Following identities from The Matrix Cookbook [Petersen and Pedersen 2012], we have

$$\partial_i \cos\phi = \begin{cases} \frac{-\sin\theta_0}{\sin\theta_1 \sin\theta_2}, & \text{if } i = 0 \\ -\cot\theta_{3-i} - \cos\phi\cot\theta_i, & \text{otherwise} \end{cases}$$
$$\partial_{ij}|\mathbf{G}| = 2s_{ij}\mathbf{C}_{ij},$$
$$\partial_{ij}\mathbf{C}_{ij} = -s_{ij}s_{kl}^2,$$
$$\partial_{il}\mathbf{C}_{ij} = -s_{il}(c_{jl} + c_{jk}c_{kl}),$$
$$\partial_{kl}\mathbf{C}_{ij} = -s_{kl}(c_{ik}c_{jl} + c_{il}c_{jk} - 2c_{ij}c_{kl}),$$

where $\mathbf{C}$ is the cofactor matrix of $\mathbf{G}$.

C Trilateration

Given the length of each edge of a tetrahedron t, we can recover vertex positions using *trilateration*. We compute positions using intersection points of circles and spheres. The process is intuitive, but computations can be tedious. We rewrite these equations here to ease implementation. Since lengths do not encode translation and rotation, we arbitrarily set a vertex at the origin, an edge on the X-axis, and a triangle on the XY-plane. Noting $\mathbf{v}_i$ the position of the vertex opposed to the ith triangle of t, and l_{ij} the length of edge incident to the ith and jth triangle of t, we have

$$\begin{aligned} \mathbf{v}_0 &= (0, 0, 0), & x_2 &= \frac{l_{23}^2 + l_{13}^2 - l_{03}^2}{2l_{23}}, \\ \mathbf{v}_1 &= (l_{23}, 0, 0), & y_2 &= \sqrt{l_{13}^2 - x_2^2}, \\ \mathbf{v}_2 &= (x_2, y_2, 0), & x_3 &= \frac{l_{23}^2 + l_{12}^2 - l_{02}^2}{2l_{23}}, \\ \mathbf{v}_3 &= (x_3, y_3, z_3), & y_3 &= \frac{l_{12}^2 - l_{01}^2 - 2x_2x_3 + y_2^2 + x_2^2}{2y_2}, \\ & & z_3 &= \sqrt{l_{12}^2 - x_3^2 - y_3^2}. \end{aligned}$$