Computers & Graphics (2017)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/results-in-physics

Hexahedral Mesh Quality Improvement via Edge-Angle Optimization

Kaoji Xu[a], Xifeng Gao[b,1], Guoning Chen[a,*]

[a] *University of Houston, Houston, TX, USA*
[b] *New York University, Newyork, NY, USA*

ARTICLE INFO

Article history:

Keywords: Hex-mesh, Untangling, Quality Improvement, Optimization

ABSTRACT

We introduce a simple and practical technique to untangle and improve hexahedral (hex-) meshes. We achieve that by enabling the deformation of the boundary surfaces during the untangling process, which provides more space to reach a valid solution. To improve the element quality, an angle optimization strategy is proposed, which has much simpler formulation than the existing method. The deformed volume after optimization is then pulled back to the original one using an inversion-free deformation. In contrast to the current methods, we perform the untangling and quality improvement within a few local regions surrounding elements with undesired quality, which can effectively improve the minimum scaled Jacobian (MSJ) quality of the mesh over the existing method. We demonstrate the effectiveness of our methods by applying it to the hex-meshes generated by a range of methods.

1. Introduction

Hexahedral (or hex-) meshes, are commonly employed by many critical applications that require to solve volumetric partial differential equations. This is mostly due to its naturally embedded tensor product structure, larger tolerance for anisotropy and less numerical stiffness, compared to unstructured meshes (e.g., tetrahedral (or tet-) meshes). These preferred properties enable the convenient imposition of a simulation basis with a higher derivative smoothness between elements of the mesh, and the handling of large deformation during simulations.

However, given any input models, generating hex-meshes with good quality elements while conforming to the surface configuration remains an ongoing challenge. The initially computed hex-meshes, produced by the state-of-the-art methods, such as the polycube mapping or frame-field based methods, often contain inverted elements (i.e., elements with a negative local volume at one or more of its corners), which cannot be directly applied for finite element calculations [1]. Therefore, there is a need for hex-mesh improvement to eliminate the inverted elements and regulate the element shapes [2] while preserving surface features.

A number of techniques have been proposed to untangle and improve hex-meshes with inverted elements without changing their connectivity [2, 3, 4, 5, 6, 7]. However, none of them is guaranteed to produce inversion-free hex-meshes. Recently, Livesu et al. [8] introduced an untangling method that optimizes the cone-shapes around the individual edges of the hex-mesh to ensure a positive volume for the tetrahedra around the edges. The formulation of their energy function contains several terms that optimize different geometric characteristics of the mesh. However, the optimization is performed globally with varying weights that prefer elements that already have a good shape. While this strategy helps retain the elements with good quality (i.e. by fixing them), it may prevent the improvement of elements with less optimal quality.

In this work, we propose a local untangling and improvement framework so that the optimization is performed only around inverted elements or elements with quality below a user-

*Corresponding author: Tel.: +1-713-743-5788;
e-mail: gchen16@uh.edu (Guoning Chen)

specified minimum value (i.e., minimum scaled Jacobian [9], or MSJ). In our local framework, the focus is on improving those elements with undesired quality (i.e., good quality element may become slightly worse), which relieves the stiffness in the global optimization caused by the elements with good quality, allowing the MSJ quality to be further improved. In the meantime, we introduce a new angle-based distortion energy that characterizes different optimization goals (e.g., orthogonality and straightness) via a unified formulation, largely simplifying the setup and solving of the system. Furthermore, to facilitate the search of a valid solution to our optimization, the boundary surface is relaxed if needed. However, relaxing the surface constraint may lead to a large surface distance between the boundary of the output mesh and the original surface. To address that, we perform an inversion-free deformation that gradually pulls the surface back to its original one while still guaranteeing an inversion-free outcome. Note that this inversion-free deformation is only performed after the untangling process. For the improvement of MSJ, this pull-back process is not applied, as it may worsen the MSJ – against the goal of MSJ improving. Instead, we directly project the surface back to the original one after improving the MSJ of an inversion-free mesh. After improving the MSJ to a user desired level, we perform a Laplacian-like smoothing to improve the average scaled Jacobian (ASJ) of the mesh. Our framework is simple to implement and can handle more challenging inputs than the existing methods. In average, our method takes 2 minutes for a mesh with 10k-20k elements. We have applied our method to over 80 meshes generated by the polycube-based methods, octree-based method, and frame-field based method , respectively, to demonstrate its effectiveness. All our results have been submitted as the supplemental material, and a reference implementation will be released upon acceptance.

2. Related Work

In this section, we review the most relevant literature for the creation and optimization of hex-meshes.

Hex-meshing. Considering its importance to finite element simulation [10], a large amount of effort has been dedicated to the generation of valid all-hex meshes. These methods range from the early sweeping and paving [11, 12], grid-based [13, 14, 15, 16] and octree-based methods [17, 18, 19, 20] to the polycube-based [21, 22, 23, 24] and frame-field based approaches [25, 26, 27, 28]. A recent survey [29] provides a detailed look at the advances in this direction. Despite these many existing hex-meshing techniques, most initial hex-meshes generated with these approaches need to undergo a quality optimization process to substantially improve their quality for practical use. Our method can be used to optimize the initial meshes produced by a variety of these methods.

Hex-Mesh Optimization. Since it is a necessary step in the meshing pipeline, an equally large amount of work for the improvement of the hex-mesh quality has been proposed. There are two different strategies to improve the mesh quality. The first strategy adopts various smoothing (e.g., the Winslow smoothing [30]) and optimization methods (e.g., via the geometric flow [31]) to optimize the mesh without changing its connectivity, while the second strategy requires the modification of the mesh connectivity to achieve the desired quality improvement, such as the padding process [18, 32] typically used in the polycube-based methods. Other methods, like the singularity alignment [33] and polycube domain simplification [34, 35] have been proposed to optimize the structure of the hex-meshes. Our method belongs to the first group.

In order to optimize the quality of a hex-mesh, a quality metric has to be identified for the optimizer to improve upon the mesh. According to a Sandia Report by Stimpson et al. [9], there are more than a dozen quality metrics for hex-meshes. Most of these quality metrics measure the difference between a given hexahedron and a canonical cube via either angle distortion, length ratio or tensor distortion. Although there is not a comprehensive study on the effectiveness of these metrics [36], the scaled Jacobian metrics are the most commonly used metrics in the meshing and simulation communities. Intuitively, the Jacobian metric measures the solid angle distortion at the corners of a hexahedron. If the solid angles at the corners are all 90°, the scaled Jacobian achieves the optimal value of 1. It is well-known that a hexahedron can be decomposed into eight overlapping tetrahedra. It may be natural to use various tet-mesh optimization techniques [37, 38] to optimize these individual tetrahedra. It is also worth noting that many simplicial and polygonal map optimization techniques [39, 40, 41] can also be applied to optimize tet-meshes. However, as already shown in the work by Livesu et al. [8], simply optimizing the tetrahedra associated with the corners of a hexahedron may not improve its quality. Fu et al. [42] introduced an advanced MIPS method for computing locally injective mappings, which can be used to substantially improve the quality of a couple hex-meshes. However, only a few simple hex-meshes with no inverted elements were used in their testing. It is unclear how general this can be when applied to other hex-meshes with a substantially lower quality.

Besides that, many other hex-mesh optimization techniques exist. As reviewed by Wilson [43] and Livesu et al. [8], these techniques generally focus on untangling inverted elements (i.e., with negative scaled Jacobian) and improving the average element quality. Knupp introduced techniques to untangle the inverted elements [2] and improve the overall quality of the hex-mesh [3], which later have been integrated into the famous Mesquite library [4]. Specifically, the Mesquite library attempts to simultaneously untangle and improve the hex-mesh by minimizing an ℓ_1 function. However, since it optimizes one vertex at a time, the performance of Mesquite is slow when applied to hex-meshes with a large number of inverted elements. Later methods resort to local Gauss-Seidel approaches to iteratively untangle and smooth meshes [5, 6, 7]. Besides the Gauss-Seidel optimization strategies, non-linear optimization has also been applied to improve the hex-mesh quality [43]. Other optimization techniques for specific types of hex-meshes also exist, such as the quality improvement method for octree-based hex-meshes by Sun et al. [44]. Like many existing approaches, our method can handle hex-meshes generated by different methods (Section 4).

Recently, Livesu et al. [8] introduced the edge cone descriptor that indirectly measures the distortion of the hexahedra via a set of tetrahedra around each mesh edge. Based on this descriptor, a non-linear energy function is defined globally. To solve it, a local-global strategy is applied. As shown by the authors, this approach can untangle meshes that previous methods may fail to untangle. Therefore, we consider this method state-of-the-art and compare our method with it in this paper.

3. Methodology

Similar to many mesh optimizers, given an input mesh with a valid all-hex connectivity, our method first corrects the inverted elements, then improves the overall mesh quality. We also allow the boundary vertices to move out of the original volume if a valid solution cannot be found during untangling. This relaxation alleviates the difficulty of untangling elements at the concave areas of the surface. However, different from most methods, we directly measure the distance of the angles between pairs of connected edges from their respective ideal angles, leading to an intuitive and unified distortion energy formulation. In summary, our method consists of the following key steps (Fig. 1).

Compute target surface. In this step we improve the quality of the surface and associate surface vertices with the features detected from the input mesh (Section 3.1).

Untangling. We detect all inverted elements based on their scaled Jacobians. A local optimizer coupled with a surface relaxation strategy is then used to untangle those inverted elements iteratively until an inversion-free outcome is obtained (Section 3.2).

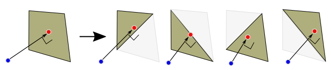
Inversion-free volume deformation. Due to the relaxation of surface constraint, after the above untangling process, the boundary surface of the output inversion-free hex-mesh may be far away from the original surface. We then perform an inversion-free deformation to pull the current surface back to its original one procedurally (Section 3.3). This step is optional, most models do not need this step.

Improve MSJ. Even though the mesh is currently inversion-free (i.e., all elements have positive scaled Jacobian), its MSJ may still be too low for practical use. To further improve the MSJ, we adopt the above untangling process but with a larger target MSJ (> 0) set by the user and perform the same local optimization (Section 3.4). In other words, the above untangling process can be considered as an optimization with the target MSJ$= 0$.

After achieving the target MSJ, the obtained hex-mesh may undergo a global optimization to improve its average element quality. However, this step is optional. In the following subsections, we provide more details on the individual steps.

3.1. Compute Target Surface Ω_t

Two different scenarios are considered: 1) the input has a reference triangle mesh of the boundary, and 2) the input does not have a reference triangle mesh of the boundary. For the former, we first smooth and project the surface vertices to the surface of reference mesh, and then take the smoothed and projected mesh as the target surface Ω_t. For the latter, we consider the boundary of the input mesh as the reference mesh to compute the target surface. We first use a simple Laplacian smoothing to improve the surface (e.g., regulate the boundary quad mesh) of the input hex-mesh. Generally, we perform 20 iterations of smoothing. Smoothing the interior vertices in the volume is optional. We then project the smoothed surface to the reference mesh. To do so, we use a perpendicular ray to project a vertex v to all planes of triangle facets on the reference mesh. Specifically, a quad facet has 4 overlapping triangle facets. If the intersecting point p is inside the triangle (i.e., the u, v parameters of its barycentric coordinates satisfy $u \geq 0, v \geq 0, u + v \leq 1$), we add it to a set S. Finally, we select the intersecting point p that is the closest to v as the projected point. Via this projection, we obtain the target surface Ω_t.



For classifying the boundary vertices, we rely on a user-specified angle threshold θ. If the dihedral angle between two facets sharing a common edge e is smaller than θ, we classify the vertices of e as on the sharp feature L. If a vertex is adjacent to more than 2 sharp edges, then we consider it as a corner C. We mark other surface vertices as regular. During the optimization, a corner could only move within a very small ball, a vertex of sharp edge could move along the feature line, and a regular vertex could move along the tangent plane. See the Eq.(7) for more detailed discussion on how to use this classification.

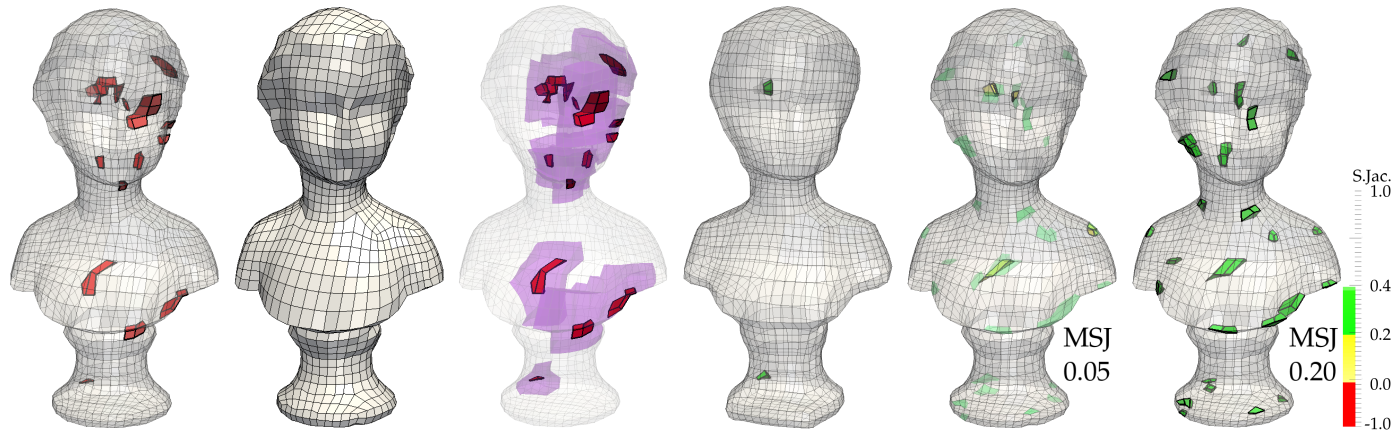
3.2. Untangling

Our untangling process is performed locally. We first detect all the inverted elements based on their scaled Jacobians. We then define a local region surrounding each inverted element. For those inverted elements that form a cluster (i.e., connected with each other), a larger region will be identified. In our implementation, the local region is defined as the union of the two-ring neighborhood surrounding each inverted element. The reason of considering a two-ring neighborhood is that one-ring neighborhood might not provide sufficient information for the subsequent target edge length computation (i.e., Eq. (8)). If the mesh contains a large portion of inverted elements (e.g., the fandisk model in Figure 6), a larger neighborhood will be constructed to enclose these elements. During the optimization, the boundary vertices of this local region are fixed. To untangle the elements within this local region, an energy function is used to compute the distortion of the individual elements from a canonical cube. In general, any proper distortion energy function can be used here, including the edge cone descriptor [8].

However, we opt for an variant of the edge cone descriptor inspired by the recently introduced local frame description [45] due to the following reasons. First, it is intuitive and easy to implement. Second, it will be shown that all different energy terms can be unified under the same representation. In the next, we describe our distortion energy.

3.2.1. Distortion Energy

Given an all-hex mesh $\mathcal{H}$ that contains the sets of vertices $v \in \mathbf{V}$, edges $e \in \mathbf{E}$, facets $f \in \mathbf{F}$ and hexahedral cells $h \in \mathbf{H}$.

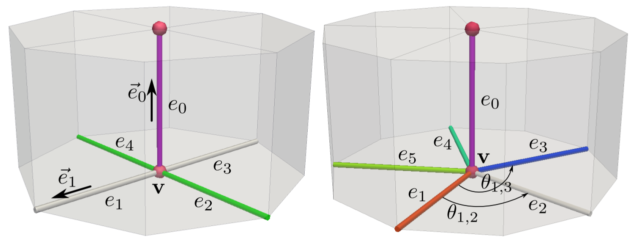


(a) input $\mathcal{H}$ (b) target surface Ω_t (c) local region around inverted elements (d) untangled mesh $\mathcal{H}'$ (e) inversion-free deformation $\hat{\mathcal{H}}$ (f) result

Fig. 1. We optimize a hex-mesh (a) with multiple inverted (red) elements. We first obtain a target surface (b) by aligning boundary vertices to features. We then produce an inversion-free mesh (d) with large surface distance error via optimizing the local regions (c) using a soft constraint on the surface. Next, we use inversion-free deformation to pull the surface of (d) back to the one obtained in (b) and obtain an inversion-free mesh (e). Using a hard constraint on the surface, we further optimize the mesh to improve its MSJ with low surface distance error (f).

Let $\mathbf{v}$ denote the coordinates of vertex v. Our goal is to minimize the following energy.

$$\min_{\mathbf{v}} E(\mathbf{v}) = E_{\mathrm{O}}(\mathbf{v}) + E_{\mathrm{S}}(\mathbf{v}) + E_{\mathrm{R}}(\mathbf{v}) \tag{1}$$



(a) $e_0 \perp e_1 - e_4, e_1 \parallel e_3, e_2 \parallel e_4$ (b) $e_0 \perp e_1 - e_5, e_i \nparallel e_j, i \neq j$

Fig. 2. The relationship of neighboring edges. $\mathbf{e}_0$ in (a) is a regular edge, while $\mathbf{e}_0$ in (b) is irregular. Different colors indicate different parameterization directions.

Orthogonality term. Consider a set of edges e_i adjacent to vertex v, the ideal configuration of two edges that are following two different parameterization directions should be as orthogonal as possible (e.g., edge e_0 versus the other edges as shown in Figure 2(a)). This leads to the orthogonality energy.

$$E_{\mathrm{O}}(\mathbf{v}) = \sum_{e_i \in \mathbf{E}} \sum_{\substack{e_i \cap e_j = \mathbf{v} \\ e_i \perp e_j}} < \frac{\vec{e}_i}{\|\vec{e}_i\|}, \frac{\vec{e}_j}{\|\vec{e}_j\|} >^2 \tag{2}$$

where $e_i \perp e_j$ indicates that the two edges are on two different parameterization directions that are orthogonal to each other. $\vec{e}_i$ and $\vec{e}_j$ are the edge vectors associated with edges e_i and e_j, respectively, pointing outwardly from the center vertex v. That is, $\vec{e}_i = \mathbf{v_i} - \mathbf{v}$.

Straightness term. Similarly, we can define the straightness energy among the connected edges that are following the same parameterization direction as follows.

$$E_{\mathrm{S}}(\mathbf{v}) = \sum_{e_i \in \mathbf{E}} \sum_{\substack{e_i \cap e_j = \mathbf{v} \\ e_i \parallel e_j}} (< \frac{\vec{e}_i}{\|\vec{e}_i\|}, \frac{\vec{e}_j}{\|\vec{e}_j\|} > +1)^2 \tag{3}$$

This energy attempts to make the connected edges that are following the same parameterization direction as straight as possible (e.g., the gray edge pair e_1 and e_3 and the green pair e_2 and e_4 in Figure 2(a)).

Irregular edge term. The above straightness term cannot handle the irregular edges whose values are not 4. Consider e_0 with valence 5 in Figure 2(b). In this case, the orthogonality between e_0 and the rest of the edges around v still holds. However, it is impossible to define the straightness among edges $e_1 - e_5$ due to the irregularity. To address that, we define an energy as the difference between their pairwise angles and their respective ideal angles.

$$E_{\mathrm{R}}(\mathbf{v}) = \sum_{e_i \in \mathbf{E}} \sum_{\substack{e_i \cap e_j = \mathbf{v} \\ e_i \not\perp e_j \\ e_i \nparallel e_j}} (< \frac{\vec{e}_i}{\|\vec{e}_i\|}, \frac{\vec{e}_j}{\|\vec{e}_j\|} > -\hat{a})^2 \tag{4}$$

where $\hat{a} = cos\hat{\theta}_{ij}$, $\hat{\theta}_{ij}$ is the ideal target angle between edge vectors $\vec{e}_i$ and $\vec{e}_j$. For instance, the ideal angle between edges e_1 and e_2 is $\frac{2\pi}{5}$, while the ideal angle is $\frac{4\pi}{5}$ between e_1 and e_3 in Figure 2(b).

Unified energy. In fact, all the above energy terms can be defined as the difference of the angles between pairs of edges from their respective ideal angles. This leads to the following unified expression of all above energy terms

$$\tilde{E}(\mathbf{v}) = \sum_{e_i \in \mathbf{E}} \sum_{e_i \cap e_j = \mathbf{v}} (< \frac{\vec{e}_i}{\|\vec{e}_i\|}, \frac{\vec{e}_j}{\|\vec{e}_j\|} > -\hat{a})^2 \tag{5}$$

where $\hat{a} = cos\theta_{ij}$, θ_{ij} is the ideal target angle between edge vectors $\vec{e}_i$ and $\vec{e}_j$. If e_i and e_j are following two orthogonal parameterization directions, their ideal angle is $\pi/2$, thus $\hat{a} = 0$; if e_i and e_j follow the same parameterization direction, their ideal angle is π, thus $\hat{a} = -1$; if e_i and e_j are edges around an irregular edge (e.g., Figure 2(b)), their ideal angle is $(k+1)\frac{2\pi}{n}$ where n is the valence of the irregular edge and k is the number of edges between e_i and e_j when traversing from e_i to e_j.

In fact, optimizing this angle based distortion energy function is equivalent to optimizing the cone descriptor with the advantage of no need to estimate the valid normal direction for each cone. That is, if all angles around a mesh edge achieve their respective ideal angles, the associated tetrahedra around this edge also have optimal configuration as indicated in Figure 2.

Boundary Handling. To achieve surface conformity, we use the same strategy introduced in [8] that allows the boundary vertices move along the surface. Specifically, the boundary vertices are constrained to stay on their respective tangent planes, feature lines, or corners, based on their classification:

$$\begin{aligned} E_{\mathbf{B}}(\mathbf{v}) &= \sum_{v\in\mathrm{S}} \beta\|\vec{n}\cdot(\mathbf{v}-\bar{\mathbf{v}})\|^2 \\ &+ \sum_{v\in\mathrm{L}} (\alpha\|\mathbf{v}-\bar{\mathbf{v}}-a\vec{t}\|^2 + a^2) \\ &+ \sum_{v\in\mathrm{C}} \alpha\|\mathbf{v}-\bar{\mathbf{v}}\|^2 \end{aligned} \tag{6}$$

Here $\bar{\mathbf{v}}$ is the reference (or closest) surface position for each vertex v, $\mathbf{v}$ is the current position of v, $\vec{n}$ is the surface normal at position $\bar{\mathbf{v}}$, $\vec{t}$ is the feature tangent at $\bar{\mathbf{v}}$, and a is an auxiliary variable added to the system to enable feature constraints. α and β are two coefficients that are used to control how strong the boundary constraint is. The larger these two coefficients, the more penalty will be applied to vertices that leave the target surface. In default, we set $\alpha = \beta = 1000$ for all our experiments. During the untangling process, these two coefficients will be updated according to the outcome of the preceding iteration.

Combined energy. By combining the above energy defined in the interior and on boundary of the volume, respectively, we solve for the following optimization problem:

$$\min_{\mathbf{v}} \mathcal{E}(\mathbf{v}) = E_{\mathbf{B}}(\mathbf{v}) + \tilde{E}(\mathbf{v}) \tag{7}$$

3.2.2. Numerical Solution

Equation (7) is not a quadratic function, which means that it is impractical to solve it directly. If we use the nonlinear solver, it will converge at a very slow speed. To address this, we use a local-global like scheme, in which we use the local (or current) values for some variables. Specifically, in the local step, we fix $\|\vec{e}_i\|$, $\|\vec{e}_j\|$and $\vec{e}_j$ in Equation (5) (i.e., they are treated as constant with their current values). Also, to determine whether a uniform-size element is enforced or not, we use $\xi * \|\tilde{e}\|$ as the target length for edge e if $\|\vec{e}\| \le \xi * \|\tilde{e}\|$ (otherwise, $\|\vec{e}\|$ is used). ξ is a user-input parameter and $\|\tilde{e}\|$ is the average surface edge length. In our experiments, we use $\xi \in [0.2, 0.6]$. A detailed discussion on the effect of ξ is provided in Section 3.5.

Using this method we can construct an over-determined linear system $\mathbf{Ax} = \mathbf{b}$. To minimize the energy (7), we iteratively solve the linear equation $\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b}$. The solver is terminated once it achieves the target MSJ (e.g. > 0 for the untangling).

To accelerate the above computation, we use the target length $\|\hat{e}\|$ for each edge e in the first iteration. The target length can be computed by minimizing the following quadratic energy.

$$E_{\text{Regularization}} = \sum_{e_i\in\mathbf{E}} \sum_{\substack{e_i\|e_j \\ e_i\cap e_j=\mathbf{v}}} (\|\hat{e}_i\| - \|\hat{e}_j\|)^2 + \sum_{e_i\in\mathbf{E}} \sum_{\substack{e_i\|e_j \\ e_i\cap e_j=\emptyset \\ e_i\cup e_j\in h}} (\|\hat{e}_i\| - \|\hat{e}_j\|)^2 \tag{8}$$

The solution of $\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b}$ is an approximate solution. To avoid overshooting, we decrease the step size $0 < \tau < 1$ linearly for each iteration to update the locations of the interior vertices gradually.

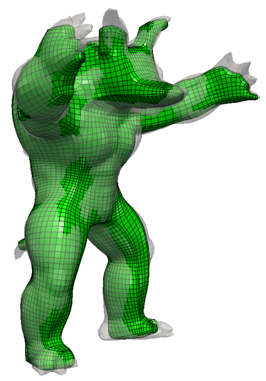
$$v = (1-\tau)\mathbf{v}_{\text{current}} + \tau\mathbf{v}_{\text{solution}} \tag{9}$$

3.2.3. Untangling Pipeline

We now describe our untangling process. Given an input hex-mesh $\mathcal{H}$, we first scale its size *w.r.t.* its center $\frac{1}{n}\sum_{i=0}^{n-1} v_i$ so that its average edge length equals to l (we set $l = 0.025$ for all our experiments). This rescaling step is crucial, which enables us to use the same default α and β values for all different models. Otherwise, different values need to be selected based on the element size of the input mesh. During the scaling, not only $\mathcal{H}$ needs to be scaled, its target surface Ω_t has to be scaled to ensure the consistent boundary constraint for the boundary handling. After this normalization, we then identify all inverted elements and construct a local region for each of them. For all these local regions, we perform the following iterative process until $\mathcal{H}$ is untangled: we first compute target edge lengths in these regions by solving Eq. (8), then set initial step size for updating the vertex positions $\tau = 1$. Next, we iteratively optimize vertex positions by solving Eq. (7) using the aforementioned local-global strategy until the maximal allowed iterations (20 by default) are reached. For each iteration, we check whether the number of inverted elements is reduced within a region. If not (likely due to the overshooting), the solution of this iteration is discarded and τ is reduced. This process guarantees that the number of inverted elements is monotonically reduced. After locally optimizing the vertices within the region, if the outcome mesh $\mathcal{H}'$ still contains inverted elements, we then decrease ξ so that the uniform-size is not enforced. If ξ is too small (e.g., ≤ 0.2 in our implementation), we decrease α and β by half and repeat the above process. Algorithm 1 provides the pseudo-code of this untangling process. After optimizing the mesh, we scale it back to its original space.

3.3. Inversion-free Volume Deformation

After the above untangling process with surface relaxation, the surface of the output untangled mesh $\mathcal{H}'$ may be far away from the target surface Ω_t (see the inset). Previous methods simply project this deformed surface onto Ω_t.



This simple projection does not guarantee that the obtained mesh is still inversion-free. To address that, we formulate the above problem as a volumetric mapping problem $g_t : \mathcal{H}' \rightarrow \hat{\mathcal{H}}$, where Ω_t is the boundary of $\hat{\mathcal{H}}$. A number of inversion-free local injective mapping techniques [41, 42] can be used to achieve the above deformation. In this work, we select the recently introduced SLIM solver [46]. To utilize the SLIM solver, we decompose each hexahedron of $\mathcal{H}'$ into eight tetrahedra (i.e., one tet for each corner). The AMIPS exponential energy [42] is used in our experiments, and 20 iterations are performed.

Algorithm 1: Local untangle

```
Input: $\mathcal{H}, \Omega_t$
Output: $\mathcal{H}'$
Scale $\mathcal{H}$ and $\Omega_t$ ;
Set $\alpha = 1000, \beta = 1000, \xi = 0.6$ ;
while current MSJ $\leq 0$ do
    while not reach maximum global iteration (default 20) do
        Identify inverted elements $\mathcal{I}$;
        Extract local regions $\mathcal{R}$ (a copy from $\mathcal{H}$);
        Classify surface vertices for $\mathcal{R}$;
        Compute target edge length by solving Eq. (8) for $\mathcal{R}$;
        $\tau = 1$;
        while not reach maximum local iteration (default 20) do
            $\tau = 0.9\tau$;
            Solve Eq. (7) for $\mathcal{R}$;
            Save $\mathcal{R}$;
            $\mathcal{R} \leftarrow$ Update vertices. using Eq (9);
            Project surface vertices of $\mathcal{R}$ to its original surface;
            if #invertedElements increased then
                Recover the saved $\mathcal{R}$;
        end
        Update the vertices of $\mathcal{R}$;
        if current MSJ $> 0$ then
            output $\mathcal{H}'$ ;
    end
    $\xi = \xi - 0.1$ ;
    if $\xi < 0.2$ then
        $\alpha = 0.5 \times \alpha, \beta = 0.5 \times \beta, \xi = 0.6$ ;
end
```

3.4. Improving MSJ

Similar to the above untangling process. The improvement of the MSJ can be performed locally. In fact, the same process to the above untangling can be employed with only the modification of the target MSJ MSJ_t, which is specified by the user. Given this target MSJ, the optimizer will first identify the elements whose scaled Jacobian is smaller than MSJ_t. A local region is then constructed for each identified element, which will be used to perform the local improvement. In all our experiments of improving MSJ, we avoid using $\alpha, \beta < 500$ to control distance error. In fact, most of the time we can achieve the target MSJ using $\alpha, \beta = 1000$. The bigger α, β are, the stronger the surface constraint is. In practice, if a larger MSJ_t is set (e.g., > 0.5), the optimizer will take longer time to converge. Sometime, it may not even find a solution. Therefore, we suggest to achieve this MSJ_t procedurally. That is, we optimize the mesh so that its MSJ is positive, then 0.1, 0.2, ..., until it reaches a value above or close to MSJ_t. This procedural strategy is shown very effective in practice (Figure 10).

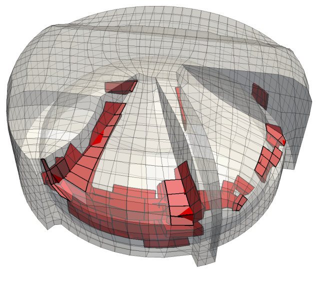
3.5. Discussion on User Parameters

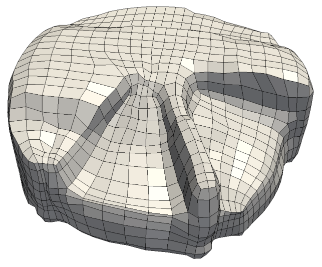
Our approach allows four user-input parameters: (1) target minimum scaled Jacobian MSJ_t, (2) surface constraint α, β, (3) angle threshold θ for sharp feature and corner identification, and (4) edge length constraint ξ that controls whether a uniform-size hex-mesh is preferred.

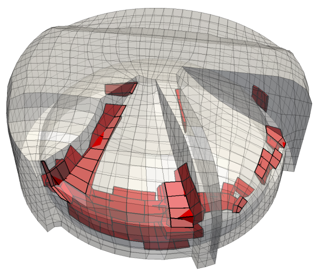
Effects of different α and β Figure 3 shows the untangling results with different values of α and β. Figure 3(a) shows the results with $\alpha = \beta = 1000$. The output mesh has small surface distance from the original surface. However, the untangler fails to correct all inverted elements (see the red elements). Figure 3(b) is the result of the same input mesh with $\alpha = \beta = 100$. Note that the untangler successfully corrects all inverted elements. However, the surface distance from the original surface is larger than the one shown in Figure 3(a). Generally, larger α, β result in smaller distance error but lower MSJ; in the opposite, smaller α, β lead to larger distance error but higher MSJ. In our untangling process and MSJ improvement, the values of α and β are automatically adjusted to find a desired solution. For a large user-specified target MSJ MSJ_t, due to the configurations of the individual surfaces, smaller α and β may be used to achieve MSJ_t, which may lead to large surface error. Although an inversion-free deformation can be applied to reduce the surface error, it may worsen the MSJ at the same time. Therefore, in our experiments, we do not allow the values of α and β to be smaller than 500 during the improvement of MSJ, which also ensures a small surface distance error. However, the user may choose to lower the values of α and β to achieve even better MSJ with the possible larger surface error.

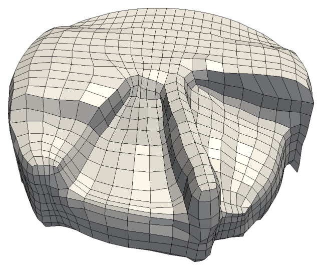
Effects of different θ. Parameter θ is used to control the extraction of surface features. Figure 3(e) and 3(f) show the effect of different θ. In general, the larger θ is, the more surface features will be detected, thus more constraints will be applied to the surface vertices. In practice, we set $\theta = 165°$. Nonetheless, the accurate detection of surface sharp features is non-trivial and tends to be very sensitive to noise. Addressing this is beyond the scope of this work.

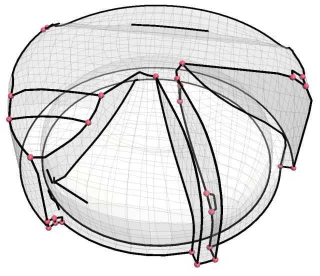
Effects of different ξ. As briefly mentioned earlier, parameter ξ is used to control whether a mesh with uniform-size elements (i.e., with constant edge length) is desired or not. In particular, the larger ξ is the stronger the constraint on uniform-size elements. For instance, in Figure 3(b) and 3(c), both α and β are set as 100, while ξ is 0.4 in 3(c) and 0.2 in 3(b). For the $\xi = 0.4$, the untangler fails to correct all inverted elements. This shows that enabling some variation in the element size will in fact help enhance the success rate of untangling. Similarly, in the MSJ improvement, a larger ξ will constraint the optimizer from finding a good solution (Figure 4(b) and 4(c)).


(a) $\alpha, \beta = 1000, \xi = 0.2, \theta = 165°$


(b) $\alpha, \beta = 100, \xi = 0.2, \theta = 165°$


(c) $\alpha, \beta = 100, \xi = 0.4, \theta = 165°$


(d) $\alpha, \beta = 100, \xi = 0.4, \theta = 150°$


(e) Vertices classification, $\theta = 165°$

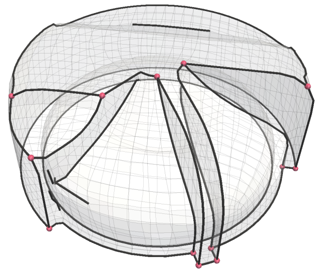
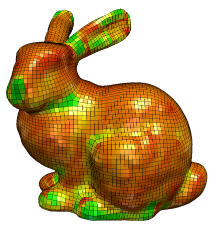

(f) Vertices classification, $\theta = 150°$

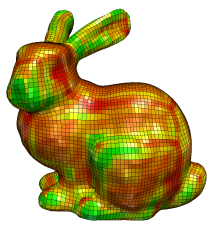
Fig. 3. (a) The output mesh has small surface distance from the original surface with $\alpha = \beta = 1000$. However, the untangler fails to correct all inverted elements (red). (b) Untangler successfully corrects all inverted elements. However, the surface distance from the original surface is larger than the one in (a). (c) Larger ξ greatly impacts the untangler for this model (cap) even using very small α and β. However, using a more relaxed θ helps untangling (d). (e) and (f) show the detected surface features (in black) with different θ values. Small θ results in less sharp feature lines and corners

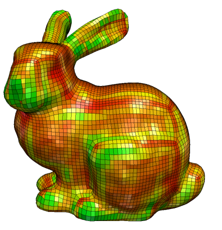
Note that among the above four parameters, the default values for α, β and ξ (i.e. 1000, 1000, and 0.6) usually work well for the majority of the models, thus need not be adjusted. However, in some cases, α, β and ξ still need careful selection in order to produce an ideal result, which we will show next.

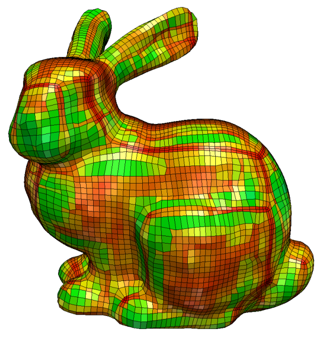
4. Results

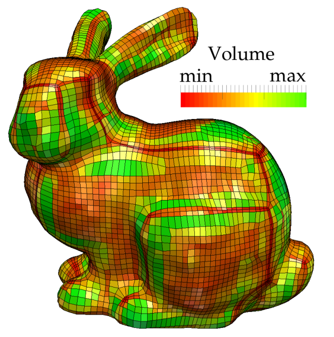
We have applied our untangling and MSJ improvement technique to a number of hex-meshes produced by a variety of


(a) $MSJ = 8e^{-5}$


(b) $\xi = 0.8$, failed


(c) $\xi = 0.6$, failed


(d) $\xi = 0.4$, MSJ = 0.301



(e) $\xi = 0.2$, MSJ = 0.310

Fig. 4. This experiment fixes the parameters $\alpha = \beta = 1000, \theta = 160°$, $targetMSJ = 0.3$ and make ξ varying. Large step of target MSJ or larger ξ will make improvement fail. (b) fails with 315 inverted elements while (c) fails with 275 elements. The volume of each element is mapped to red-to-green to show the effect of using different ξ. The more constant the color, the more uniform sized the elements are.

methods. Figure 9 provides the gallery view of our results. Note that the color coding is based on the volumes of the individual elements in the output mesh. The more constant the color, the more uniform sized the elements are. The statistics of our results is reported in Tables 1, 2, 3 and 4, respectively. Specifically, we use the exact values of the parameters (e.g., $\alpha, \beta, \xi, \theta$ and iterations) as described in Algorithm 1 to generate all the results shown in Tables 2, 3 and 4, as well as for all the octree-based meshes. However, these values may not be the optimal ones for other meshes (e.g., the meshes in Tables 1). For the meshes in Tables 1, we produce the results by customizing the values of those parameters (see the scripts provided as the supplemental material).

Comparison with the edge-cone technique. We apply our method to optimize the dataset provided by the authors of the edge cone technique [8]. Table 1 shows the comparison of the two methods, where the results of our method are highlighted with $*$. From the comparison, we see that our method produces meshes with better MSJ in all cases. We also achieve better surface errors for the majority of the meshes. However, the average scaled Jacobians of our meshes are generally not as good as the edge cone technique. This is mostly because we allow the variation of the element sizes to focus on the MSJ improvement.

Comparison with the AMIPS. Fu et al. [42] applied their local injective mapping technique to further optimize a couple hex-meshes that already have high quality. We apply our method and the edge cone technique to optimize these meshes, respectively. Table 2 compares the results of the three methods. From this comparison, we see that our method is superior in improving the MSJ of these meshes. We also achieve the best ASJ for the RockerArm mesh.

Table 1. Comparison with the edge cone technique. † denotes results of edge-cones [8], * denotes ours. We use metro tools to compute Hausdorff distance w.r.t. bounding box diagonal [47].

Model	#hexes	#flip	Error	MSJ	ASJ
Armadillo†	29935	323	0.011262	0.14	0.9
Armadillo*	29935	323	0.019349	**0.163**	0.834
Block†	2520	31	0.004555	0.250	0.870
Block*	2520	31	**0.002737**	**0.252**	0.857
Bunny†	37734	1	0.007955	0.606	0.972
Bunny*	37734	1	**0.006477**	**0.651**	0.953
Bust†	5258	30	0.007404	0.114	0.922
Bust(Fig1)*	5258	30	0.008843	**0.201**	0.869
Bust*	5258	30	**0.006795**	**0.183**	0.865
Cap†	4420	50	0.009933	0.106	0.870
Cap*	4420	50	**0.005320**	**0.114**	0.775
Dancing†	35293	5	0.008480	0.354	0.942
Children*	35293	5	**0.006905**	**0.582**	0.931
Hanger†	4539	3930	0.002709	0.716	0.987
Hanger*	4539	3930	**0.001486**	**0.723**	0.974
Impeller†	11174	8857	0.000935	0.184	0.942
Impeller*	11174	8857	**0.000493**	**0.192**	0.934
KingKong†	159488	11	0.010483	0.268	0.967
KingKong*	159488	11	0.016948	**0.500**	0.954

Table 2. Comparison with AMIPS and edge cone. The first row of each model shows the result by AMIPS [42]. † denotes results of edge-cone [8], * denotes ours.

Model	#hexes	input MSJ	MSJ	ASJ
Fertility	10600	0.209	0.46	0.937
Fertility†	10600	0.209	0.478	0.951
Fertility*	10600	0.209	**0.602**	0.933
RockerArm	19870	0.196	0.550	0.923
RockerArm†	19870	0.196	0.556	0.937
RockerArm*	19870	0.196	**0.700**	**0.939**

Improve hex-meshes from polycube map. Next, we apply our technique to improve tens of hex-meshes generated from the polycube map database by Fu et al. [49]. The initial hex-meshes consist of varying numbers of inverted elements. Our technique successfully untangled all of these meshes and managed to improve their MSJ substantially. As a comparison, we also show the results of the improved elephant and bottle1 meshes produced by the authors of the edge-cone technique. The comparison shows that our method produces much higher-quality meshes for these two cases in all quality metrics. Note that the surface error of the elephant mesh produced by the edge-cone method is not measurable as the resulting mesh does not have the same scale as the input mesh. We also note that a padding process is applied during the generation of the initial polycube hex-meshes to push the surface singularities into volume. This ensures that degenerate cases (i.e., corner is located on the flat region of the surface) do not occur; otherwise, the meshes may not be able to untangle as already shown by Livesu et al. [8].

Table 3. Stress Test. Fandisk is created by the frame field method, Kitty is generated by the $\ell_1 - PolyCube$ [23], and airplane is obtained using MeshGems [48]. #flip shows the number of inverted elements after the artificial perturbation. * denotes our results. We use metro tools to compute Hausdorff distance w.r.t. bounding box diagonal [47]. '–' means the mesh has no reference surface to compute the Hausdorff distance error.

Model	#hexes	#flip	Error	MSJ	ASJ
Fandisk	357	0	–	0.609	0.936
Fandisk*	357	286	0.004769	**0.752**	**0.950**
Kitty	7083	0	–	0.424	0.910
Kitty*	7083	3232	0.012656	**0.652**	**0.937**
airplane1	4972	0	–	0.030	0.838
airplane1*	4972	3510	0.004369	**0.503**	**0.875**

Table 4. The results on a set of meshes produced from the polycube map database [49]. † denotes results of edge-cone [8], * denotes ours. We use metro tools to compute Hausdorff distance wrt. bounding box diagonal [47]. '–' means the mesh's surface is not in the same scale with the input for computing the Hausdorff distance error.

Model	#hexes	#flip	Error	MSJ	ASJ
airplane1*	17913	467	0.006257	0.731	0.959
bird*	16934	288	0.005774	0.732	0.961
cup1*	16862	40	0.006944	0.723	0.960
chair1*	20344	709	0.004720	0.690	0.941
horse*	44145	304	0.017018	0.600	0.944
blade*	14792	141	0.008885	0.650	0.946
kiss*	19976	247	0.014755	0.500	0.913
bottle1†	15478	127	0.008066	0.132	0.925
bottle1*	15478	127	0.009675	**0.604**	**0.948**
elephant†	46525	421	–	0.012	0.881
elephant*	46525	421	0.009899	**0.500**	**0.915**

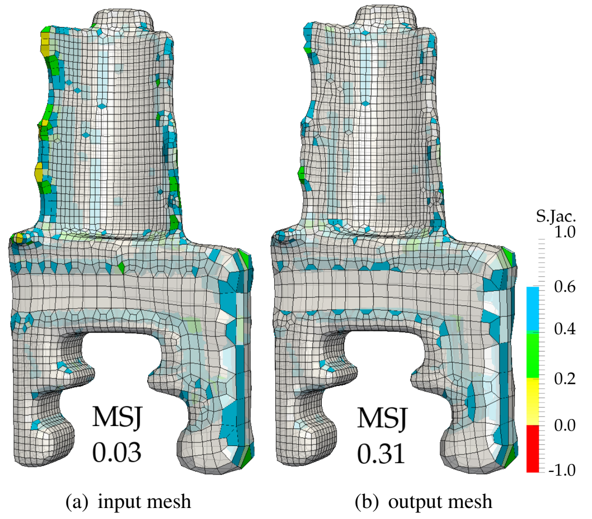
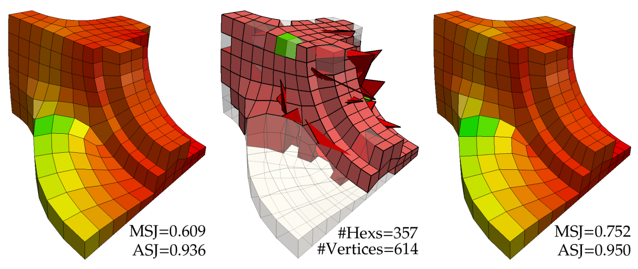


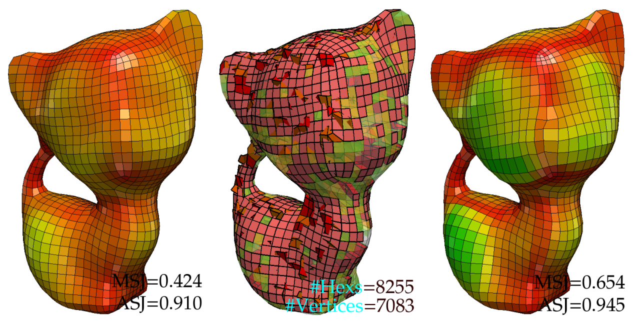
Fig. 5. Blade example(created by the Octree-based method)

Improve hex-meshes generated by the octree-based method. We also apply our technique to improve hex-meshes generated using the MeshGems [48]–an octree-based method [19, 50]. The initial hex-meshes consist of elements with very low scaled Jacobian (< 0.1). For most models, our method can improve their MSJ to be greater than 0.2. Figure 5 and Table 5 provide examples of the improvement of octree-based meshes. More details are in the supplementary document.



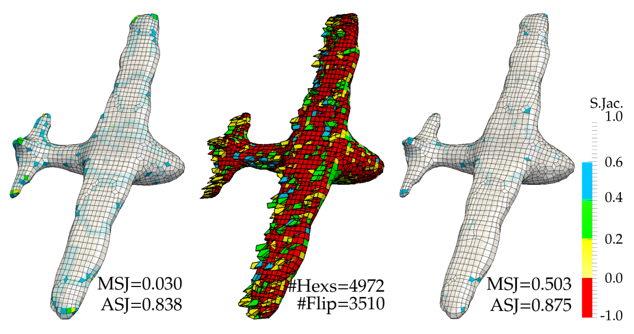
(a) origin mesh (b) stress test input (c) stress test output

Fig. 6. Fandisk example(created by the framed field method)



(a) origin mesh (b) stress test input (c) stress test output

Fig. 7. Kitty example(created by the ℓ_1–PolyCube method)



(a) origin mesh (b) stress test input (c) stress test output

Fig. 8. airplane1 example(created by the octree-based method)

Stress test. In the stress test experiments, we untangle hexahedral meshes perturbed from the initial hex-meshes generated by the frame-field based, polycube based and octree-based methods, respectively. Our method successfully untangles the perturbed meshes and produces meshes with much better quality than the original ones (Figures 6, 7 and 8). Despite the meshes in our stress test contain large portions of inverted elements, they are generated with artificial perturbation. In the future, we plan to further assess our optimization technique with meshes containing large numbers of inverted elements in practice.

Table 5. The results on a set of meshes produced from MeshGems [48]. We use metro tools to compute Hausdorff distance w.r.t. bounding box diagonal [47].'–' means the there is no reference surface to compute the Hausdorff distance error. ^{i} and ^{o} show the input and output, respectively

Model	#hexes	Error	MSJ	ASJ
bird^i	4247	–	0.0313	0.820
bird^o	4247	0.011580	0.553	0.868
blade^i	10996	–	0.025	0.845
blade^o	10996	0.007911	0.312	0.868
block^i	1624	–	0.179	0.661
block^o	1624	0.016877	0.550	0.815
bone^i	2751	–	0.154	0.781
bone^o	2751	0.006475	0.207	0.794
dragonstand2^i	23917	–	0.013	0.837
dragonstand2^o	23917	0.004598	0.304	0.857
fish1^i	9537	–	0.015	0.815
fish1^o	9537	0.008377	0.308	0.845
gargoyle^i	41610	–	0.024	0.834
gargoyle^o	41610	0.005247	0.200	0.849
kiss^i	18418	–	0.027	0.844
kiss^o	18418	0.005075	0.224	0.857
rocker^i	16608	–	0.108	0.865
rocker^o	16608	0.007742	0.241	0.874

Performance study. As mentioned in Section 3.4, our improvement of MSJ is performed in several stages. The benefit of this divide-and-conquer strategy is that the number of elements that have quality lower than the current target MSJ remains small each step, which facilitates our optimizer to quickly find a solution. Figure 10 shows a timing plot of this gradual improvement process. The times spent on the individual stages are shown as the histogram, and the orange curve shows the accumulated time. As expected, more time will be needed to achieve a higher MSJ as more elements will have quality lower than the target MSJ. In general, our method takes about 2 minutes on average to process a mesh with $10-20K$ elements. The smaller the MSJ, the faster the computation will be as already shown in Figure 10.

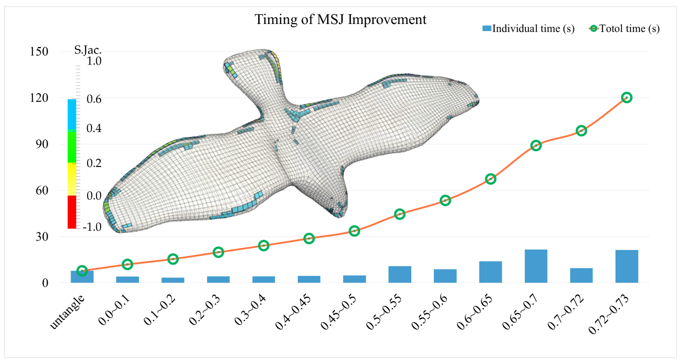


Fig. 10. Performance plot of our technique.

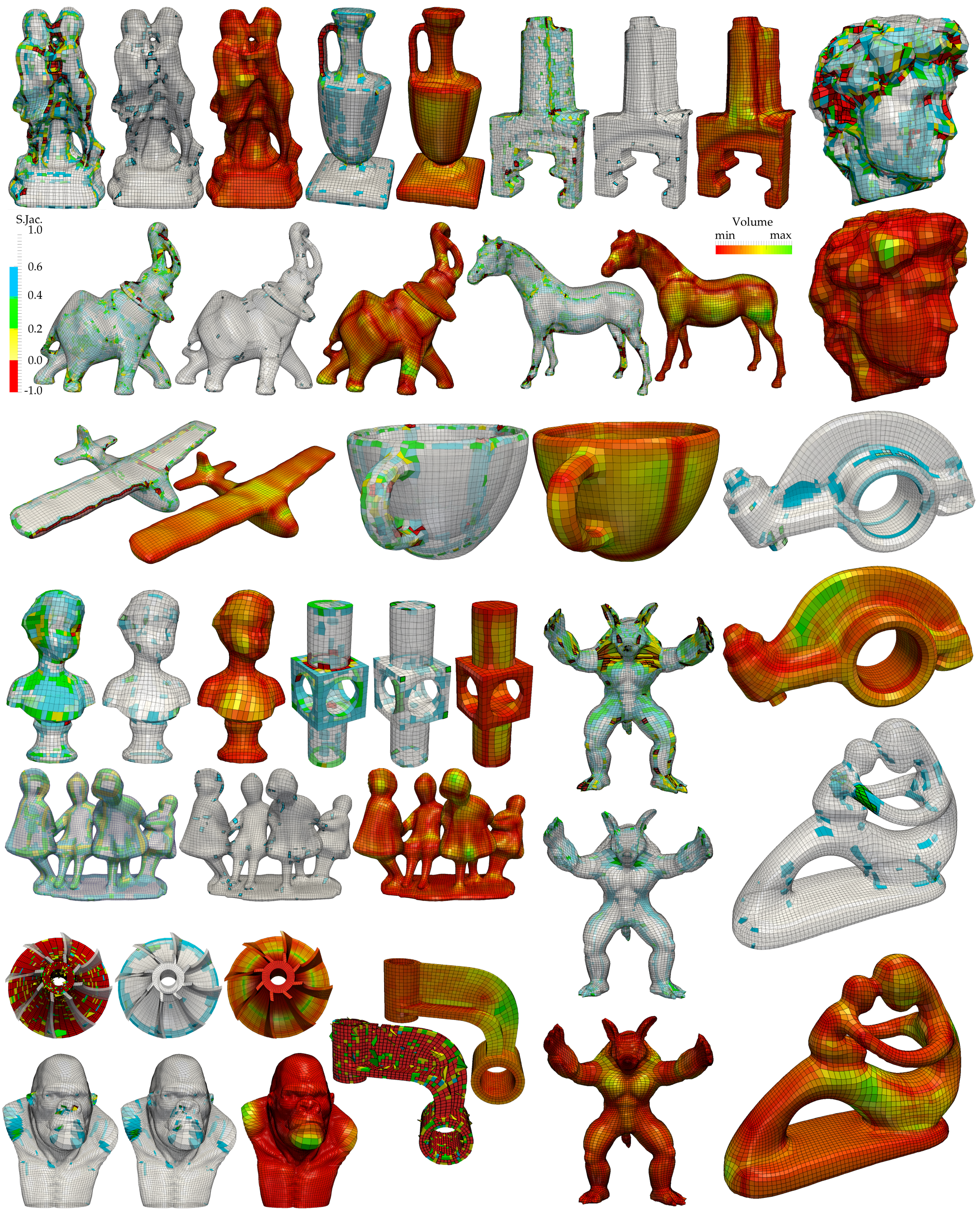


Fig. 9. Result gallery. Elements with scaled Jacobian $SJ > 0.6$ are transparent in the input (first figure for each model). Output meshes are colored using element volume info (last figure for each model). Elements with $SJ < 0.6$ are also showed in the output (e.g., the middle image for each model). We hope to further improve these elements in the future work. If all elements of a mesh have $SJ > 0.6$, we do not show its middle image (e.g., the airplane and cup models).

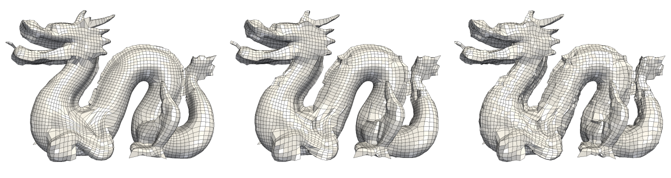

(a) input mesh (b) untangled mesh (c) SLIM deformation

Fig. 11. SLIM fails to map back the surface of dragon.

Failure case. Figure 11 shows a failure case where the SLIM solver fails to map the deformed surface after MSJ improvement back to the target surface. This may be an issue of the SLIM solver that typically requires an accurate correspondence between the boundary vertices of the current mesh with those on the target surface, while our current projection based method may not be sufficiently accurate. In the future, we plan to experiment with other more robust inversion-free mapping technique and improve our surface correspondence calculation.

5. Conclusion

In this paper, we introduce a simple yet effective hex-mesh improvement technique. This technique is based on a new and intuitive angle based optimization strategy. To enable our optimizer to find a valid solution, we allow the boundary surface to move out from the original volume, which will be mapped to the original surface with the inversion-free guarantee. To accelerate the computation, we perform the optimization within a local region surrounding the inverted elements or elements with quality lower than the user-specified threshold. Our method is easy to implement. We also discuss the effects of the different values of a number of parameters used in our framework to help users choose proper values for their needs. We have applied our method to a large number of hex-meshes generated with a variety of methods to demonstrate its effectiveness.

Limitations.. First, although our method produces meshes with higher MSJ for all the test meshes and better Hausdorff distance for the majority of meshes when compared to the state-of-the-art techniques, our method may not improve the average scaled Jacobian substantially. Again, this is due to the relaxation of the constraint on uniform element sizes. Also, our sub-optimal ASJ may also attribute to the selection of the parameter ξ. For most models, we find that $\xi = 0.2-0.6$ can produce ideal results. But for some models (e.g. Hanger), the result using $\xi = 1.2$ is better than the one obtained with other values of ξ. Nonetheless, the fixed value of ξ throughout the entire mesh may constrain the improvement of ASJ. Should the ξ of an edge be a function with respect to its neighboring configuration, ASJ might be improved further. Second, to ensure an inversion-free outcome, the meshes generated with our method may have a surface distance larger than the user-specified error. Third, our current surface feature detection is sensitive to the user-specified angle threshold θ. A robust feature detection technique may be required to resolve this issue. Fourth, in the extreme case (i.e., a complete inverted mesh), our angle based energy will vanish. However, since we enforce the boundary constraint of non-inverted elements, such an extreme case will not occur. Finally, our method for solving the non-linear energy minimization problem is not a typical local-global scheme, which may not converge to meet the required mesh quality. However, it enables us to effectively minimize our angle distortion energy. We plan to address these limitations in the future.

Acknowledgement

We would like to thank Marco Livesu for generating the edge-cone results for the comparison and all the anonymous reviewers for their valuable comments. This work is partially supported by NSF IIS-1553329.

References

[1] Pébay, PP, Thompson, D, Shepherd, J, Knupp, P, Lisle, C, Magnotta, VA, et al. New Applications of the Verdict Library for Standardized Mesh Verification Pre, Post, and End-to-End Processing. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-540-75103-8; 2008, p. 535–552.

[2] Knupp, PM. Hexahedral mesh untangling and algebraic mesh quality metrics. In: Proceedings, 9th International Meshing Roundtable. 2000, p. 173–183.

[3] Knupp, PM. A method for hexahedral mesh shape optimization. International journal for numerical methods in engineering 2003;58(2):319–332.

[4] Brewer, M, Diachin, LF, Knupp, P, Leurent, T, Melander, D. The mesquite mesh quality improvement toolkit. In: Proceedings of International Meshing Roundtable. 2003,.

[5] Wilson, TJ, Sarrate Ramos, J, Roca Ramón, X, Montenegro Armas, R, Escobar Sánchez, JM. Untangling and smoothing of quadrilateral and hexahedral meshes 2012;.

[6] Ruiz-Gironés, E, Roca, X, Sarrate, J. Optimizing mesh distortion by hierarchical iteration relocation of the nodes on the cad entities. Procedia Engineering 2014;82:101–113.

[7] Ruiz-Gironés, E, Roca, X, Sarrate, J, Montenegro, R, Escobar, JM. Simultaneous untangling and smoothing of quadrilateral and hexahedral meshes using an object-oriented framework. Advances in Engineering Software 2015;80:12–24.

[8] Livesu, M, Sheffer, A, Vining, N, Tarini, M. Practical hex-mesh optimization via edge-cone rectification. Transactions on Graphics (Proc SIGGRAPH 2015) 2015;34(4).

[9] Stimpson, CJ, Ernst, CD, Knupp, P, Â´ebayand, PPP, Thompson, D. The verdict geometric quality library. SANDIA REPORT 2007;.

[10] Owen, SJ. A survey of unstructured mesh generation technology. In: Proceedings of the 7th International Meshing Roundtable. -; 1998, p. 239–267.

[11] Staten, ML, Owen, SJ, Blacker, TD. Unconstrained paving and plastering: A new idea for all hexahedral mesh generation. In: Proc. 14 th Int. Meshing Roundtable. 2005, p. 399–416.

[12] Staten, ML, Kerr, RA, Owen, SJ, Blacker, TD, Stupazzini, M, Shimada, K. Unconstrained plastering hexahedral mesh generation via advancing-front geometry decomposition. International journal for numerical methods in engineering 2010;81(2):135–171.

[13] Schneiders, R. A grid-based algorithm for the generation of hexahedral element meshes. Engineering with computers 1996;12(3-4):168–177.

[14] Zhang, H, Zhao, G. Adaptive hexahedral mesh generation based on local domain curvature and thickness using a modified grid-based method. Finite Elements in Analysis and Design 2007;43(9):691–704.

[15] Edgel, JD. An adaptive grid-based all hexahedral meshing algorithm based on 2-refinement 2010;.

[16] Sun, L, Zhao, G. Adaptive hexahedral mesh generation and quality optimization for solid models with thin features using a grid-based method. Engineering with Computers 2016;32(1):61–84.

[17] Zhang, YJ, Bajaj, C. Adaptive and quality quadrilateral/hexahedral meshing from volumetric data. Computer Methods in Applied Mechanics and Engineering 2006;195(9-12):942–960.

[18] Maréchal, L. Advances in octree-based all-hexahedral mesh generation: handling sharp features. In: proceedings of the 18th International Meshing Roundtable. Springer; 2009, p. 65–84.

[19] Ito, Y, Shih, AM, Soni, BK. Octree-based reasonable-quality hexahedral mesh generation using a new set of refinement templates. Int J Numer Meth Engng 2009;77(13):1809–1833.

[20] Zhang, YJ, Liang, X, Xu, G. A robust 2-refinement algorithm in octree or rhombic dodecahedral tree based all-hexahedral mesh generation. Computer Methods in Applied Mechanics and Engineering 2013;256:88–100.

[21] Gregson, J, Sheffer, A, Zhang, EG. All-hex mesh generation via volumetric polycube deformation. Computer Graphics Forum 2011;30(5):1407–1416.

[22] Livesu, M, Vining, N, Sheffer, A, Gregson, J, Scateni, R. Polycut: Monotone graph-cuts for polycube base-complex construction. Acm Transactions on Graphics 2013;32(6).

[23] Huang, J, Jiang, TF, Shi, ZY, Tong, YY, Bao, HJ, Desbrun, M. l1-based construction of polycube maps from complex shapes. Acm Transactions on Graphics 2014;33(3).

[24] Fang, X, Xu, W, Bao, H, Huang, J. All-hex meshing using closed-form induced polycube. Acm Transactions on Graphics (Proceedings of SIGGRAPH 2016) 2016;35(4).

[25] Huang, J, Tong, Y, Zhou, K, Bao, H, Desbrun, M. Boundary aligned smooth 3d cross-frame field. Acm Transactions on Graphics 2011;30(6):143:1–143:8.

[26] Nieser, M, Reitebuch, U, Polthier, K. Cubecover - parameterization of 3d volumes. Computer Graphics Forum 2011;30(5):1397–1406.

[27] Li, YF, Liu, Y, Xu, WW, Wang, WP, Guo, BN. All-hex meshing using singularity-restricted field. Acm Transactions on Graphics (Proceedings of SIGGRAPH 2012) 2012;31(6).

[28] Jiang, T, Huang, J, Wang, Y, Tong, Y, Bao, H. Frame field singularity correction for automatic hexahedralization. IEEE Trans Vis Comput Graphics 2014;20(8):1189–1199.

[29] Shepherd, JF, Johnson, CR. Hexahedral mesh generation constraints. Eng with Comput 2008;24(3):195–213.

[30] Knupp, PM. Winslow smoothing on two-dimensional unstructured meshes. Engineering with Computers 1999;15(3):263–268.

[31] Leng, J, Xu, G, Zhang, Y, Qian, J. Quality improvement of segmented hexahedral meshes using geometric flows. In: Image-Based Geometric Modeling and Mesh Generation. Springer; 2013, p. 195–221.

[32] Shepherd, JF. Topologic and geometric constraint-based hexahedral mesh generation. Ph.D. thesis; The University of Utah; 2007.

[33] Gao, X, Deng, Z, Chen, G. Hexahedral mesh re-parameterization from aligned base-complex. Acm Transactions on Graphics (Proceedings of SIGGRAPH 2015) 2015;35(4).

[34] Yu, WY, Zhang, K, Wan, SH, Li, X. Optimizing polycube domain construction for hexahedral remeshing. Computer-Aided Design 2014;46:58–68.

[35] Cherchi, G, Livesu, M, Scateni, R. Polycube simplification for coarse layouts of surfaces and volumes. Computer Graphics Forum 2016;35(5):11–20.

[36] Gao, X, Huang, J, Li, S, Deng, Z, Chen, G. An evaluation of the quality of hexahedral meshes via modal analysis. In: 1st Workshop on Structured Meshing: Theory, Applications, and Evaluation. 2014,.

[37] Diachin, LF, Knupp, P, Munson, T, Shontz, S. A comparison of two optimization methods for mesh quality improvement. Engineering with Computers 2006;22(2):61–74.

[38] Sastry, SP, Shontz, SM. A parallel log-barrier method for mesh quality improvement and untangling. Engineering with Computers 2014;30(4):503–515.

[39] Aigerman, N, Lipman, Y. Injective and bounded distortion mappings in 3d. ACM Transactions on Graphics (TOG) 2013;32(4):106.

[40] Schüller, C, Kavan, L, Panozzo, D, Sorkine-Hornung, O. Locally injective mappings. In: Computer Graphics Forum. 5; Wiley Online Library; 2013, p. 125–135.

[41] Fu, XM, Liu, Y. Computing inversion-free mappings by simplex assembly. ACM Trans Graph 2016;35(6):216:1–216:12.

[42] Fu, XM, Liu, Y, Guo, B. Computing locally injective mappings by advanced mips. ACM Trans Graph 2015;34(4):71:1–71:12.

[43] Wilson, TJ. Simultaneous untangling and smoothing of hexahedral meshes. Ph.D. thesis; Universitat Politécnica de Catalunya; 2011.

[44] Sun, L, Zhao, G, Ma, X. Quality improvement methods for hexahedral element meshes adaptively generated using grid-based algorithm. International Journal for Numerical Methods in Engineering 2012;89(6):726–761.

[45] Gao, X, Chen, G. A local frame based hexahedral mesh optimization. In: In 25th International Meshing Roundtable(IMR2016), Research Notes. Elsevier; 2016,.

[46] Rabinovich, M, Poranne, R, Panozzo, D, Sorkine-Hornung, O. Scalable locally injective maps. ETH Technical Report 2016;.

[47] Guthe, M, Borodin, P, Klein, R. Fast and accurate hausdorff distance calculation between meshes. In: In WSCG. 2; 2005, p. 41–48.

[48] MeshGems, . Volume meshing: Meshgems-hexa. 2015.

[49] Fu, X, Bai, C, Liu, Y. Efficient volumetric polycube-map construction. Computer Graphics Forum (Pacific Graphics) 2016;35(7):97 – 106.

[50] Marechal, L. Advances in octree-based allhexahedral mesh generation: handling sharp features. In: Proceedings of International Meshing Roundtable. 2009, p. 65–84.