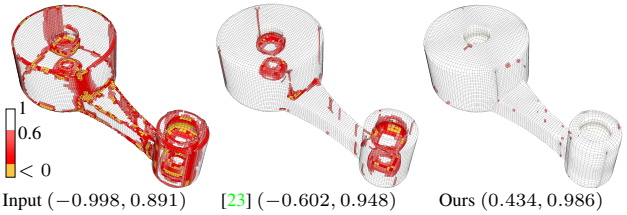
Untangling All-Hex Meshes via Adaptive Boundary Optimization

Qing Huang Wen-Xiang Zhang Qi Wang Ligang Liu
Xiao-Ming Fu
School of Mathematical Sciences, University of Science and Technology of China
Hefei, China 230026
hq803@mail.ustc.edu.cn, zwx111@mail.ustc.edu.cn, wq2014@mail.ustc.edu.cn,
lgliu@ustc.edu.cn, fuxm@ustc.edu.cn

Abstract

We propose a novel method to untangle and optimize all-hex meshes. Central to this algorithm is an adaptive boundary optimization process that significantly improves practical robustness. Given an all-hex mesh with many inverted hexahedral elements, we first optimize a high-quality quad boundary mesh with a small approximation error to the input boundary. Since the boundary constraints limit the optimization space to search for the inversion-free meshes, we then relax the boundary constraints to generate an inversion-free all-hex mesh. We develop an adaptive boundary relaxation algorithm to implicitly restrict the shape difference between the relaxed and input boundaries, thereby facilitating the next step. Finally, an adaptive boundary difference minimization is developed to effectively and efficiently force the distance difference between the relaxed boundary and the optimized boundary of the first step to approach zero while avoiding inverted elements. We demonstrate the efficacy of our algorithm on a data set containing 1004 all-hex meshes. Compared to previous methods, our method achieves higher practical robustness.



Input $(-0.998, 0.891)$ [23] $(-0.602, 0.948)$ Ours $(0.434, 0.986)$

Figure 1. Comparison with [23]. The hexes are color-coded by their scaled Jacobian values using the left-most color bar. The result of [23] generated by the authors' provided implementation contains inverted hexes (mustard yellow), whereas our result is inversion-free. The text below each model indicates the minimum scaled Jacobian $J_{\min}$ and the average scaled Jacobian J_{avg}.

Input (-1.000, 0.843) [44] (-0.930, 0.899) Ours (0.366, 0.966)

Figure 2. Comparison with [44]. We use the implementation provided by the authors to generate the result of [44]. Our method succeeds in removing all inverted hexes, whereas they fail. The text below each model shows $J_{\min}$ and J_{avg}.

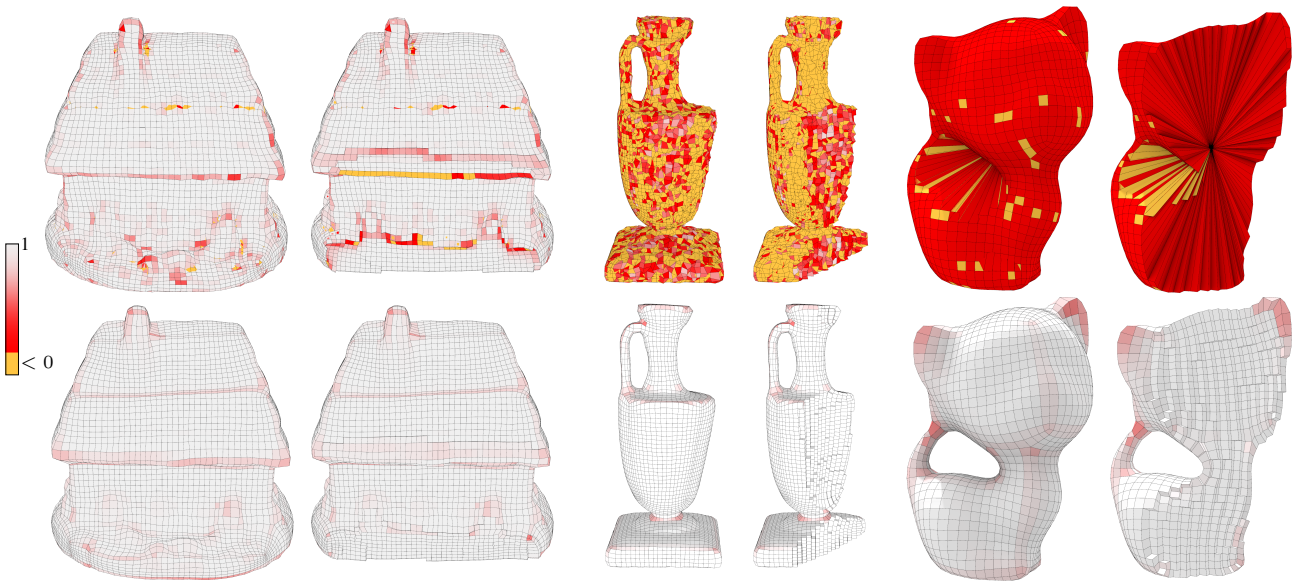
1. Introduction

All-hex meshes are widely used in finite element method to perform physical simulation [2, 33, 24]. Simulation results rely on both average and minimum hexahedral element quality [25]. To avoid numerical instability, at least the elements of all-hex meshes cannot be inverted [8].

The all-hex mesh generation process usually contains two steps: (1) generate an initial mesh whose connectivity is optimized to fit the input mesh; and (2) update the positions of vertices to improve the mesh quality without changing the connectivity. In this paper, we focus on the second step, which is still an open and challenging research problem (Figure 1). In practice, there are two common requirements. First, the resulting all-hex mesh is of high-quality without inverted hexahedral elements. Second, boundary surface preservation is required to obtain a high degree of similarity between the input and the optimized shapes.

These two requirements affect each other. In general, preserving boundary surfaces limits the movement of the boundary vertices, so that the interior vertices cannot be updated freely to obtain inversion-free meshes. To mitigate these mutual influences, the following pipeline is developed [44]. First, an inversion-free high-quality mesh is generated without considering the boundary preservation constraint. Second, the relaxed boundary is pulled back to reduce the distance difference from the input boundary while explicitly keeping high quality and avoiding inverted elements.

It is challenging to design a practically robust algorithm



$J_{\min} = 0.407, J_{\text{avg}} = 0.974$ $J_{\min} = 0.440, J_{\text{avg}} = 0.971$ $J_{\min} = 0.502, J_{\text{avg}} = 0.951$

Figure 3. Given input meshes containing inverted elements (upper row), we untangle them to be inversion-free (bottom row). To test the practical robustness, we place the interior vertices randomly (middle column) and at one position (right column). For one model, we show its boundary surface, one cross section, the minimum scaled Jacobian $J_{\min}$, and the average scaled Jacobian J_{avg} of the output mesh.

based on this pipeline, especially in the second step. The second step relies on two factors. First, as the relaxed shape after the first step is the initial shape of the second step, the distance between the relaxed and the input boundaries should be controlled to facilitate the second step. Second, the optimization methods to effectively generate inversion-free meshes and decrease the boundary distance are desired. To the best of our knowledge, only one former method uses this pipeline [44]; however, it does not carefully and holistically consider these factors. Then, it fails to generate inversion-free results for most models (Figures 2, 15, and 16).

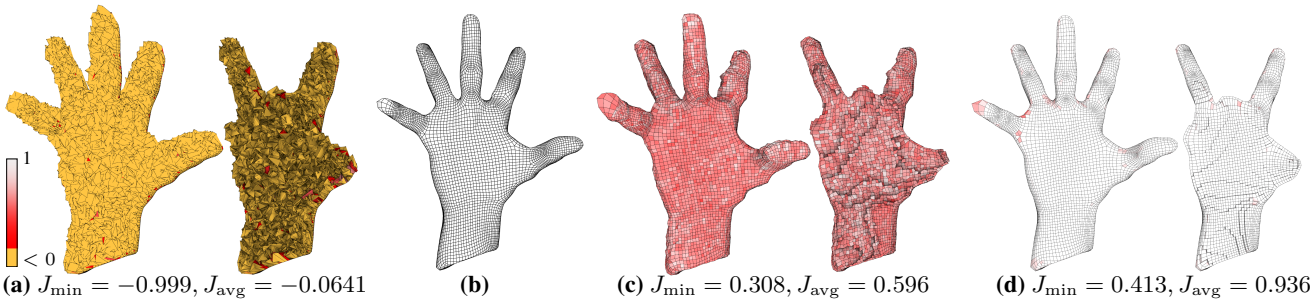
In this paper, we propose a novel method to untangle and optimize all-hex meshes. The algorithm follows the aforementioned pipeline. For the first factor, our first step is an adaptive boundary relaxation procedure that gradually relaxes the boundary constraint to obtain an inversion-free mesh. Since the used optimization solver [35] can eliminate most inverted elements by just moving the interior vertices, only small movements are required for all vertices to obtain an inversion-free mesh, thereby implicitly constraining the distance between the relaxed and the input boundaries. For convenience, we define the distance between boundaries as the sum of squared distance between the corresponding vertices. To achieve a high-quality result, the input boundary quad mesh is optimized to serve as the target of the relaxed boundary in the second step. For the second factor, we adaptively reduce the boundary distance to nearly zero while keeping the mesh always inversion-free based on an elegant second-order solver.

Although we cannot theoretically guarantee inversion-free all-hex meshes in every case, our method has succeeded in producing inversion-free meshes on a data set containing 1004 examples (Figure 3). Compared to existing methods, our method is more practically robust and efficient (Figures 1, 2, 15, and 16).

2. Related Work

All-hex mesh untangling and optimization The most common optimization approach is moving vertices to the weighted average of their neighbors. Geometric flows are also applied to improve all-hex mesh quality [48, 27]. However, there is no guarantee that the result mesh is inversion-free [24]. Several methods [20, 39] start from an inversion-free mesh and relocate vertices while avoiding inversions. In practice, many raw hex-meshing outputs contain inverted elements, limiting the utility of this approach. In addition, keeping the intermediate solutions in the inversion-free space causes that the result is far from the optimal solution.

Many methods first untangle inverted elements, and then improve mesh quality. To correct inverted hex elements, Gauss-Seidel style optimizations are used [16, 20, 43, 29, 30]. Global non-linear optimization methods have also been developed [31, 42]. Other optimization techniques for specific types of hex-meshes also exist, such as the quality improvement method for octree-based hex-meshes [36]. Edge-cone rectification is presented to generate inversion-free elements [23] by minimizing a combined energy, which considers both the boundary surface preservation and inversion-free constraints. These two considerations limit the search space to eliminate the inverted elements (Figure 1). [44] proposes



(a) $J_{\min} = -0.999, J_{\text{avg}} = -0.0641$ **(b)** **(c)** $J_{\min} = 0.308, J_{\text{avg}} = 0.596$ **(d)** $J_{\min} = 0.413, J_{\text{avg}} = 0.936$

Figure 4. Workflow of our method. Given an initial all-hex mesh $\mathcal{M}$ containing a lot of inverted elements (mustard yellow) (a), our method first optimizes the boundary quad surface $\mathcal{B}$ to generate a high-quality quad surface $\mathcal{B}_{\text{opt}}$ (b), then relaxes the boundary constraint to obtain an inversion-free mesh $\mathcal{R}$ (c), and finally minimizes the difference between the boundary of $\mathcal{R}$ and $\mathcal{B}_{\text{opt}}$ to achieve the resulting mesh $\mathcal{N}$ (d). In (a, c, d), we show the boundary surface, one cross section, the minimum scaled Jacobian $J_{\min}$, and the average scaled Jacobian J_{avg}.

an edge-angle optimization method for untangling. Moreover, this method enables the deformation of the boundary surface during the untangling process, which provides more space to gain a valid solution. Our method also allows relaxing boundary surface preservation. However, [44] does not use the relaxation insight carefully and holistically, thereby leading to failure cases (Figure 2).

In addition to moving vertices while fixing connectivity, there are some topological optimization methods to untangle inverted elements and improve mesh quality [40, 41].

Inversion-free volumetric mappings Computing inversion-free volumetric mappings is usually formulated as a non-convex and nonlinear constrained optimization problem (cf. the survey in [7]). Maintenance-based methods start from an inversion-free initialization and minimize mapping distortion while keeping the mapping always staying in the inversion-free space through barrier functions [32, 6, 22] or explicit check [19, 28, 15, 49]. Although these methods guarantee inversion-free, it is very challenging to compute inversion-free initializations in 3D. We first generate an inversion-free initialization and then use this method to reduce the boundary difference and improve mesh quality. Given initializations with inverted elements, many methods try to eliminate the inverted elements, such as bounded distortion mappings [1, 17], projection-based methods [35, 18, 5], area-based methods [3] and penalty-based methods [37, 4, 45, 10]. To generate an inversion-free initialization, we use the projection-based method [35] while relaxing the boundary constraint.

3. Method

3.1. Overview

Input and goal The input is an all-hex mesh $\mathcal{M}$ that contains many inverted hexahedral elements. Its boundary surface is a quad mesh $\mathcal{B}$. Our goal is to untangle the input mesh $\mathcal{M}$ to generate an inversion-free mesh $\mathcal{N}$ that shares the same connectivity with $\mathcal{M}$. The approximation error (e.g., two-sided Hausdorff distance) between the boundaries of $\mathcal{M}$ and $\mathcal{N}$ is small. Namely, we should preserve the boundary surface $\mathcal{B}$ of $\mathcal{M}$ after the mesh optimization.

Methodology The boundary surface preservation requirement and the inversion-free constraint are mutually restrictive, so it is very challenging to generate the desired mesh $\mathcal{N}$. To decouple the mutual influence, we first relax the boundary surface preservation constraint to produce an inversion-free mesh $\mathcal{R}$ whose boundary surface $\mathcal{B}_r$ may have a large distance from $\mathcal{B}$, same as [44]. Then, we deform $\mathcal{R}$ to reduce the approximation error between $\mathcal{B}_r$ and $\mathcal{B}$ while keeping inversion-free.

Workflow According to the methodology above, we propose a novel algorithm containing three steps:

1. We optimize the vertices of $\mathcal{B}$ to produce a new mesh $\mathcal{B}_{\text{opt}}$ that contains high-quality quads and has low approximation error from $\mathcal{B}$.
2. We initialize the interior vertices of $\mathcal{R}$ as those of $\mathcal{M}$ and the boundary vertices of $\mathcal{R}$ as $\mathcal{B}_{\text{opt}}$. Then we optimize $\mathcal{R}$ via an adaptive boundary relaxation procedure that progressively relaxes the boundary preservation constraint (Section 3.2).
3. We generate $\mathcal{N}$ by deforming $\mathcal{R}$ via an adaptive boundary difference minimization process that adaptively enhances the penalization of the difference between $\mathcal{B}_r$ and $\mathcal{B}_{\text{opt}}$ until it is close to zero while avoiding introduction of inverted elements (Section 3.3).

Figure 4 shows an example of our workflow. The adaptive boundary relaxation is developed to control the difference between $\mathcal{B}_r$ and $\mathcal{B}_{\text{opt}}$, thereby reducing the difficulty in minimizing their difference in the last step. Our adaptive boundary difference minimization is to quickly reduce the difference between $\mathcal{B}_r$ and $\mathcal{B}_{\text{opt}}$ to nearly zero.

The optimized quad mesh $\mathcal{B}_{\text{opt}}$ serves as the target boundary of the resulting mesh. Thus, we want to optimize each

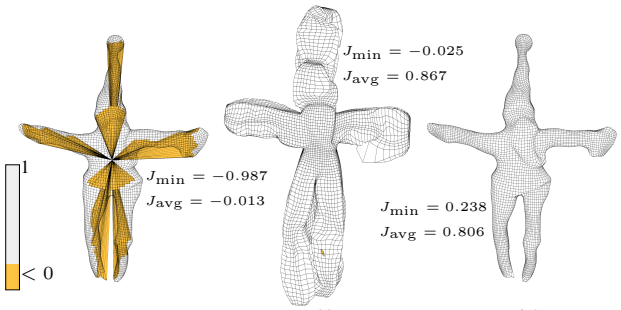


Figure 5. Left: the input mesh, whose interior vertices are at one position. Middle: the relaxation result (2 inverted elements) without adaptive boundary relaxation, i.e., $\lambda = 0$ is fixed. Right: our relaxation result with adaptive boundary relaxation.

quad to be similar to a square while preserving the sharp features of $\mathcal{B}$. This goal has been realized by many previous methods [11, 14, 21, 13], and the elegant method of [13] is applied to compute $\mathcal{B}_{\text{opt}}$ here.

3.2. Adaptive boundary relaxation

Formulation and solver We relax the boundary preservation constraint to generate an inversion-free mesh. The optimization problem is formulated as follows:

$$\min_{\mathcal{R}} \quad E_{\text{inversion}}(\mathcal{R}) + \lambda E_{\text{distance}}(\mathcal{B}_r, \mathcal{B}_{\text{opt}}), \qquad (1)$$

where $E_{\text{inversion}}(\mathcal{R})$ is used to remove inverted elements, $E_{\text{distance}}(\mathcal{B}_r, \mathcal{B}_{\text{opt}})$ indicates the distance between two boundaries, and λ is a positive weight to control the strength of the boundary relaxation. As $\mathcal{B}_r$ and $\mathcal{B}_{\text{opt}}$ share the same connectivity, $E_{\text{distance}}(\mathcal{B}_r, \mathcal{B}_{\text{opt}})$ is defined as the sum of the squared distance between the corresponding vertices.

A hex is treated as a union of eight overlapping tetrahedrons, each of which is formed by one corner vertex and its three adjacent vertices in this hex. A piecewise linear map is defined from a trirectangular tetrahedron of a unit cube to one tetrahedron. To obtain an inversion-free mesh $\mathcal{R}$, we require that the determinant of the Jacobian matrix of each piecewise linear map is greater than zero. We follow [35] to define $E_{\text{inversion}}(\mathcal{R})$:

$$E_{\text{inversion}}(\mathcal{R}) = \sum_{\mathbf{h}\in\mathcal{H}} \sum_{i=1}^{8} \|J_{\mathbf{h},i} - A_{\mathbf{h},i}\|_F^2, \qquad (2)$$

where $\mathbf{h}$ is a hex of the hex set $\mathcal{H}$ of $\mathcal{R}$, $J_{\mathbf{h},i}$ is the Jacobian matrix of the piecewise linear map of i^{th} tetrahedron in $\mathbf{h}$, $A_{\mathbf{h},i}$ represents an auxiliary variable for the closest projection of $J_{\mathbf{h},i}$ onto a bounded conformal distortion space, and $\|\cdot\|_F$ denotes the Frobenius norm.

Since $E_{\text{distance}}(\mathcal{B}_r, \mathcal{B}_{\text{opt}})$ is a quadratic energy term, the solver of [35] can be directly used to optimize (1) with a λ.

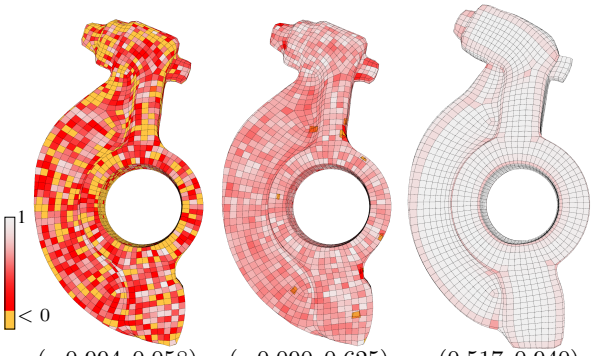


Figure 6. The process of adaptive boundary relaxation. Left: the input mesh. Middle: the resulting mesh (11 inverted elements) after optimization by setting $\lambda = \infty$. Right: the optimized result (no inverted elements) after using $\lambda = 0$. Below each model, we show $J_{\min}$ and J_{avg}.

Adaptive relaxation λ plays an important role in the boundary relaxation process. Intuitively, relaxing the boundary indicates $\lambda = 0$. However, this choice does not impose any constraints on the boundary, so that the relaxed boundary $\mathcal{B}_r$ is far away from $\mathcal{B}_{\text{opt}}$, thereby causing two main issues (Figure 5):

- The difficulty of the next optimization step is significantly increased.
- In extreme cases, the inversion-free mesh $\mathcal{R}$ cannot be generated.

Here, we propose an adaptive adjustment of λ to resolve these two issues. As shown in [35], optimizing $E_{\text{distance}}(\mathcal{B}_r, \mathcal{B}_{\text{opt}})$ while fixing the boundary can remove most inverted elements. Therefore, we adjust λ as follows:

1. By setting $\lambda = \infty$, i.e., fixing the boundary vertices, we solve (1) until termination.
2. We allow the boundary to move freely by setting $\lambda = 0$ until termination.

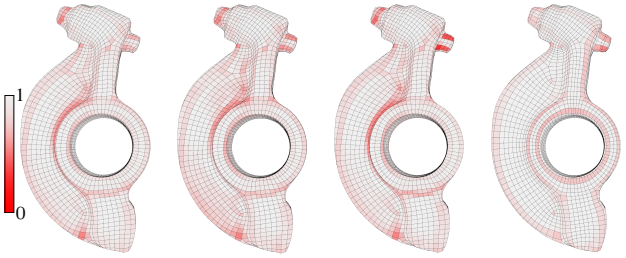
This simple strategy succeeds in generating inversion-free meshes without producing large $E_{\text{distance}}(\mathcal{B}_r, \mathcal{B}_{\text{opt}})$ for our testing date set containing 1004 input meshes. Figure 6 shows an example of our adaptive strategy.

3.3. Adaptive boundary difference reduction

Formulation As $\mathcal{R}$ is inversion-free after the adaptive boundary relaxation, we want to keep inversion-free when reducing the boundary difference. We formulate the optimization problem as follows:

$$\min_{\mathcal{R}} \quad E_{\text{distortion}}(\mathcal{R}) + \lambda_2 E_{\text{distance}}(\mathcal{B}_r, \mathcal{B}_{\text{opt}}), \qquad (3)$$

where $E_{\text{distortion}}(\mathcal{R})$ is an inversion-preventing term that goes to infinity when a hex flips or degenerates and λ_2 is a positive



(a) (0.401,0.884) **(b)** (0.338,0.877) **(c)** (0.309,0.869) **(d)** (0.545,0.958)

Figure 7. Adaptive boundary difference reduction process, the initialization of which is shown in Figure 6 - Right. (a) The mesh ($E_{\text{distance}} = 3204.220$) after the first iteration with $\lambda_2 = 0.002$. (b) The mesh ($E_{\text{distance}} = 35.521$) after the second iteration with $\lambda_2 = 0.203$. (c) The mesh ($E_{\text{distance}} = 0.009$) after the third iteration with $\lambda_2 = 775.721$. (d) The mesh after post-optimization. The text below each model indicates $J_{\min}$ and J_{avg}.

weight. At the same time, $E_{\text{distortion}}(\mathcal{R})$ is also a term to measure the quality of hexes. We use the symmetric Dirichlet energy to define $E_{\text{distortion}}(\mathcal{R})$:

$$E_{\text{distortion}}(\mathcal{R}) = \sum_{\mathbf{h}\in\mathcal{H}} \sum_{i=1}^{8} D(\hat{J}_{\mathbf{h},i}), \tag{4}$$

where $\hat{J}_{\mathbf{h},i}$ is the Jacobian of the mapping from a trirectangular tetrahedron of a cuboid to one tetrahedron, and

$$D(\hat{J}_{\mathbf{h},i}) = \begin{cases} \|\hat{J}_{\mathbf{h},i}\|_F^2 + \|\hat{J}_{\mathbf{h},i}^{-1}\|_F^2, & \text{if } \det(\hat{J}_{\mathbf{h},i}) > 0, \\ \infty, & \text{otherwise.} \end{cases}$$

We compute the length, width and height of the reference cuboid as the average lengths of the four edges on the corresponding directions.

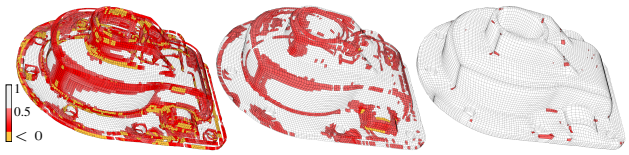
Adaptive reduction To make $E_{\text{distance}}(\mathcal{B}_r, \mathcal{B}_{\text{opt}})$ approach zero, the following optimization process is developed:

1. Set the iteration number $k = 1$.
2. Adaptively update λ_2 as follows [47]:

$$\lambda_2 \leftarrow \min(\lambda_{\min} \cdot \max(100 \times \frac{E_{\text{distortion}}(\mathcal{R}^{(k)})}{E_{\text{distance}}(\mathcal{B}_r^{(k)}, \mathcal{B}_{\text{opt}})}, 1), \lambda_{\max}), \tag{5}$$

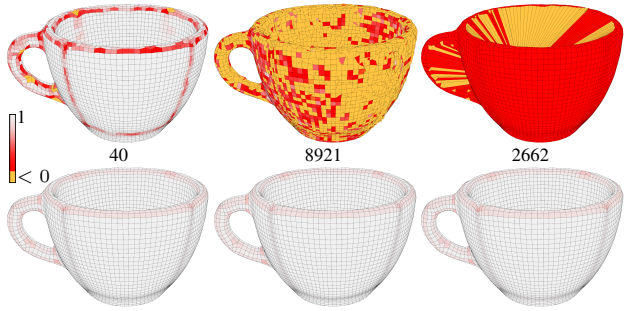
   where $\mathcal{R}^{(k)}$ is the hex mesh at k^{th} iteration, $\mathcal{B}_r^{(k)}$ is the boundary surface of $\mathcal{R}^{(k)}$, $\lambda_{\min} = 10^{-2}$, and $\lambda_{\max} = 10^{15}$ in our experiments.
3. The second-order solver of [34] is used to compute a descent direction $\mathbf{d}^{(k)}$ of $E_{\text{distortion}}(\mathcal{R}^{(k)}) + \lambda_2 E_{\text{distance}}(\mathcal{B}_r^{(k)}, \mathcal{B}_{\text{opt}})$.
4. Along $\mathbf{d}^{(k)}$, a standard Armijo backtracking algorithm is performed to determine the step size and compute $\mathcal{R}^{(k+1)}$.
5. $k \leftarrow k + 1$. We terminate the iteration if $E_{\text{distance}}(\mathcal{B}_r^{(k)}, \mathcal{B}_{\text{opt}}) < 10^{-15}$ or $k > 10$; otherwise, we go to Step 2.

We show an example in Figure 7. In the aforementioned optimization process, λ_2 will increase after E_{distance} decreases, thereby making the solver focus on penalizing E_{distance} to approach zero. This adaptive optimization process works very well in practice, and it takes at most 5 iterations to converge in all our tested examples. To further improve our mesh quality, we use the method in [9] to optimize the vertex position without modification of the mesh connectivity (Figure 7 (d)).



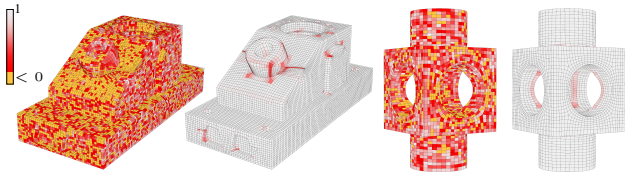
(a) (-0.988, 0.756) **(b)** (-0.702, 0.814) **(c)** (0.178, 0.897)

Figure 8. Left: the input mesh. Middle: the resulting mesh (26 inverted elements) by setting $\lambda = 1000$ without adaptive boundary relaxation. Right: our result (no inverted elements) with adaptive boundary relaxation. Below each model, $J_{\min}$ and J_{avg} are shown.



(a) $(0.637, 0.969)$ **(b)** $(0.652, 0.971)$ **(c)** $(0.669, 0.963)$

Figure 9. Various initializations. (a) The input mesh with 16862 hexahedral elements. (b) The mesh generated by random placement of vertices. (c) The mesh generated by relocating all interior vertices to one position. The number in the first row indicates the number of input inverted elements. We show $J_{\min}$ and J_{avg} below each result model.



(a) (-0.999,0.044) **(b)** (0.254,0.984) **(c)** (-0.753,0.887) **(d)** (0.471,0.976)

Figure 10. Feature preservation. Given the input mesh with sharp features(a)(c), our algorithm succeeds in generating high-quality meshes while preserving sharp features(b)(d).

4. Experiments

We have tested our algorithm on various all-hex meshes to evaluate its performance. Our method is implemented in C++, and all the experiments are performed on a desktop PC

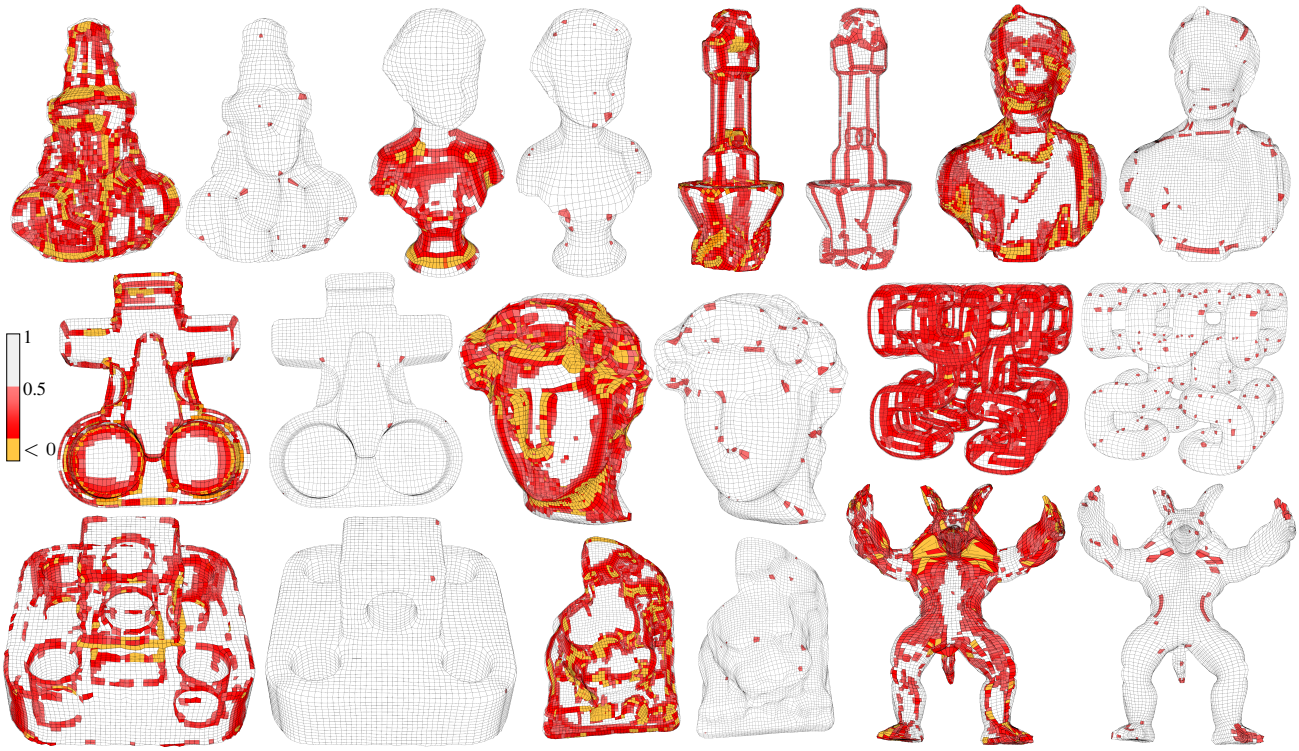


Figure 11. Gallery. Our method succeeds in generating inversion-free hex-meshes.

with a 4.00 GHz Intel Core i7-4790K and 16 GB of RAM. The linear systems are solved using the Intel® Math Kernel Library. Statistics and timings for all the demonstrated examples are reported in Table 1.

Without boundary relaxation We test the necessity of the boundary relaxation process. A large λ in (4) can be applied to preserve the boundary surface. Then, we solve (4) with $\lambda = 1000$ to consider both the boundary preservation constraint and inversion-free constraint. We compare this setting in Figure 8. Although boundary preservation is a soft constraint, it still limits the valid space for searching the inversion-free meshes. Then, the movements of vertices near the boundary are restricted to a large extent. Thereby, some inverted elements near the boundary can not be removed. In short, the boundary relaxation is necessary.

Various initializations We test three types of initializations for one model: (1) original mesh, (2) randomly placing the vertices, (3) relocating all interior vertices to one position. Our algorithm succeeds for every type, as shown in Figure 9. The computational times are (4.6, 70.3 for Figure 9 (a)), (6.0, 48.1 for Figure 9 (b)), and (9.3, 66.5 for Figure 9 (c)) in seconds, respectively. We observed that the convergence speed is not seriously affected by the initialization. Besides, the resulting scaled Jacobian values are very similar, demonstrating that our algorithm is not sensitive to the initialization.

Feature preservation Preserving features are important for all-hex mesh generation [26, 12]. Our optimization method can be modified for feature preservation. First, the boundary edges whose dihedral angle is smaller than 140° are detected as sharp features. Secondly, we classify all the boundary vertices into three types, i.e., corner vertices, feature line vertices, and regular vertices, as introduced in [23]. During the optimization of the boundary quad-mesh, the corner vertice is fixed, the feature line vertex can move along the feature line, and the regular vertex is projected to its tangent plane. After pulling the boundary back to appraoch the optimized quad boundary, the features are preserved. We test our algorithm on models with sharp features provided by [12], as shown in Figure 10. High-quality meshes are obtained while preserving sharp features.

Testing on a data set To verify the effectiveness and robustness of our method, we test our algorithm on a data set containing 1004 models. The data set contains three parts. The first part is provided by [46]. Then, we artificially corrupt the models in the first part by random vertex displacement to have a high ratio of inverted elements. Finally, we place all interior vertices of each model in the first part at one position to generate the third part. Our method succeeds in generating inversion-free and high-quality hex-meshes for

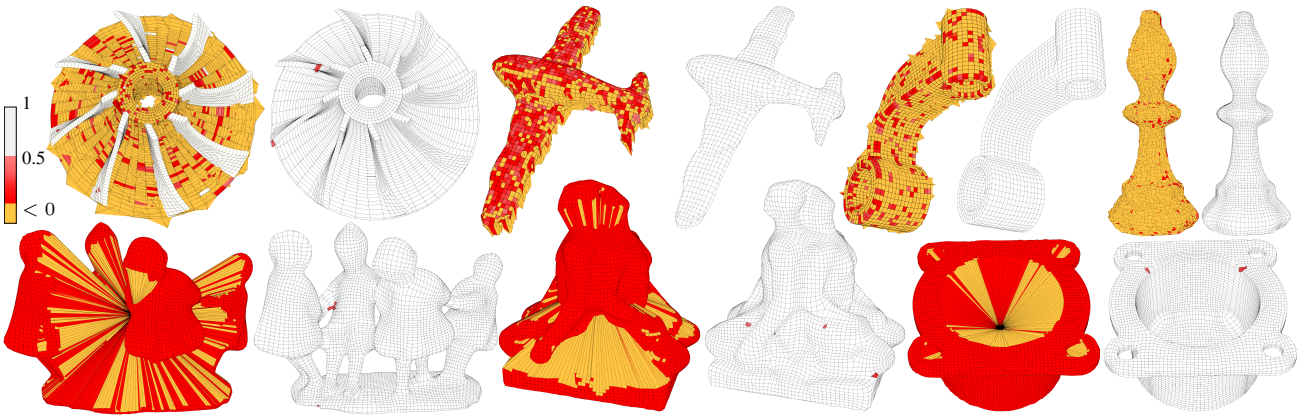


Figure 12. Results on artificially corrupted inputs. Our method succeeds in generating inversion-free results.

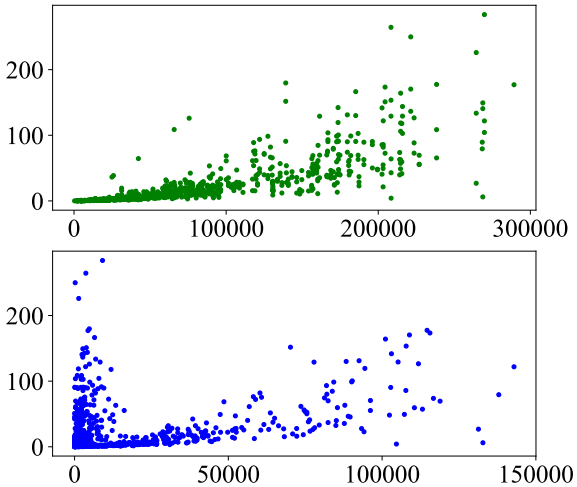


Figure 13. We plot the computation time (in minutes) vs. the number of hexes (N_{hex}) (upper), and the running time (in minutes) vs. the number of inverted elements (N_{inv}) (bottom).

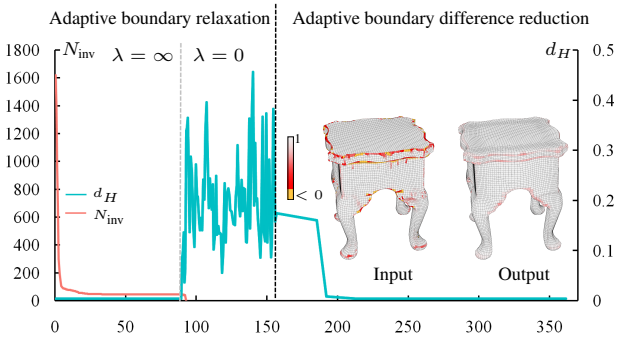


Figure 14. Timings. The graph plots the number of inverted elements (N_{inv}) vs. time (in seconds) and the two-sided Hausdorff distance (d_H) (w.r.t. the diagonal length of the bounding box of the input mesh) vs. time (in seconds) for the Stool model.

all models (Figures 11, 12, and 15). Results on our data set demonstrate the practical robustness of our method.

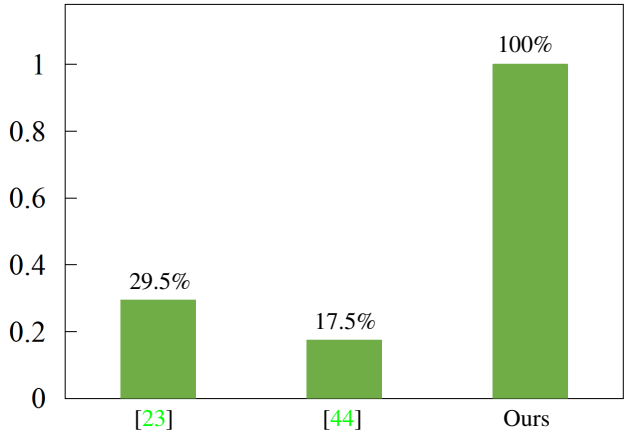


Figure 15. The success rates of three methods.

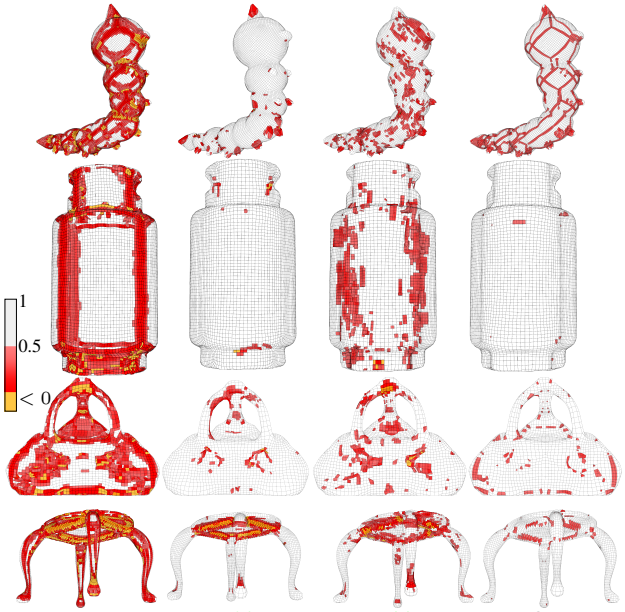


Figure 16. Comparisons with [23] (second column) and [44] (third column) on four models.

Complexity of our algorithm The complexity of our algorithm is affected by both the size and quality of input models, i.e., the number of hexes and the number of inverted hexes. We plot the running time with respect to the number of hexes in Figure 13 - upper and the computational time with respect to the number of inverted hexes in Figure 13 - bottom. When one of the two numbers increases, the computation time becomes longer. The longest computational time in our experiment is 283.8 minutes, where the model contains 269786 hexes and 9090 inverted hexes. What's worse, almost all inverted elements of this model are near the boundary, increasing difficulty to remove the inversions. In practice, our algorithm is efficient for most models.

Timings We plot the number of inverted elements and the two-sided Hausdorff distance with respect to time in Figure 14. The number of inverted hexes tends to decrease over time in the adaptive boundary relaxation, and it is fixed to zero during the adaptive boundary difference reduction step. The Hausdorff distance becomes larger in adaptive boundary relaxation step and decreases in the adaptive boundary difference reduction step.

Comparisons We compare with the two previous methods: [23] and [44]. We use authors' provided implementations to generate their results. Our method successfully computes inversion-free results for all models while their success rates are both below 30%, as shown in Figure 15. Figure 16 shows comparisons on four examples. Our method is able to eliminate all the inverted elements and generate high-quality meshes while they both fail. Besides, our approach is faster than theirs. We observed that [23] does not generate inversion-free results for some examples even if it runs for a long time. For example, the result of [23]in Figure 1 still contains 109 inverted elements after running for more than 6.5 hours. For Figure 2, it takes over 4 minutes to get the result containing 18 inverted elements using the method of [44], while our method succeeds in generating an inversion-free result within 1 minute. Besides, the method of [44] is very sensitive to the selection of the algorithm parameters. Our method is more robust and general due to the adaptive boundary optimization strategy.

5. Conclusion

In this paper, we present a novel framework to geometrically optimize hexahedral meshes. Given an input all-hex mesh with inversions, we firstly generate a quad mesh with high quality and small difference to the input boundary mesh. Then we relex the boundary preservation constraint and optimize for an inversion-free hex mesh. At last, we minimize the difference between the boundary of inversion-free mesh and the optimized quad mesh of the first step without bringing in inverted elements. We have tested our algorithm on a large data set with different initializations. The results show that our algorithm outperforms the previous methods.

Theoretical guarantee Although we successfully compute high-quality inversion-free results with small Hausdorff distances for 1004 hexahedral meshes, our success is not theoretically guaranteed. Since mesh connectivity constrains the valid space of inversion-free volumetric mappings, adaptively changing the connectivity during the optimization may be a way to guarantee our success. Developing a theoretically guaranteed method by employing connectivity modifications is still an interesting direction for future research.

Inversion check We verify the inversion of a hexahedron by checking whether one of the eight corresponding tetrahedrons is inverse. However, [38] empirically studied this metric and found it insufficient. We would like to extend our method to support a more reliable metric for inversion check of hexahedrons.

References

[1] N. Aigerman and Y. Lipman. Injective and bounded distortion mappings in 3d. *ACM Transactions on Graphics (TOG)*, 32(4):1–14, 2013. 3

[2] T. Blacker. Meeting the challenge for automated conformal hexahedral meshing. In *9th international meshing roundtable*, pages 11–20. Citeseer, 2000. 1

[3] X. Du, N. Aigerman, Q. Zhou, S. Z. Kovalsky, Y. Yan, D. M. Kaufman, and T. Ju. Lifting simplices to find injectivity. *ACM Transactions on Graphics (TOG)*, 39(4):120–1, 2020. 3

[4] J. M. Escobar, E. Rodrıguez, R. Montenegro, G. Montero, and J. M. González-Yuste. Simultaneous untangling and smoothing of tetrahedral meshes. *Computer Methods in Applied Mechanics and Engineering*, 192(25):2775–2787, 2003. 3

[5] X.-M. Fu and Y. Liu. Computing inversion-free mappings by simplex assembly. *ACM Transactions on Graphics (TOG)*, 35(6):1–12, 2016. 3

[6] X.-M. Fu, Y. Liu, and B. Guo. Computing locally injective mappings by advanced mips. *ACM Transactions on Graphics (TOG)*, 34(4):1–12, 2015. 3

[7] X.-M. Fu, J.-P. Su, Z.-Y. Zhao, Q. Fang, C. Ye, and L. Liu. Inversion-free geometric mapping construction: A survey. *Computational Visual Media*, 7(3):289–318, 2021. 3

[8] X. Gao and G. Chen. A local frame based hexahedral mesh optimization. *Proceedings of the 25th International Meshing Roundtable*, 2, 2016. 1

[9] X. Gao, D. Panozzo, W. Wang, Z. Deng, and G. Chen. Robust structure simplification for hex re-meshing. *ACM Transactions on Graphics (TOG)*, 36(6):1–13, 2017. 5

[10] V. Garanzha, I. Kaporin, L. Kudryavtseva, F. Protais, N. Ray, and D. Sokolov. Foldover-free maps in 50 lines of code. *arXiv preprint arXiv:2102.03069*, 2021. 3

[11] A. Garg, A. O. Sageman-Furnas, B. Deng, Y. Yue, E. Grinspun, M. Pauly, and M. Wardetzky. Wire mesh design. *ACM Transactions on Graphics*, 33(4), 2014. 4

Model	N_{hex}	Pre			After			$d_H(\times d_{\text{bb}})$	$t_1(s)$	$t_2(s)$
		$J_{\min}$	J_{avg}	N_{inv}	$J_{\min}$	J_{avg}	N_{inv}			
Fig. 1 (b)	83062	-0.998	0.891	853	-0.602	0.948	109	0.00739	1547.3	\
Fig. 1 (c)	83062	-0.998	0.891	853	0.434	0.986	0	0.00612	35.7	480.2
Fig. 2 (b)	19496	-1.000	0.843	726	-0.930	0.899	18	0.01319	243.9	\
Fig. 2 (c)	19496	-1.000	0.843	726	0.366	0.966	0	0.00980	12.1	55.0
Fig. 3 (a)	52730	-0.982	0.883	1000	0.407	0.974	0	0.00923	60.2	366.0
Fig. 3 (b)	16720	-1.000	-0.037	7964	0.440	0.971	0	0.00502	5.3	126.5
Fig. 3 (c)	7083	-0.981	0.187	187	0.502	0.951	0	0.00501	3.1	24.3
Fig. 4 (d)	33232	-1.000	-0.064	17141	0.413	0.936	0	0.00981	11.2	193.1
Fig. 7 (d)	10600	-0.994	-0.058	5690	0.545	0.958	0	0.00657	40.3	34.1
Fig. 8 (c)	69523	0.756	-0.988	2103	0.178	0.897	0	0.00554	28.7	672.0
Fig. 9 (a)	16862	-0.999	-0.068	40	0.637	0.969	0	0.00569	4.6	70.3
Fig. 9 (b)	16862	-0.999	-0.068	8921	0.652	0.971	0	0.00604	6.0	48.1
Fig. 9 (c)	16862	-0.999	-0.068	2662	0.669	0.963	0	0.00632	9.3	66.5
Fig. 10 (b)	85719	-0.999	0.044	36600	0.254	0.984	0	0.00204	16.3	719.1
Fig. 10 (d)	40488	-0.753	0.887	822	0.471	0.976	0	0.00717	12.0	41.2
Fig. 11 row 1,col 1	17964	-0.908	0.795	663	0.392	0.953	0	0.00859	20.6	70.3
Fig. 11 row 1,col 2	13716	-0.717	0.817	271	0.414	0.875	0	0.00315	2.2	48.1
Fig. 11 row 1,col 3	22817	-0.920	0.817	643	0.264	0.915	0	0.00596	49.3	152.4
Fig. 11 row 1,col 4	65646	-0.992	0.836	1423	0.338	0.970	0	0.00972	90.7	318.3
Fig. 11 row 2,col 1	40724	-0.955	0.827	1188	0.379	0.967	0	0.00996	7.1	235.4
Fig. 11 row 2,col 2	14130	-0.958	0.753	380	0.241	0.868	0	0.00427	3.7	60.1
Fig. 11 row 2,col 3	36240	-0.414	0.824	32	0.423	0.936	0	0.00443	14.3	122.2
Fig. 11 row 3,col 1	50063	-0.576	0.925	139	0.447	0.982	0	0.00863	10.3	318.4
Fig. 11 row 3,col 2	65322	0.998	0.905	893	0.381	0.982	0	0.00976	89.6	769.2
Fig. 11 row 3,col 3	29935	-0.755	0.686	323	0.283	0.879	0	0.01030	6.3	78.0
Fig. 12 row 1,col 1	11174	-0.999	-0.274	8854	0.387	0.960	0	0.00751	12.2	86.5
Fig. 12 row 1,col 2	4972	-1.000	-0.270	3510	0.521	0.895	0	0.00496	2.7	21.3
Fig. 12 row 1,col 3	4539	-0.999	-0.482	4116	0.744	0.980	0	0.00551	19.6	20.5
Fig. 12 row 1,col 4	18679	-0.999	-0.073	9739	0.580	0.963	0	0.00451	9.2	137.3
Fig. 12 row 2,col 1	35293	-0.998	0.075	4168	0.417	0.958	0	0.00667	14.5	86.3
Fig. 12 row 2,col 2	37119	-0.925	-0.011	1303	0.258	0.911	0	0.00255	13.9	269.1
Fig. 12 row 2,col 3	33880	-1.000	-0.108	4947	0.307	0.960	0	0.00976	14.6	157.8
Fig. 14	69278	-0.999	0.801	2461	0.314	0.971	0	0.00762	156.5	203.7
Fig. 16 row 1,col 2	57196	-0.924	0.823	1119	-0.949	0.945	81	0.00972	1719.6	\
Fig. 16 row 1,col 3	57196	-0.924	0.823	1119	-0.981	0.858	54	0.02023	2221.7	\
Fig. 16 row 1,col 4	57196	-0.924	0.823	1119	0.250	0.907	0	0.00683	13.3	409.5
Fig. 16 row 2,col 2	55769	-0.924	0.891	250	-0.173	0.971	7	0.00821	840.8	\
Fig. 16 row 2,col 3	55769	-0.924	0.891	250	-0.531	0.900	11	0.00779	1323.6	\
Fig. 16 row 2,col 4	55769	-0.924	0.891	250	0.426	0.982	0	0.00734	8.7	511.3
Fig. 16 row 3,col 2	20288	-0.997	0.819	375	-0.401	0.912	8	0.00818	353.4	\
Fig. 16 row 3,col 3	20288	-0.997	0.819	375	-0.987	0.856	44	0.01318	245.2	\
Fig. 16 row 3,col 4	20288	-0.997	0.819	375	0.297	0.936	0	0.00791	33.0	102.4
Fig. 16 row 4,col 2	23744	-0.997	0.719	1297	-0.796	0.834	497	0.00961	817.0	\
Fig. 16 row 4,col 3	23744	-0.997	0.719	1297	-0.971	0.794	94	0.00942	663.2	\
Fig. 16 row 4,col 4	23744	-0.997	0.719	1297	0.322	0.935	0	0.00945	39.1	139.5

Table 1. Statistics and timings. For each model, we report the number of the input hexahedra elements (N_{hex}), the minimum scaled Jacobian ($J_{\min}$), the average scaled Jacobian (J_{avg}), the number of inverted elements (N_{inv}) before and after optimization, and the Hausdorff distance(d_H) between the output boundary mesh and the input boundary mesh. We show the computational time in seconds required for the first step (t_1) and the second step (t_2) for our method. We also include the statistics and timing results for competitors. For each competitor, t_1 means its total computational time and t_2 is not used.

[12] H.-X. Guo, X. Liu, D.-M. Yan, and Y. Liu. Cut-enhanced polycube-maps for feature-aware all-hex meshing. *ACM Trans. Graph.*, 39(4), 2020. 6

[13] B. Hu, C. Ye, J.-P. Su, and L. Liu. Manifold-constrained geometric optimization via local parameterizations. *IEEE Transactions on Visualization & Computer Graphics*, (01):1–1, 2021. 4

[14] C. Jiang, C. Wang, F. Rist, J. Wallner, and H. Pottmann. Quad-mesh based isometric mappings and developable surfaces. *ACM Transactions on Graphics (TOG)*, 39(4):128–1, 2020. 4

[15] Z. Jiang, S. Schaefer, and D. Panozzo. Simplicial complex augmentation framework for bijective maps. *ACM Transactions on Graphics*, 36(6), 2017. 3

[16] P. M. Knupp. Hexahedral and tetrahedral mesh untangling. *Engineering with Computers*, 17(3):261–268, 2001. 2

[17] S. Z. Kovalsky, N. Aigerman, R. Basri, and Y. Lipman. Controlling singular values with semidefinite programming. *ACM Trans. Graph.*, 33(4):68–1, 2014. 3

[18] S. Z. Kovalsky, N. Aigerman, R. Basri, and Y. Lipman. Large-scale bounded distortion mappings. *ACM Trans. Graph.*,

34(6):191–1, 2015. 3

[19] S. Z. Kovalsky, M. Galun, and Y. Lipman. Accelerated quadratic proxy for geometric optimization. *ACM Transactions on Graphics (TOG)*, 35(4):1–11, 2016. 3

[20] P. Kunnp. A method for hexahedral mesh shape optimization. *International Journal for Numerical methods in Engineering*, 58:319–32, 2003. 2

[21] H.-Y. Liu, Z.-Y. Liu, Z.-Y. Zhao, L. Liu, and X.-M. Fu. Practical fabrication of discrete chebyshev nets. In *Computer Graphics Forum*, volume 39, pages 13–26. Wiley Online Library, 2020. 4

[22] T. Liu, M. Gao, L. Zhu, E. Sifakis, and L. Kavan. Fast and robust inversion-free shape manipulation. In *Computer Graphics Forum*, volume 35, pages 1–11. Wiley Online Library, 2016. 3

[23] M. Livesu, A. Sheffer, N. Vining, and M. Tarini. Practical hex-mesh optimization via edge-cone rectification. *ACM Transactions on Graphics (TOG)*, 34(4):1–11, 2015. 1, 2, 6, 7, 8

[24] S. J. Owen. A survey of unstructured mesh generation technology. *IMR*, 239:267, 1998. 1, 2

[25] P. P. Pébay, D. Thompson, J. Shepherd, P. Knupp, C. Lisle, V. A. Magnotta, and N. M. Grosland. New applications of the verdict library for standardized mesh verification pre, post, and end-to-end processing. In *Proceedings of the 16th International Meshing Roundtable*, pages 535–552. Springer, 2008. 1

[26] J. Qian and Y. Zhang. Automatic unstructured all-hexahedral mesh generation from b-reps for non-manifold cad assemblies. *Engineering with Computers*, 28(4):345–359, 2012. 6

[27] J. Qian, Y. Zhang, W. Wang, A. Lewis, M. Qidwai, and A. Geltmacher. Quality improvement of non-manifold hexahedral meshes for critical feature determination of microstructure materials. *International Journal for Numerical Methods in Engineering*, 82(11):1406–1423, 1 2010. 2

[28] M. Rabinovich, R. Poranne, D. Panozzo, and O. Sorkine-Hornung. Scalable locally injective mappings. *ACM Transactions on Graphics (TOG)*, 36(4):1, 2017. 3

[29] E. Ruiz-Gironés, X. Roca, and J. Sarrate. Optimizing mesh distortion by hierarchical iteration relocation of the nodes on the cad entities. *Procedia Engineering*, 82:101–113, 2014. 2

[30] E. Ruiz-Gironés, X. Roca, J. Sarrate, R. Montenegro, and J. M. Escobar. Simultaneous untangling and smoothing of quadrilateral and hexahedral meshes using an object-oriented framework. *Advances in Engineering Software*, 80:12–24, 2015. 2

[31] S. P. Sastry and S. M. Shontz. A comparison of gradient- and hessian-based optimization methods for tetrahedral mesh quality improvement. In *Proceedings of the 18th International Meshing Roundtable*, pages 631–648. Springer, 2009. 2

[32] C. Schüller, L. Kavan, D. Panozzo, and O. Sorkine-Hornung. Locally injective mappings. In *Computer Graphics Forum*, volume 32, pages 125–135. Wiley Online Library, 2013. 3

[33] J. F. Shepherd and C. R. Johnson. Hexahedral mesh generation constraints. *Engineering with Computers*, 24(3):195–213, 2008. 1

[34] B. Smith, F. D. Goes, and T. Kim. Analytic eigensystems for isotropic distortion energies. *ACM Transactions on Graphics (TOG)*, 38(1):1–15, 2019. 5

[35] J.-P. Su, X.-M. Fu, and L. Liu. Practical foldover-free volumetric mapping construction. In *Computer Graphics Forum*, volume 38, pages 287–297. Wiley Online Library, 2019. 2, 3, 4

[36] L. Sun, G. Zhao, and X. Ma. Quality improvement methods for hexahedral element meshes adaptively generated using grid-based algorithm. *International journal for numerical methods in engineering*, 89(6):726–761, 2012. 2

[37] T. Toulorge, C. Geuzaine, J.-F. Remacle, and J. Lambrechts. Robust untangling of curvilinear meshes. *Journal of Computational Physics*, 254:8–26, 2013. 3

[38] O. V. Ushakova. Nondegeneracy tests for hexahedral cells. *Computer methods in applied mechanics and engineering*, 200(17-20):1649–1658, 2011. 8

[39] D. Vartziotis and M. Papadrakakis. Improved getme by adaptive mesh smoothing. *Computer Assisted Methods in Engineering and Science*, 20(1):55–71, 2017. 2

[40] R. Wang, S. Gao, Z. Zheng, and J. Chen. Hex mesh topological improvement based on frame field and sheet adjustment. *Computer-Aided Design*, 103:103–117, 2018. 3

[41] R. Wang, Z. Zheng, W. Yu, Y. Shao, and S. Gao. Structure-aware geometric optimization of hexahedral mesh. *Computer-Aided Design*, 138:103050, 2021. 3

[42] T. J. Wilson. *Simultaneous untangling and smoothing of hexahedral meshes*. PhD thesis, master's thesis, Universitat Politecnica de Catalunya, Spain, 2011. 2

[43] T. J. Wilson, J. Sarrate Ramos, X. Roca Ramón, R. Montenegro, and J. Escobar. Untangling and smoothing of quadrilateral and hexahedral meshes. *Civil-Comp Proceedings*, 2012. 2

[44] K. Xu, X. Gao, and G. Chen. Hexahedral mesh quality improvement via edge-angle optimization. *Computers & Graphics*, 70:17–27, 2018. 1, 2, 3, 7, 8

[45] Y. Yang, X.-M. Fu, S. Chai, S.-W. Xiao, and L. Liu. Volume-enhanced compatible remeshing of 3d models. *IEEE transactions on visualization and computer graphics*, 25(10):2999–3010, 2018. 3

[46] Y. Yang, X.-M. Fu, and L. Liu. Computing surface polycube-maps by constrained voxelization. *Comput. Graph. Forum*, 38(7):299–309, 2019. 6

[47] G.-J. Yuan, H. Liu, J.-P. Su, and X.-M. Fu. Computing planar and volumetric b-spline parameterizations for iga by robust mapping fitting. *Computer Aided Geometric Design*, 86:101968, 2021. 5

[48] Y. Zhang, C. Bajaj, and G. Xu. Surface smoothing and quality improvement of quadrilateral/hexahedral mesheswith geometric flow. *Communications in numerical methods in engineering*, 25(1):1, 1 2009. 2

[49] Y. Zhu, R. Bridson, and D. M. Kaufman. Blended cured quasi-newton for distortion optimization. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018. 3