

# グラフニューラルネットワークを用いたインテリジェントメッシュスムージング手法の提案

Zhichao Wang<sup>1,2</sup>, Xinhai Chen<sup>1,2\*</sup>, Junjun Yan<sup>1,2</sup>, Jie Liu<sup>1,2</sup>

<sup>1</sup>Science and Technology on Parallel and Distributed Processing Laboratory, National University of Defense Technology, Changsha, 410073, China.

<sup>2</sup>Laboratory of Digitizing Software for Frontier Equipment, National University of Defense Technology, Changsha, 410073.

\*Corresponding author(s). E-mail(s): [chenxinhai16@nudt.edu.cn](mailto:chenxinhai16@nudt.edu.cn);  
Contributing authors: [wangzhichao@nudt.edu.cn](mailto:wangzhichao@nudt.edu.cn); [yanjunjun@nudt.edu.cn](mailto:yanjunjun@nudt.edu.cn);  
[liujie@nudt.edu.cn](mailto:liujie@nudt.edu.cn);

## Abstract

CFDでは、メッシュの平滑化手法は、高精度な数値シミュレーションを実現するために、メッシュの品質を改良するために一般的に利用されている。具体的には、最適化ベースの平滑化は高品質なメッシュ平滑化に使用されるが、計算オーバーヘッドが大きい。先駆的な研究は、高品質なメッシュから平滑化手法を学習するために教師あり学習を採用することで、平滑化効率を向上させる。しかし、メッシュノードの次数を変化させた場合、メッシュノードの平滑化が困難であり、ノード入力シーケンスの問題に対処するためデータ補強が必要である。さらに、ラベル付けされた高品質なメッシュが必要なため、提案手法の適用性がさらに制限される。本論文では、インテリジェントメッシュスムージングのための軽量ニューラルネットワークモデルであるGMSNetを紹介する。GMSNetは、ノードの近傍の特徴を抽出し、最適なノード位置を出力するために、グラフニューラルネットワークを採用する。平滑化の際にも、GMSNetが負の体積要素を生成するのを防ぐために、フォールトトレランス機構を導入する。軽量なモデルにより、GMSNetは程度の差こそあれメッシュノードを効果的に平滑化することができ、入力データの順序に影響されない。また、高品質なメッシュを必要としない新しい損失関数MetricLossを開発し、学習中に安定かつ迅速な収束を実現した。GMSNetを、一般的に用いられているメッシュスムージング手法と2次元三角メッシュ上で比較する。実験結果より、GMSNetは前モデルの5%のモデルパラメータで優れたメッシュ平滑化性能を達成し、最適化ベースの平滑化よりも8.62倍高速に達成することがわかった。

キーワード Unstructured Mesh, メッシュスムージング, グラフニューラルネットワーク, 最適化ベース

Smoothing

# Proposing an intelligent mesh smoothing method with graph neural networks

Zhichao Wang<sup>1,2</sup>, Xinhai Chen<sup>1,2\*</sup>, Junjun Yan<sup>1,2</sup>, Jie Liu<sup>1,2</sup>

<sup>1</sup>Science and Technology on Parallel and Distributed Processing Laboratory, National University of Defense Technology, Changsha, 410073, China.

<sup>2</sup>Laboratory of Digitizing Software for Frontier Equipment, National University of Defense Technology, Changsha, 410073.

\*Corresponding author(s). E-mail(s): [chenxinhai16@nudt.edu.cn](mailto:chenxinhai16@nudt.edu.cn);

Contributing authors: [wangzhichao@nudt.edu.cn](mailto:wangzhichao@nudt.edu.cn); [yanjunjun@nudt.edu.cn](mailto:yanjunjun@nudt.edu.cn);  
[liujie@nudt.edu.cn](mailto:liujie@nudt.edu.cn);

## Abstract

In CFD, mesh smoothing methods are commonly utilized to refine the mesh quality to achieve high-precision numerical simulations. Specifically, optimization-based smoothing is used for high-quality mesh smoothing, but it incurs significant computational overhead. Pioneer works improve its smoothing efficiency by adopting supervised learning to learn smoothing methods from high-quality meshes. However, they pose difficulty in smoothing the mesh nodes with varying degrees and also need data augmentation to address the node input sequence problem. Additionally, the required labeled high-quality meshes further limit the applicability of the proposed method. In this paper, we present GMSNet, a lightweight neural network model for intelligent mesh smoothing. GMSNet adopts graph neural networks to extract features of the node's neighbors and output the optimal node position. During smoothing, we also introduce a fault-tolerance mechanism to prevent GMSNet from generating negative volume elements. With a lightweight model, GMSNet can effectively smoothing mesh nodes with varying degrees and remain unaffected by the order of input data. A novel loss function, MetricLoss, is also developed to eliminate the need for high-quality meshes, which provides a stable and rapid convergence during training. We compare GMSNet with commonly used mesh smoothing methods on two-dimensional triangle meshes. The experimental results show that GMSNet achieves outstanding mesh smoothing performances with 5% model parameters of the previous model, and attains 8.62 times faster than optimization-based smoothing.

**Keywords:** Unstructured Mesh, Mesh Smoothing, Graph Neural Network, Optimization-based Smoothing

# 1 Introduction

コンピュータ技術の急速な進歩に伴い、計算流体力学(CFD)は流体力学の原理を研究するための重要な手法として浮上してきた。その幅広い応用範囲は、航空宇宙、水工学、自動車工学、生物医学など、多様な分野に及んでいる[1-4]。通常、CFDシミュレーションは、支配的な物理方程式を離散化し、その後、離散化された方程式の大規模な代数系を解いて流体変数を求めるこによって行われる。離散化はCFDの重要なステップであり、支配的な物理方程式の離散化と計算領域の離散化という2つの重要な側面を包含している[5]。後者のプロセスはメッシュ生成と呼ばれ、CFDにおいて基本的な役割を果たす。計算領域を、2次元領域のポリゴンや3次元領域のポリヘドラのような、重ならないメッシュ要素に分割することを含む[6]。生成されたメッシュの品質は、数値シミュレーションの収束性、精度、効率に大きな影響を与える。メッシュ要素の直交性、滑らかさ、分配性、密度分布は、解行列の安定性と収束性に大きく影響します[7]。その結果、高品質なメッシュ生成の探求は、CFD研究において活気に満ちた活発な分野であり続けている。実用的なメッシュ生成プロセスにおいて、最初に生成されたメッシュは、しばしばシミュレーションの要件を満たさない。メッシュの品質を向上させるために、メッシュスマージング、フェイススワップ、エッジスワップ、ポイント挿入/削除などのメッシュ品質向上技術が一般的に採用されています[8- 10]。このような手法の中で、メッシュの品質を向上させるために最もよく使われる手法はメッシュスマージング法である。

メッシュスマージングは、ヒューリスティックスマージングと最適化ベーススマージングの2種類に大別される[11]。代表的なヒューリスティック手法として、ラプラシアンスマージング[12]がある(図1a)。ラプラシアンスマージングでは、メッシュノードをStarPolygon(図1に示すように、ノードを含む多面体)のノードの座標の算術平均に配置して平滑化する。この方法は効率的であるが、StarPolygonが非凸の場合、負の体積要素を生成する可能性がある。

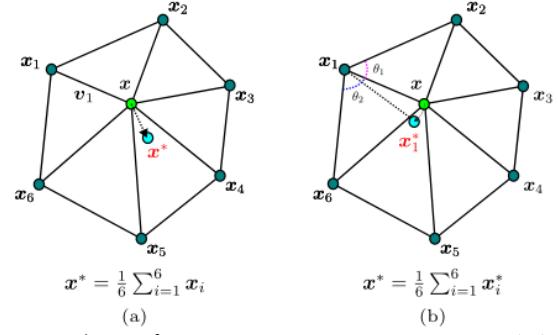


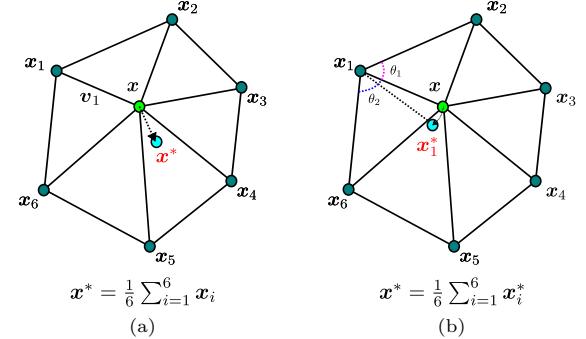
図1: a) ラプラシアン平滑化。点xのスター多角形は $S(x) = \{x_1, x_2, \dots, x_6\}$ を六角形とする。最適化されたメッシュノードは、StarPolygonのノードの座標の算術平均に位置する。b) 角度ベースの平滑化。図は $x_1$ に基づく最適位置を示している。これは、 $x$ を $x_1$ の位置にある角度の角度二等分線に回転させることで、 $x_1^*$ が得られる。この処理をメッシュ内の他の点についても繰り返し、算術平均化により最終的に最適化された点を得る。

角度ベースの平滑化は、StarPolygonのノードの角度二等分線上にメッシュノードを配置することでメッシュ平滑化を実現します(図1bに示す)[13]。ノードベースの手法に加えて、Centroidal Voronoi tessellation (CVT) smoothing [14]は、Lloyd反復[15]によって各ノードのボロノイ領域を再計算し、平滑化のためにボロノイ領域の重心に点を再配置する。Lloydアルゴリズムの効率を高めるために、新しい点位置を計算する決定論的手法が提案されている[16]。ヒューリスティックに基づくメッシュ平滑化手法はシンプルで効率的であるが、その最適化能力は限られている。これらは逆要素につながる可能性があり、平滑化効果はヒューリスティック関数の設計に大きく依存する。一方、最適化ベースの手法は、局所領域におけるメッシュ品質評価指標を最適化することで、メッシュスマージングを実現する[17-20]。Parthasarathy and Kodiyalam [17]は、メッシュ平滑化問題を制約付き最適化問題として定式化し、メッシュノード位置を最適化するために反復最適化アルゴリズムを利用している。

# 1 Introduction

With the rapid advancement of computer technology, computational fluid dynamics (CFD) has emerged as a crucial method for studying the principles of fluid dynamics. Its wide applications span diverse fields, including aerospace, hydraulic engineering, automotive engineering, and biomedicine [1–4]. Normally, CFD simulations are performed by discretizing the governing physical equations and subsequently solving large-scale algebraic systems of discretized equations to obtain fluid variables. Discretization, a critical step in CFD, encompasses two key aspects: discretizing the governing physical equations and discretizing the computational domain [5]. The latter process, known as mesh generation, plays a fundamental role in CFD. It involves partitioning the computational domain into non-overlapping mesh elements, such as polygons in the two-dimensional region and polyhedra in the three-dimensional region [6]. The quality of the generated mesh profoundly impacts the convergence, accuracy, and efficiency of numerical simulations. Orthogonality, smoothness, distributivity and density distribution of mesh elements significantly influence the stability and convergence of the solution matrix [7]. Consequently, the quest for high-quality mesh generation remains a vibrant and active area in CFD research. During the practical mesh generation process, the initial generated mesh often fails to meet simulation requirements. To enhance the quality of the mesh, mesh quality improvement techniques are commonly employed, including mesh smoothing, face-swapping, edge-swapping, point insertion/deletion, and other techniques [8–10]. Among such techniques, mesh smoothing methods are the most commonly used approach to improve mesh quality.

Mesh smoothing can be categorized into two main types: heuristic smoothing and optimization-based smoothing [11]. One representative heuristic method is Laplacian smoothing [12] (shown in Figure 1a). In Laplacian smoothing, mesh node is placed at the arithmetic average of the coordinates of the nodes in the *StarPolygon* (a polyhedron containing the node, as shown in Figure 1) for smoothing. This method is efficient but may produce negative volume elements when the *StarPolygon* is non-convex. Angle-based smoothing achieves mesh smoothing by placing mesh node



**Fig. 1:** a) Laplacian smoothing. The *StarPolygon* of point  $\mathbf{x}$  is formed by  $\mathbf{S}(\mathbf{x}) = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_6\}$  as a hexagon. The optimized mesh nodes are then located at the arithmetic average of the coordinates of the nodes in the *StarPolygon*. b) Angle-Based smoothing. The figure shows the optimal position based on  $\mathbf{x}_1$ . This is achieved by rotating  $\mathbf{x}$  to the angle bisector of the angle where  $\mathbf{x}_1$  is located, resulting in  $\mathbf{x}_1^*$ . This process is repeated for other points in the mesh, and the final optimized points are obtained through arithmetic averaging.

on the angle bisectors of the nodes of the *StarPolygon* (shown in Figure 1b) [13]. In addition to node-based methods, Centroidal Voronoi tessellation (CVT) smoothing [14] recalculates the Voronoi regions of each node through Lloyd iterations [15] and relocates the point to the centroid of the Voronoi region for smoothing. To enhance the efficiency of the Lloyd algorithm, deterministic methods have been proposed for calculating the new point location [16]. Although heuristic-based mesh smoothing methods are simple and efficient, their optimization capabilities are limited. They may lead to inverted elements, and the smoothness effect heavily relies on the design of heuristic functions. On the other hand, optimization-based methods achieve mesh smoothing by optimizing the mesh quality evaluation metric in the local area [17–20]. Parthasarathy and Kodiyalam [17] formulate the mesh smoothing problem as a constrained optimization problem and utilize iterative optimization algorithms to optimize the mesh node positions. Despite the adoption of different mesh quality evaluation metrics and optimization

その後の研究で異なるメッシュ品質評価指標と最適化手法が採用されているにもかかわらず、最適化ベースの手法は通常、メッシュスマージングのために最適化問題を反復的に解く必要があり、その結果、効率が低くなる。

近年、人工知能(AI)手法もメッシュ関連分野で広く利用されている。これらの研究成果の多くは、メッシュ品質評価[21-23]、メッシュ密度制御[24, 25]、メッシュ生成[26-28]、メッシュ精密化[29, 30]、メッシュアダパテーション[31-33]へのAI手法の適用に費やされている。しかし、AIベースのメッシュスマージングに関する研究は比較的少ない。Guoら[11]は、フィードフォワードニューラルネットワークを用いて最適化ベースの平滑化手法を模倣する教師あり学習アプローチを最初に導入した。提案モデルであるNN-Smoothingは、最適なノード位置を直接与えることで、最適化ベースのスマージングの効率を向上させる。しかし、フィードフォワードニューラルネットワークは固定次元の入力に悩まされ、異なる次数のメッシュノードに対しては別々のモデルが必要となり、異なる入力ノードのシーケンスに対してはデータ増強が必要となり、それによってモデルの学習コストが増加する。さらに、教師あり学習でモデルを学習させるために、高品質なメッシュ生成は、負担の大きい計算オーバーヘッドも発生させる。

このような制限を克服するために、本論文ではグラフニューラルネットワーク(GNN)[34]に基づく新しいメッシュスマージングモデルGMSNetを提示する。メッシュ平滑化の過程を学習するための軽量で効率的なGNNモデルを提案する。GMSNetは、メッシュノードの隣接ノードの特徴を抽出することで、最適化問題を解く際のオーバーヘッドを回避し、より良いノード位置を直接出力する。グラフニューラルネットワークが非構造化データを扱う能力により、GMSNetは単一のモデルで様々な程度のノードを平滑化でき、データ増強なしでノード入力シーケンス問題をエレガントに解決できることを示す。一度学習すれば、異なる形状の滑らかなメッシュに適用することができる。また、メッシュを平滑化する際に負の体積要素を導入しないために、シフト切り捨て操作を提案する。提案するGMSNetにとどまらず、メッシュ品質メトリクスに基づく新しい損失関数MetricLossを導入し、モデルを学習することで、高品質なメッシュを生成するオーバーヘッドをさらに排除する。GMSNetと一般的に使用されるメッシュ平滑化アルゴリズムについて、2次元三角メッシュ上で広範な実験を行う。

実験結果は、我々のモデルが最適化ベースの平滑化と比較して8.62倍の高速化を達成し、同時に同程度の性能を達成し、他の全てのヒューリスティック平滑化アルゴリズムを凌駕することを示している。また、GMSNetは学習時に未見であったメッシュにも適用できることが示された。一方、従来のNN-Smoothingモデルと比較すると、GMSNetはモデルパラメータが5%しかないが、メッシュスマージング性能は優れている。また、提案するMetricLossとシフト切り捨ての有効性を比較実験により検証する。我々の貢献をまとめると以下のようになる：

1. インテリジェントメッシュスマージングのための軽量グラフニューラルネットワークモデルGMSNetを提案する。GMSNetは程度の差こそあれノードを平滑化することができ、データ入力順序の影響を受けない。さらに、GMSNetが負の体積要素を生成するのを防ぐために、フォールトトレランス機構であるシフトトランケーションを提供する。
2. メッシュ品質メトリクスに基づき、高品質なメッシュを必要としないモデルを学習するための新しい損失関数MetricLossを導入する。MetricLossはモデル学習中に安定した収束を示す。
3. 2次元三角メッシュを用いた広範な実験により、GMSNetの有効性を検証する。実験結果は、GMSNetが優れたメッシュスマージング性能を達成し、平均8.62倍のスピードアップで最適化ベースのスマージングを大幅に上回ることを示している。また、提案するMetricLossとシフト切り捨て操作の有効性を示すために、比較実験を行った。

本稿の残りの部分は以下のように構成されている。セクション2では、一般的に使用されているメッシュ平滑化手法と、メッシュ分野におけるニューラルネットワークの応用を紹介する。セクション3では、提案モデルを紹介し、メッシュデータの前処理、モデルのアーキテクチャ設計、損失関数、学習方法について詳細に説明する。セクション4では、提案モデルの性能をベースラインモデルと比較する実験を行い、損失関数とシフト切り捨ての有効性について議論する。

methods in subsequent works, the optimization-based methods usually require solving optimization problems iteratively for mesh smoothing, resulting in low efficiency.

Recently, Artificial Intelligence (AI) methods have also been widely used in mesh-related fields. Most of these research works are devoted to applying AI methods to mesh quality evaluation [21–23], mesh density control [24, 25], mesh generation [26–28], mesh refinement [29, 30], mesh adaptation [31–33]. However, there is relatively little work on AI-based mesh smoothing. Guo et al. [11] firstly introduced a supervised learning approach to imitate optimization-based smoothing methods with feedforward neural networks. The proposed model, NN-Smoothing, improves the efficiency of optimization-based smoothing by directly giving the optimal node position. However, feedforward neural networks suffer from fixed-dimensional inputs, necessitating separate models for mesh nodes with different degrees and data augmentation for different sequences of input nodes, thereby increasing the model’s training cost. Moreover, to train the model through supervised learning, high-quality mesh generation also incurs burdensome computational overhead.

To overcome such limitations, we present a novel mesh smoothing model, GMSNet, based on graph neural networks (GNNs) [34] in this paper. We propose a lightweight and efficient GNN model to learn the process of mesh smoothing. GMSNet avoids the overhead of solving the optimization problem by extracting features of the neighboring nodes of the mesh node to directly output better node position. We show that through the ability of graph neural networks to handle unstructured data, GMSNet can smooth nodes of varying degrees with a single model and elegantly solves the node input sequence problem without data augmentation. Once trained, it can be applied to smooth mesh with different shapes. We also propose a shift truncation operation to avoid introducing negative volume elements when smoothing the mesh. Beyond the proposed GMSNet, we introduce a novel loss function, MetricLoss, based on the mesh quality metrics to train the model, which further eliminates the overhead of generating high-quality meshes. We conduct extensive experiments among GMSNet and commonly used mesh smoothing algorithms on two-dimensional triangular meshes. The experimental

results show that our model achieves a speedup of 8.62 times compared with optimization-based smoothing while achieving similar performance, and outstands all the other heuristic smoothing algorithms. The results also indicate that GMSNet can be applied to meshes that were unseen during training. Meanwhile, compared to previous NN-Smoothing model, GMSNet has only 5% of its model parameter, but obtains superior mesh smoothing performance. We also validate the effectiveness of proposed MetriLoss and shift truncation with comparative experiments. We summary our contributions as follows:

1. We propose a lightweight graph neural network model, GMSNet, for intelligent mesh smoothing. GMSNet can smoothing node with varying degrees and remain unaffected by the data input order. Additionally, we offer a fault-tolerance mechanism, shift truncation, to prevent GMSNet from generating negative volume elements
2. Basing on the mesh quality metrics, we introduce a novel loss function, MetricLoss, to train the model without necessity for high-quality meshes. MetricLoss exhibits a stable and rapid convergence during model training.
3. We validate the effectiveness of GMSNet through extensive experiments conducted on two-dimensional triangular meshes. The experimental results demonstrate that GMSNet achieves excellent mesh smoothing performance and significantly outperforms optimization-based smoothing with a average speedup of 8.62 times. Comparative experiments are also conducted to showcase the effectiveness of the proposed MetricLoss and shift truncation operation.

The remaining parts of this paper are organized as follows. In Section 2, we introduce commonly used mesh smoothing methods and applications of neural networks in the field of mesh. In Section 3, we present the proposed model, providing detailed explanations of the mesh data preprocessing, design of the model’s architecture, loss function and training method. In Section 4, we conduct experiments to compare the performance of our proposed model with baseline models and discuss the effectiveness of loss function and shift truncation. Finally, in the conclusion section, we

最後に、結論のセクションでは、論文全体を要約し、将来的に探求できる可能性のある研究を提案する。

## 2 リアルワーク

### 2.1 ヒューリスティックメッシュ平滑化

ラプラスアンスマージングは、ヒューリスティックメッシュスマージング法として最もよく使われる方法である。StarPolygonのノードの算術平均にノード座標を更新する。重み付きラプラスアンスマージング[35]は、スマージングプロセス中に隣接するノードやエッジに重みや重要度係数を追加導入し、スマージング効果をより制御し、メッシュの特定の特徴を保持する。スマートラプラスアンスマージング[18]は、各ノードの移動の前にチェックを行い、操作によってメッシュ品質が向上するかどうかを評価する。メッシュ品質が向上しない場合、メッシュノードの動きがスキップされ、より効率的な処理となる。角度ベースのメッシュ平滑化は、メッシュノードの角度を考慮することで、滑らかさを実現する。この方法では、ノードを回転させ、StarPolygonの各ノードの角度二等分線に合わせる。スター・ポリゴンの様々なノードの角度二等分線は一致しない可能性があるため、最終的なノード位置は追加計算が必要である。これは、ノード座標の平均を計算するか、最小二乗問題を解くことで実現できる。CVTスマージングは、メッシュノードによって定義されたボロノイ領域のセントロイドにノードを配置する。解法プロセスにおけるLloydアルゴリズムの効率を高めるために、以下に述べるように、セントロイドを計算するより効率的な方法が設計されている：

$$\mathbf{x}_i^* = \frac{1}{|\Omega_i|} \sum_{T_j \in \Omega_i} |T_j| C_j \quad (1)$$

ここで、 $\mathbf{x}_i^*$ は新しいノードの位置を表し、 $|\Omega_i|$ はノードのスター・ポリゴンの総面積、 $|T_j|$ はj番目の三角形の面積、 $C_j$ はj番目の三角形の円周率である。

ヒューリスティック・スマージングと最適化ベースのスマージングの間に絶対的な境界がないことは注目に値する。別の観点からは、ヒューリスティックメッシュスマージングも最適化ベースのアプローチと見なすことができる。例えば、ラプラスアンスマージングは、

ここで、 $v_i$ は $x$ から $x_i$ への辺を表し、 $K$ はStarPolygonのノード数である(図1a)。同様に、角度に基づくメッシュ平滑化は、エネルギー関数 $E = \sum_{i=1}^{KPK} \theta_i^2$ を最小化すると見なすことができる。ここで、 $\theta_i$ は $x$ から $x_i$ までの辺と多角形の辺との間の角度である(図1bに示す)。ヒューリスティックメッシュスマージングの主な利点は、その効率にある。しかし、その平滑化性能は最適化ベースの平滑化性能に劣ることが多い。さらに、ヒューリスティックメッシュスマージングの有効性は、ヒューリスティック関数の設計に大きく依存する。

### 2.2 最適化に基づく平滑化

最適化に基づくメッシュ平滑化手法は、以下の問題として定式化できる：ノード位置 $x_i$ と制約条件の集合と目的関数 $f$ を持つ初期メッシュが与えられたとき、目的関数 $f$ を最小化する新しいノード位置 $x_i^*$ を見つけることが目的である。数学的には、これは次のように表すことができる：

$$\begin{aligned} \mathbf{x}_i^* &= \arg \min_{\mathbf{x}_i} f(\mathbf{x}_i, \mathbf{S}(\mathbf{x}_i)) \\ \text{s.t. } \mathbf{x}_i &\in \mathcal{X} \end{aligned} \quad (2)$$

エネルギー関数 $E = \sum_{i=1}^{KPK} |v_i|^2$ を最小化すると見なすことができる、 $f$ はメッシュ品質評価関数、 $X$ は $x_i^*$ の制約を満たす実行可能集合、 $S(x_i)$ は $x_i$ のスター・ポリゴンのノード集合である。この関数の入力にはノード間の接続性も含まれていることを述べておくが、ここではわかりやすくするために省略する。評価関数の選択が異なると、メッシュの品質が異なることを反映して、メッシュの平滑化方法が異なる。一般的に使用される評価関数には、最大最小角度[36]、リスペクト比、歪み比などがある[17]。関数 $f$ が微分可能であれば、勾配 $\Delta f = 0$ とすることで、この制約付き最適化問題の最適点を解いて、明示的な式を得ることができる。しかし、ほとんどの場合、ラプラスアン平滑化や角度ベース平滑化のように $x_i^*$ の明示的な式を導くことは難しい。 $x_i^*$ の解法には反復法がよく使われるが、これは効率が悪い。したがって、最適なポジションを効率的に解く方法を開発することは、取り組むべき問題である。

summarize the entire paper and propose potential future work that can be explored.

## 2 Realted work

### 2.1 Heuristic mesh smoothing

Laplacian smoothing is the most commonly used heuristic mesh smoothing method. It updates the node coordinate to the arithmetic average of nodes in *StarPolygon*. Weighted Laplacian smoothing [35] introduces additional weights or importance factors to the neighboring nodes or edges during the smoothing process, offering more control over the smoothing effect and preserving specific features of the mesh. Smart Laplacian smoothing [18] includes a check before each node movement to assess whether the operation will improve the mesh quality. If the mesh quality does not improve, the movement of the mesh node is skipped, resulting in a more efficient process. Angle-based mesh smoothing achieves smoothness by considering the angles of the mesh nodes. In this method, the node is rotated to align with the angle bisectors of each node in *StarPolygon*. Since the angle bisectors of the various nodes of the *StarPolygon* may not coincide, the final node positions require additional calculation. This can be achieved by calculating the average of the node coordinates or by solving a least squares problem. CVT smoothing positions the node at the centroids of the Voronoi region defined by the mesh node. To enhance the efficiency of the Lloyd algorithm in the solving process, more efficient method has been designed to calculate the centroids, as described below:

$$\mathbf{x}_i^* = \frac{1}{|\Omega_i|} \sum_{T_j \in \Omega_i} |T_j| C_j \quad (1)$$

where  $\mathbf{x}_i^*$  represents the new node position,  $|\Omega_i|$  is the total area of the node's *StarPolygon*,  $|T_j|$  is the area of the  $j$  th triangle, and  $C_j$  is the circumcenter of the  $j$  th triangle.

It is worth noting that there is no absolute boundary between heuristic smoothing and optimization-based smoothing. From another perspective, heuristic mesh smoothing can also be viewed as an optimization-based approach. For instance, Laplacian smoothing can be viewed as minimizing the energy function  $E = \frac{K}{2} \sum_{i=1}^K |\mathbf{v}_i|^2$ ,

where  $\mathbf{v}_i$  represents the edge from  $\mathbf{x}$  to  $\mathbf{x}_i$ , and  $K$  is the number of nodes in the *StarPolygon* (shown in Figure 1a). Similarly, Angle-based mesh smoothing can be seen as minimizing the energy function  $E = \frac{K}{2} \sum_{i=1}^{2K} \theta_i^2$ , where  $\theta_i$  is the angle between the edge from  $\mathbf{x}$  to  $\mathbf{x}_i$  and the edge of the polygon (shown in Figure 1b). The primary advantage of heuristic mesh smoothing lies in its efficiency. However, its smoothing performance often is inferior to that of optimization-based smoothing. Furthermore, the effectiveness of heuristic mesh smoothing heavily relies on the design of the heuristic functions.

### 2.2 Optimization-based smoothing

The mesh smoothing method based on optimization can be formalized as the following problem: Given an initial mesh with node positions  $\mathbf{x}_i$  and a set of constraints and the objective function  $f$ , the goal is to find a new node positions  $\mathbf{x}_i^*$  that minimizes the objective function  $f$ . Mathematically, this can be expressed as:

$$\begin{aligned} \mathbf{x}_i^* &= \arg \min_{\mathbf{x}_i} f(\mathbf{x}_i, \mathbf{S}(\mathbf{x}_i)) \\ \text{s.t. } \mathbf{x}_i &\in \mathcal{X} \end{aligned} \quad (2)$$

where  $\mathbf{x}_i^*$  represents the new position of the mesh node  $\mathbf{x}_i$ ,  $f$  is some mesh quality evaluation function,  $\mathcal{X}$  is the feasible set satisfying constraints on  $\mathbf{x}_i^*$ , and  $\mathbf{S}(\mathbf{x}_i)$  is the set of nodes of the *Star-Polygon* of  $\mathbf{x}_i$ . It is important to mention that the input of this function also includes the connectivity between nodes, which is omitted here for clarity. Different choices of evaluation functions lead to different mesh smoothing methods, reflecting our emphasis on different mesh qualities. Commonly used evaluation functions include the maximum minimum angle [36], respect ratio, and distort ratio, among others [17]. If the function  $f$  is differentiable, the optimal point of this constrained optimization problem may be solved by setting the gradient  $\nabla f = 0$  to obtain an explicit expression. However, in most cases, it is difficult to derive explicit expressions for  $\mathbf{x}_i^*$  as in Laplacian smoothing and Angle-based smoothing. Iterative methods are often used to solve  $\mathbf{x}_i^*$ , which are of low efficiency. Therefore, developing an efficient way to solve the optimal positions is a problem that needs to be addressed.

## 2.3 メッシュ関連分野のニューラルネットワーク

ヒトの脳のニューロンからヒントを得て、人工ニューラルネットワークを利用して複雑な機能マッピングを学習する。彼らは、画像認識、音声認識、自然言語処理などのタスクに取り組み、機械学習や人工知能における広範なアプリケーションを見出している[37-40]。近年、ニューラルネットワークはメッシュに関連する様々な領域で重要な応用が見出されている。メッシュ品質評価の領域では、構造化メッシュ品質の自動評価を容易にするために、NACA-Marketデータセットとともに、畳み込みニューラルネットワークモデルであるGridNetがChenら[21]によって導入された。この概念を非構造化メッシュに拡張し、Wangら[22]はメッシュ品質評価にグラフニューラルネットワークを採用した。メッシュ分布の精密化を追求するために、Zhangら[24]は、従来のメッシュ生成ソフトウェアを強化するために人工ニューラルネットワークを採用し、領域全体の局所メッシュ密度の予測を可能にした。このアプローチは、広範なテストによって説得力のある実証がなされたように、四面体メッシュにさらに拡張された[25]。インテリジェントメッシュ生成の領域では、Daroyaら[26]が、点群からのグローバルな構造情報を活用して、高品質なメッシュ再構成を実現するアルゴリズムを発表した。同様に、Papageannopoulosら[27]は、メッシュ化された輪郭から抽出されたデータを利用してニューラルネットワークを学習し、メッシュ領域内のノードの数、配置、相互接続性を正確に近似することを可能にした。Chenら[28]は、新しい差分アプローチを用いて、構造化メッシュを生成するための教師なしニューラルネットワーク手法であるMGNetを導入し、有望な結果を得た。人工知能技術は、生成だけでなく、メッシュの改良や適応にも大きく貢献している。Bohn and Feischl [29]は、最適なメッシュ精密化アルゴリズムを学習するためリカレントニューラルネットワークを採用し、広範な偏微分方程式を強化するための効果的なブラックボックスツールとしての能力を確立した。変分メッシュ適応を強化し、Tingfanら[31]は、更新されたメッシュ上の流れ場推定を迅速化するために、機械学習回帰モデルをシームレスに統合した。

一方、Wallworkら[32]は、学習されたニューラルネットワークに支えられた、データ駆動型の目標指向メッシュ適応戦略を考案し、適応プロセスにおける計算コストのかかる誤差推定フェーズに効果的に取って代わる。さらに、FidkowskiとChen[33]は、計算メッシュの最適な異方性を確認するために、独創的に人工ニューラルネットワークを採用し、従来の方法と比較してメッシュ効率を向上させた。メッシュスマージングに関しては、NN-Smoothing [11]がフィードフォワードニューラルネットワークを用いた最適化ベースのメッシュスマージングを模倣し、最適化ベースのメッシュスマージングの効率を大幅に向上させている。しかし、別々のモデルの学習と高価な高品質メッシュ生成には、かなりの計算オーバーヘッドが発生する。

従来のディープラーニングアルゴリズムに加え、非構造化データを扱うための人工知能手法の学習能力を向上させるために、グラフニューラルネットワーク[34]が導入された。GNNはグラフ畳み込み[41]を利用して、グラフ上の特徴学習プロセスにトポロジカルな接続を組み込んでいる。メッシュはグラフデータとして自然に表現できるため、GNNはメッシュの精密化、流れ場のシミュレーション、乱流モデリングなど、様々な計算流体力学分野で広範な応用が見出されている[42-46]。したがって、メッシュスマージングにGNNを適用することは、有望かつ効果的な解決策であると主張する。

## 3 Methodology

### 3.1 問題の定式化

メッシュスマージング問題は、ノード間の接続性を維持しながら、そのノードの位置を調整することで、メッシュの品質を向上させることを含む。メッシュスマージング処理は、メッシュノードとその接続を入力とし、メッシュノードに新しい座標を出力として与え、メッシュ上で動作する関数として定義される。各メッシュノードについて、平滑化関数の入力は、それ自身と、そのノードの1リング近傍を構成するそのStarPolygonである。本論文では、メッシュをノードグラフとして定義する。具体的には、ノード $x_0$ とそのスター polygon が与えられたとき、それをグラフ $G = (V, E)$ で表現する。

## 2.3 Neural networks in mesh-related fields

Inspired by the neurons in the human brain, artificial neural networks are utilized to learn complex function mappings. They find extensive applications in machine learning and artificial intelligence, addressing tasks like image recognition, speech recognition, natural language processing, and more [37–40]. Recently, neural networks have found significant application in various domains related to meshes. In the realm of mesh quality evaluation, GridNet, a convolutional neural network model, was introduced by Chen et al. [21], along with the NACA-Market dataset, to facilitate automated evaluation of structured mesh quality. Extending this notion to unstructured meshes, Wang et al. [22] employed graph neural networks for mesh quality assessment. In the pursuit of refining mesh distribution, Zhang et al. [24] employed an artificial neural network to enhance conventional mesh generation software, enabling prediction of local mesh density throughout the domain. This approach was further expanded to tetrahedral meshes, as demonstrated convincingly through extensive testing [25]. In the domain of intelligent mesh generation, Daroya et al. [26] presented an algorithm that leverages global structural information from point clouds to achieve high-quality mesh reconstruction. Similarly, Papagiannopoulos et al. [27] harnessed data extracted from meshed contours to train neural networks, enabling accurate approximation of the number, placement, and interconnectivity of nodes within the meshing domain. Taking a novel differential approach, Chen et al. [28] introduced MGNet, an unsupervised neural network methodology for generating structured meshes, yielding promising results. Beyond generation, artificial intelligence techniques have also made significant contributions to mesh refinement and adaptation. Bohn and Feischl [29] employed recurrent neural networks to learn optimal mesh refinement algorithms, establishing its prowess as an effective black-box tool for enhancing a wide spectrum of partial differential equations. Enriching variational mesh adaptation, Tingfan et al. [31] seamlessly integrated a machine learning regression model to expedite flow field estimation on updated meshes. Meanwhile, Wallwork et al. [32]

devised a data-driven goal-oriented mesh adaptation strategy, underpinned by a trained neural network, effectively supplanting the computationally expensive error estimation phase in the adaptation process. Furthermore, Fidkowski and Chen [33] ingeniously employed Artificial Neural Networks to ascertain optimal anisotropy in computational meshes, yielding enhanced mesh efficiency in comparison to conventional methods. In term of mesh smoothing, NN-Smoothing [11] imitates optimization-based mesh smoothing using feedforward neural networks, significantly enhancing the efficiency of optimization-based mesh smoothing. However, separate models training and expensive high-quality mesh generation incur significant computational overhead.

In addition to conventional deep learning algorithms, graph neural networks [34] were introduced to enhance the learning capacity of artificial intelligence methods for handling non-structured data. GNNs utilize graph convolutions [41] to incorporate the topological connections in the feature learning process on graphs. As meshes can be naturally represented as graph data, GNNs have found extensive applications in various computational fluid dynamics fields, including mesh refinement, flow field simulation, turbulence modeling, and more [42–46]. Therefore, we argue that applying GNNs to mesh smoothing is a promising and effective solution.

## 3 Methodology

### 3.1 Problem formulation

The mesh smoothing problem involves enhancing the quality of a mesh by adjusting the positions of its nodes while maintaining the connectivity between them. Mesh smoothing processes are defined as functions that operate on the mesh, taking the mesh nodes and their connections as inputs and providing new coordinates for the mesh nodes as outputs. For each mesh node, the input of the smoothing function is itself and its *StarPolygon*, which comprises the node's one-ring neighbors. In this paper, we define the mesh as a node graph. Specifically, given a node  $\mathbf{x}_0$  and its *StarPolygon*, we represent it with a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , where  $\mathcal{V} = \{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$  represents the node and nodes in the *StarPolygon*,  $\mathbf{x}_i$  is the coordinate of node  $i$  in *StarPolygon*,  $n$  is the number

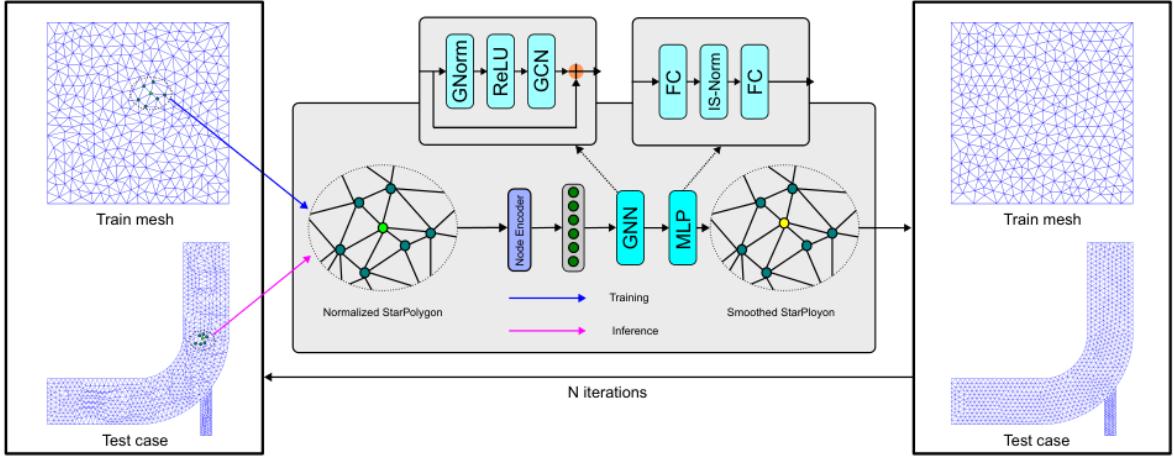


図2:GMSNetのアーキテクチャGMSNetは、各メッシュノードに対して最適化された位置を計算することにより、メッシュを平滑化する。このプロセスは、入力されたStarPolygonの正規化から始まる。次に、正規化された特徴量に対して特徴量変換を行い、StarPolygonノードからの情報をグラフ畳み込みにより統合する。最後に、モデルは完全連結(FC)層を通して最適化されたノード位置を予測する。図中、GNormはGraphNorm演算を表し、IS-NormはInstanceNorm演算を表す。モデルはメッシュ全体に対してN回反復し、平滑化操作を実行する。学習後、GMSNetはパイプのメッシュのような、以前に見たことのないメッシュに適用することができる。

$x_n$  }はスター polygon のノードとノードを表し、 $x_i$ はスター polygon のノード  $i$  の座標、 $n$ はスター polygon のノード数、 $E = \{(i, j) \mid \text{if } x_i \text{ is connected with } x_j\}$  はノード間の接続を表す。典型的な平滑化処理は反復処理であり、 $t$  回目の反復ステップで、初期ノードグラフを  $G_t$  とし、中心ノードの最適化されたノード位置は次のように表すことができる：

$$x_0^{t+1} = \mathcal{F}(G_t) = \mathcal{F}(\mathcal{V}_t, \mathcal{E}) \quad (3)$$

本論文では、メッシュ平滑化のための関数  $\mathcal{F}$  を学習するために、グラフニューラルネットワークを採用する。

## 3.2 GMSNet

### 3.2.1 軽量モデルの設計

グラフニューラルネットワークに基づくメッシュスマージングモデルを図2に示す。メッシュノードとエッジからなるグラフが与えられたとき、我々の目標は、メッシュを平滑化するために、各メッシュノードに対してより最適なメッシュノード位置を計算することである。モデルでは、座標位置をノード特徴量として用い、 $X \in \mathbb{R}^{(n+1) \times 2}$  とする(StarPolygonの  $n$  個のノードと移動するための1個の空きノード)。

ノード間の接続性は隣接行列  $A \in \{\mathbb{R}\}^{(n+1) \times (n+1)}$  で表される(ノードインデックス  $i$  はわかりやすくするために省略)。

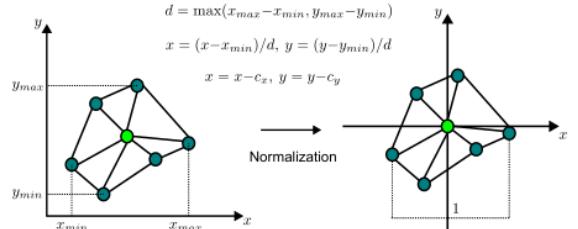
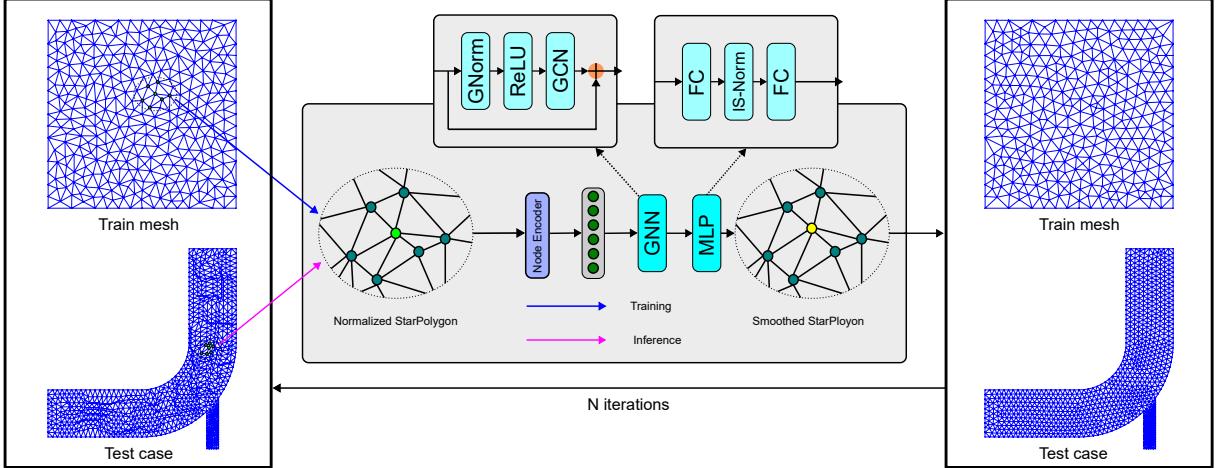


図3:データの正規化。処理するStarPolygonは座標軸の中心に対して正規化されている。図中、 $d$ はStarPolygonの最大サイズ、 $x$ と $y$ はスケーリングする座標、 $c_x$ と $c_y$ はスケーリング後のターゲットノードの正規化座標を表す。

モデルのスケール不変性を確保するため、処理中にノード正規化を適用する。このモデルは、 $X$ に対して min-max 正規化を用いてノード入力を正規化し、0と1の範囲に制限し、その後、座標原点を中心にノードをセンタリングする変換を実行する。



**Fig. 2:** The architecture of GMSNet. GMSNet smoothes the mesh by calculating the optimized position for each mesh node. The process begins with the normalization of the input *StarPolygon*. Next, feature transformation is performed on the normalized features, and information from the *StarPolygon* nodes is integrated using graph convolution. Finally, the model predicts the optimized node positions through fully connected (FC) layers. In the figure, **GNorm** represents the GraphNorm operation, and **IS-Norm** represents the InstanceNorm operation. The model iterates  $N$  times over the entire mesh to perform the smoothing operation. After training, GMSNet can be applied to previously unseen meshes, such as the pipe’s mesh.

of nodes in the *StarPolygon*, and  $\mathcal{E} = \{(i, j) \mid \text{if } \mathbf{x}_i \text{ is connected with } \mathbf{x}_j\}$  represents the connections between the nodes. A typical smoothing process is iterative, where at the  $t$  th iteration step, the initial node graph is denoted as  $\mathcal{G}_t$ , and the optimized node position for the center node can be represented as:

$$\mathbf{x}_0^{t+1} = \mathcal{F}(\mathcal{G}_t) = \mathcal{F}(\mathcal{V}_t, \mathcal{E}) \quad (3)$$

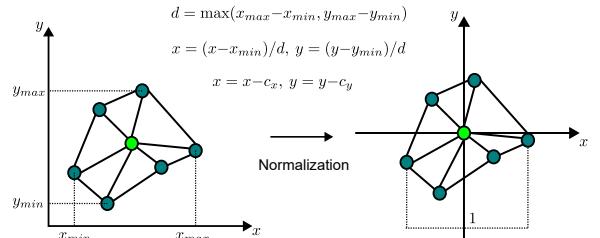
In this paper, we employ graph neural networks to learn the function  $\mathcal{F}$  for mesh smoothing.

## 3.2 GMSNet

### 3.2.1 Lightweight model design

The mesh smoothing model based on graph neural networks is depicted in Figure 2. Given a graph consisting of mesh nodes and edges, our goal is to compute a more optimal mesh node position for each mesh node to smoothing the mesh. In the model, we use the coordinate positions as node features, denoted as  $X \in \mathbb{R}^{(n+1) \times 2}$  ( $n$  nodes in the *StarPolygon* and one free node to move). The connectivity between nodes is represented by the

adjacency matrix  $A \in \mathbb{R}^{(n+1) \times (n+1)}$  (node index  $i$  is omitted for clarity).



**Fig. 3:** Data normalization. The *StarPolygon* to be processed is normalized to the center of the coordinate axis. In the figure,  $d$  represents the maximum size of the *StarPolygon*,  $x$  and  $y$  represent the coordinates to be scaled, and  $c_x$  and  $c_y$  are the normalized coordinates of the target node after the scaling.

To ensure the model’s scale invariance, we apply node normalization during processing. The model normalizes the node input using min-max normalization on  $X$ , restricting it to the range of 0 and 1, and subsequently performs a translation to center the node around the coordinate origin.

モデル処理後、アフィン変換を採用し、ノードの位置を元のスケールに戻す。データの正規化値を図3に示す。データを正規化した後、特徴量を線形層で変換し、次のように表すことができる。

$$X_h = \text{Norm}(X) \mathbf{W}_l + \mathbf{b}_l \quad (4)$$

ここで、 $\text{Norm}$ は正規化演算を表し、 $\mathbf{W}_l \in \mathbb{R}^{2 \times H}$ 、 $\mathbf{b}_l$ は線形層の変換行列とバイアス、 $H$ は隠れ特徴の次元、 $X_h \in \mathbb{R}^{(n+1) \times H}$ は線形層の出力である。その後、残差グラフ畳み込みネットワーク(GCN)層[41, 47]を採用し、StarPolygonのノードの特徴に基づいて隠れ層の特徴を計算する。このプロセスでは、GraphNorm [48]を使用して線形層が出力する特徴を正規化し、その後、活性化層、畳み込み層、およびノードの隠れた特徴を計算するための和層が続く。この過程は次のように表すことができる：

$$\hat{X}_h = \text{GraphNorm}(X_h) \quad (5)$$

$$X_g = \text{GCN}(\text{ReLU}(\hat{X}_h), \tilde{A}) + \hat{X}_h \quad (6)$$

$$= \tilde{A}[\text{ReLU}(\hat{X}_h)]\mathbf{W}_g + \hat{X}_h \quad (7)$$

ここで、 $\tilde{A}$ は正規化隣接行列<sup>1</sup>、ReLUは活性化関数、 $\mathbf{W}_g$ はGCN層のパラメータ、 $X_g$ は残差GCN層が outputする最終特徴量を表す。中心ノードの最終位置は、InstanceNorm [49]を用いた2層の完全連結ニューラルネットワークによって得られる。中心ノードのインデックスを $i_c$ とすると、最適化されたノード位置は次式で与えられる：

$$\mathbf{x}^* = \text{MLP}(X_g)[i_c] \quad (8)$$

平滑化アルゴリズムは、程度の差こそあれノードを処理する能力を持ち、ノード入力の順序に影響されないようにする必要がある。NN-Smoothingモデルでは、異なる次数のノードを別々のモデルで学習することで処理し、ノード入力の順序は、StarPolygonのリング内の開始ノードを変化させることでデータ増強することで対処する。

---

<sup>1</sup> $\hat{A} = A + I$ ,  $\hat{D}$  is the degree matrix of  $A$ , and  $\tilde{A} = \hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}}$

しかし、グラフニューラルネットワークモデルの順列不变性により、入力順序(和関数や平均関数など)の順列にもかかわらず出力が変化しないため、GMSNetは別々のモデルを学習したり、データ補強を行うことなく、程度の異なるノードを効果的に扱うことができる。

### 3.2.2 モデル学習

NN-Smoothing法では、教師あり学習でモデルを学習し、最適化ベースの平滑化でラベル付き高品質メッシュを生成するが、これは時間のかかる作業である。一方、提案するGMSNetは、メッシュ平滑化のための最適化処理を直接学習する。このプロセスは、最適化ベースのスマージングと比較することで説明できる。最適化ベースの平滑化では、勾配下降を最適化手法とし、最適化されるメッシュ品質関数 $f$ が与えられると、その最小値を求めることが目的である。 $k$ 回目の反復において、最適化アルゴリズムは位置を $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_x f(\mathbf{x}_k, S(\mathbf{x}_0))$ として更新する。ここで、 $\Delta_x f(\mathbf{x}_k, S(\mathbf{x}_0))$ は $f$ の勾配、 $\alpha$ はステップサイズである。 $\mathbf{x}_k$ を繰り返し更新することで、関数は局所最適または大域最適  $\mathbf{x}^*$  に収束する。一方、NNSmoothingはメッシュ平滑化の最適点を直接予測する、すなわち、 $\mathbf{x}^* = \text{NN}(\mathbf{x}_0, S(\mathbf{x}_0))$ となる。しかし、このアプローチでは、最適化アルゴリズムによって生成されたラベル付き高品質メッシュが必要であり、時間がかかる。一方、GMSNetモデルは、メッシュノードの平滑化処理を学習するために、ラベル付きデータを必要としない。代わりに、学習プロセスはメッシュ要素の品質評価指標によって駆動され、それは次のように表すことができる：

$$\begin{aligned} \mathbf{W}^* &= \arg \min_{\mathbf{W}} \mathcal{L}(\hat{\mathbf{x}}^*, S(\mathbf{x}_0)) \\ &= \arg \min_{\mathbf{W}} L(F_W(\mathbf{x}_0, S(\mathbf{x}_0)), S(\mathbf{x})) \end{aligned} \quad (9)$$

ここで、 $L$ はメッシュ品質指標を用いて構築された損失関数、 $F_W$ は提案モデルを通して学習され、 $\mathbf{W}$ はモデルのパラメータである。損失関数はメッシュ要素の評価指標に基づいており、モデルはこの関数を最小化することでメッシュノードの位置を最適化する。

After model processing, we employ an affine transformation to map the node positions back to their original scale. Data normalization is illustrated in Figure 3. After normalizing the data, we transform the features by a linear layer, which can be expressed as

$$X_h = \text{Norm}(X) \mathbf{W}_l + \mathbf{b}_l \quad (4)$$

where  $\text{Norm}$  represents the normalization operation,  $\mathbf{W}_l \in \mathbb{R}^{2 \times H}$  and  $\mathbf{b}_l$  is the transformation matrix and bias of the linear layer,  $H$  is the dimension of the hidden feature, and  $X_h \in \mathbb{R}^{(n+1) \times H}$  is the output of the linear layer. Subsequently, we employ a residual graph convolutional network (GCN) layer [41, 47] to compute the features of the hidden layer based on the features of the nodes in the *StarPolygon*. The process involves normalizing the features outputted by the linear layer using GraphNorm [48], followed by an activation layer, a convolutional layer, and a summation layer to calculate the hidden features of the nodes. This process can be represented as:

$$\hat{X}_h = \text{GraphNorm}(X_h) \quad (5)$$

$$X_g = \text{GCN}(\text{ReLU}(\hat{X}_h), \tilde{A}) + \hat{X}_h \quad (6)$$

$$= \tilde{A}[\text{ReLU}(\hat{X}_h)]\mathbf{W}_g + \hat{X}_h \quad (7)$$

where  $\tilde{A}$  is the normalized adjacency matrix<sup>1</sup>, ReLU is activation function,  $\mathbf{W}_g$  is the parameter of GCN layer, and  $X_g$  represents the final features output by the residual GCN layer. The final position of the center node is obtained through a two-layer fully connected neural network with InstanceNorm [49]. Let  $i_c$  be the index of the center node, then the optimized node position is given by:

$$\mathbf{x}^* = \text{MLP}(X_g)[i_c] \quad (8)$$

The smoothing algorithm should possess the capability to handle nodes with varying degrees and remain unaffected by the order of node inputs. In the NN-Smoothing model, handling nodes with different degrees is achieved by training separate models, while the order of node inputs is addressed through data augmentation by varying the starting node in the ring of *StarPolygon*.

---

<sup>1</sup> $\hat{A} = A + I$ ,  $\hat{D}$  is the degree matrix of  $\hat{A}$ , and  $\tilde{A} = \hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}$

However, due to the permutation invariance property of graph neural network models, where the output remains unchanged despite permutations in the input order (such as in the sum function or mean function), GMSNet can effectively handle nodes with different degrees without the need for training separate models or performing data augmentation.

### 3.2.2 Model training

In the NN-Smoothing method, the model is trained using supervised learning, and the labeled high-quality meshes are generated using optimization-based smoothing, which is a time-consuming task. In contrast, the proposed GMSNet directly learns the optimization process for mesh smoothing. This process can be illustrated by comparing it with optimization-based smoothing. In optimization-based smoothing, taking gradient descent as the optimization method and given a mesh quality function  $f$  to be optimized, the objective is to find its minimum value. In the  $k$  th iteration, the optimization algorithm updates the position as  $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha \nabla_{\mathbf{x}} f(\mathbf{x}_k, \mathbf{S}(\mathbf{x}_0))$ , where  $\nabla_{\mathbf{x}} f(\mathbf{x}_k, \mathbf{S}(\mathbf{x}_0))$  is the gradient of  $f$  and  $\alpha$  is the step size. By iteratively updating  $\mathbf{x}_k$ , the function converges to a local or global optimum  $\mathbf{x}^*$ . In contrast, NN-Smoothing directly predicts the optimal point for mesh smoothing, i.e.,  $\hat{\mathbf{x}}^* = \text{NN}(\mathbf{x}_0, \mathbf{S}(\mathbf{x}_0))$ . However, this approach requires labeled high-quality meshes generated through the optimization algorithm, which is time-consuming. On the other hand, the GMSNet model does not require labeled data to learn the smoothing process for mesh nodes. Instead, the training process is driven by the quality evaluation metric of the mesh elements, which can be expressed as:

$$\begin{aligned} \mathbf{W}^* &= \arg \min_{\mathbf{W}} \mathcal{L}(\hat{\mathbf{x}}^*, \mathbf{S}(\mathbf{x}_0)) \\ &= \arg \min_{\mathbf{W}} \mathcal{L}(\mathcal{F}_{\mathbf{W}}(\mathbf{x}_0, \mathbf{S}(\mathbf{x}_0)), \mathbf{S}(\mathbf{x}_0)) \end{aligned} \quad (9)$$

where  $\mathcal{L}$  is the loss function constructed using a mesh quality metric,  $\mathcal{F}_{\mathbf{W}}$  is learned through the proposed model, and  $\mathbf{W}$  is the parameters of the model. The loss function is based on the mesh element evaluation metric, and the model optimizes the positions of mesh nodes by minimizing this function.

表1:最適化ベースのスムージング、NN-Smoothing、GMSNetの比較

Method	Speed	Labeled high-quality mesh	ノードの度数を変化させる	Node input order
OptimSmoothing <sup>1</sup>	Slow	Not acquiring	Not affected	Not affected
NN-Smoothing	Fast	Acquiring	Training separate models	Data augmentation
GMSNet	Fast	Not acquiring	Not affected	Not affected

<sup>1</sup> 最適化ベースの平滑化

このアプローチとNN-Smoothingの主な違いは、学習データにある。提案手法では、ラベルに高品質なメッシュは提供されない。その代わりに、学習プロセスはメッシュ品質メトリック関数の最小化のみに依存する。最適化ベースのメッシュスムージング手法との主な相違点は、このアプローチが最適化問題を解く必要なく、メッシュノードの最適化された位置を直接提供することであり、その結果、メッシュスムージングの効率が大幅に改善される。前述の3つの手法の包括的な比較を表1に示す。

### 3.2.3 メトリックロス

メッシュ要素の品質を評価するために、最大角度、最小角度、ヤコビアン行列、アスペクト比など、いくつかのメトリクスが利用可能である。我々の場合、メッシュ品質を評価するためにアスペクト $m = n / l$  + 1, ratio  $q = \sqrt{m^2 + n^2 + l^2}$  を採用し、 $4 \leq m, n, l \leq 1$  は三角形の辺、 $S$ は三角形の面積である。正三角形の場合、この値は1であるが、縮退した三角形の場合、 $+\infty$ に近づく。しかし、この指標の範囲が大きすぎたため、特に形状の悪い要素では勾配爆発を引き起こす可能性がある。この問題に対処するため、メッシュ要素の評価指標として $1 - \frac{1}{q}$  を用いる。正三角形の場合、この値は0であり、縮退三角形の場合、1である。損失関数の正式な定義は以下の通りである：

$$\begin{aligned} \mathcal{L}(x, S(x)) &= \frac{1}{|S(x)|} \sum_{i=1}^{|S(x)|} \left(1 - \frac{1}{q_i}\right) \\ &= \frac{1}{|S(x)|} \sum_{i=1}^{|S(x)|} \left(1 - \frac{4\sqrt{3}S_i}{m_i^2 + n_i^2 + l_i^2}\right) \end{aligned} \quad (10)$$

ここで、 $|S(x)|$ はStarPolygonのノード数、 $m_i, n_i, l_i$

{i}は三角形 $T_i$ の辺、 $q_i$ は $T_i$ の品質である。セクション4.4で、我々が設計した損失関数の有効性を検証した。

### 3.2.4 シフトの切り捨て

メッシュ生成とメッシュ平滑化の過程では、生成されたメッシュが負の体積要素を回避することが重要である。しかし、メッシュスムージングアルゴリズムは、図4に描かれているように、負の体積要素の生成につながることがある。例えば、ラプラシアンスムージングの場合、非凸のスター・ポリゴンを扱うと、負の体積要素が発生することがある（図4aに示すように）。同様に、CVTスムージング中、細長いメッシュ要素のボロノイ重心を計算すると、図4bに示すように、スター・ポリゴン領域から遠く離れた方向にシフトすることができます（縮退したメッシュ要素の場合、重心位置は無限距離になることさえあります）。さらに、最適化アルゴリズムでは、メッシュ要素のスケールが異なる場合に一様なステップサイズを使用すると、最適化プロセスが最適位置をオーバーシュートする可能性があるため、負の体積要素の生成につながる可能性もある。

負の体積要素は、ニューラルネットワークベースの平滑化アルゴリズムの学習と推論の段階でも発生する可能性がある。負の体積要素の生成は、セクション4.3で検証したように、学習プロセスを混乱させない。モデル学習により、負の体積要素の発生は徐々に減少する。しかし、ニューラルネットワークの不確実性により、稀ではあるが、推論段階で負の体積要素を導入することが可能である。したがって、負の体積要素の発生を防ぐための方法が必要である。最も単純なアプローチは、負の体積要素をもたらす変位をゼロに設定することである。

**Table 1:** Comparison among optimization-based smoothing, NN-Smoothing and GMSNet

Method	Speed	Labeled high-quality mesh	Varying node degrees	Node input order
OptimSmoothing <sup>1</sup>	Slow	Not acquiring	Not affected	Not affected
NN-Smoothing	Fast	Acquiring	Training separate models	Data augmentation
GMSNet	Fast	Not acquiring	Not affected	Not affected

<sup>1</sup> Optimization-based smoothing.

The main distinction between this approach and NN-Smoothing lies in the training data. In the proposed method, there are no high-quality meshes provided for the labels. Instead, the learning process relies solely on minimizing the mesh quality metric function. The primary divergence from optimization-based mesh smoothing methods is that this approach directly offers optimized positions for mesh nodes without the need to solve an optimization problem, resulting in a significant improvement in the efficiency of mesh smoothing. A comprehensive comparison of the three aforementioned methods is shown in Table 1.

### 3.2.3 MetricLoss

There are several metrics available to evaluate the quality of mesh elements, including the maximum angle, minimum angle, Jacobian matrix, aspect ratio, and others. In our case, we adopt the aspect ratio  $q = \frac{m^2+n^2+l^2}{4\sqrt{3}S}$  to assess the mesh quality, where  $m, n, l$  are the edges of the triangle, and  $S$  is the area of the triangle. For an equilateral triangle, this value is 1, while for a degenerate triangle, it approaches  $+\infty$ . However, the range of this metric is too large, which can cause gradient explosion, especially for poor-shape elements. To address this issue, we use  $1 - \frac{1}{q}$  as the evaluation metric for mesh elements. For equilateral triangles, this value is 0, while for degenerate triangles, it is 1. The formal definition of the loss function is as follows:

$$\begin{aligned}\mathcal{L}(\mathbf{x}, \mathbf{S}(\mathbf{x})) &= \frac{1}{|\mathbf{S}(\mathbf{x})|} \sum_{i=1}^{|\mathbf{S}(\mathbf{x})|} \left(1 - \frac{1}{q_i}\right) \\ &= \frac{1}{|\mathbf{S}(\mathbf{x})|} \sum_{i=1}^{|\mathbf{S}(\mathbf{x})|} \left(1 - \frac{4\sqrt{3}S_i}{m_i^2 + n_i^2 + l_i^2}\right)\end{aligned}\quad (10)$$

where  $|\mathbf{S}(\mathbf{x})|$  is the number of nodes in the *Star-Polygon*,  $m_i, n_i, l_i$  are the edges of triangle  $T_i$ , and

$q_i$  is the quality of  $T_i$ . We validated the effectiveness of our designed loss function in Section 4.4.

### 3.2.4 Shift truncation

In the process of mesh generation and mesh smoothing, it is important to ensure that the generated mesh avoids negative volume elements. However, mesh smoothing algorithms can sometimes lead to the generation of negative volume elements, as depicted in Figure 4. For instance, in the case of Laplacian smoothing, when dealing with non-convex *StarPolygons*, negative volume elements may arise (as illustrated in Figure 4a). Similarly, during CVT smoothing, the calculation of the Voronoi centroid for elongated mesh elements can result in shifts that extend far away from the *StarPolygon* region, as shown in Figure 4b (in the case of degenerate mesh elements, the centroid position may even be at an infinite distance). Additionally, in optimization algorithms, using a uniform step size for different scales of mesh elements can also lead to the production of negative volume elements, as the optimization process may overshoot the optimal position.

Negative volume elements can also occur during the training and inference stage of neural network-based smoothing algorithms. The generation of negative volume elements does not disrupt the learning process, as verified in Section 4.3. With the model training, the occurrence of negative volume elements gradually decreases. However, due to the uncertainty of neural networks, although rare, it is still possible for the model to introduce negative volume elements during the inference stage. Hence, a method is required to prevent the occurrence of negative volume elements. The simplest approach is to set the displacements that result in negative volume elements to zero. However, this approach hinders the update of poorly shaped mesh elements, which is

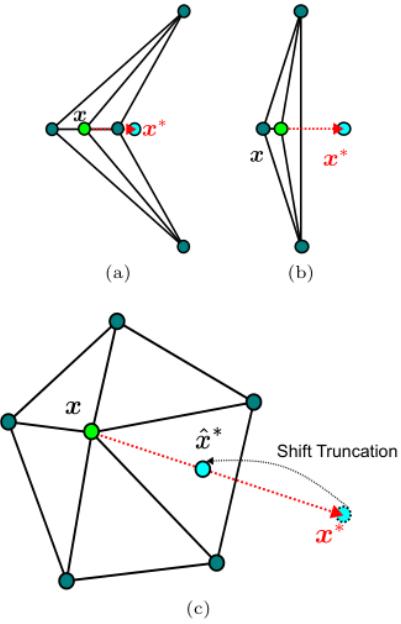


図4: a) 非凸のStarPolygonの場合、ラプラス平滑化によりStarPolygonの外側でノードを移動させることができる。b) CVTの平滑化には三角形の円周率の計算が必要である。高度に歪んだメッシュ要素の場合、その円周方向はStarPolygonから遠く離れており、負のボリューム要素になる。b) シフトの切り捨て。モデルの学習と推論の段階では、負の体積要素の生成を避けるためにシフトを切り捨てる。

しかし、このアプローチは、メッシュ平滑化の最適化目標である形状の悪いメッシュ要素の更新を妨げる。そこで、図4cに示すような負の体積要素を扱うために、線探索法を採用した。負の体積要素が生成されなくなるまで、負の体積要素を導入するシフトを半分に繰り返す。4.3節でシフト切り捨てがモデルに与える影響を調査する。

## 4 実験

### 4.1 実験セットアップ

本節では、提案するGMSNetと5つのベースライン平滑化手法(アルゴリズム)との包括的な性能比較を行った。評価したベースライン手法は以下の通りである: ラプラス平滑化[18]、

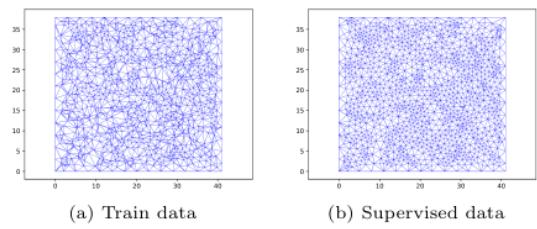
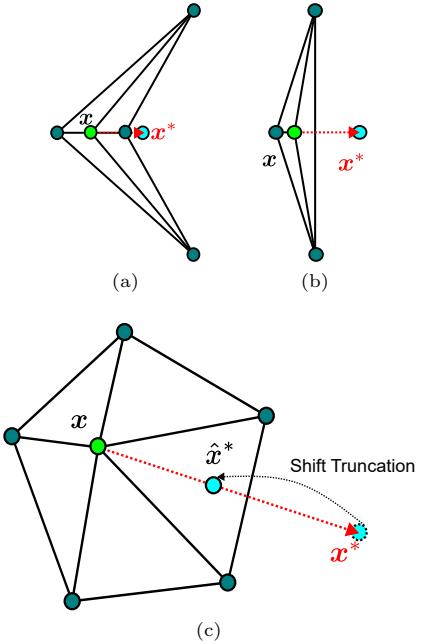


図5:NNSmoothingとGMSNetの学習用データセットのメッシュ例。GMSNetの学習時に教師ありデータは利用されない。

角度ベースの平滑化[13]、CVT平滑化[14]、最適化ベースの平滑化、NNS平滑化[11]。以下は、各ベースラインモデルの実装の詳細である:

- すべてのモデルは非同期更新を採用しており、メッシュノードは各最適化ステップの後に直接更新されます(すべてのノードの更新を計算し、それらと一緒に更新するのとは対照的です)。
- 負の体積要素を防ぐために、スマートラプラスメッセージングを採用した。
- 角度ベースの平滑化では、最終的なノード位置は、StarPolygonの各角度について計算された最適化された位置を平均化することによって得られた。
- CVTスマージングは、アルゴリズムの性能向上させるために式1を採用した。
- 最適化ベースの平滑化では、3.2.3節で定義したMetricLossを目的関数として用い、最適化器としてAdam [50]を採用した。各ノードは最大20回の反復で最適化された。
- NN-Smoothingは原著論文の実装アプローチに従った。次数の異なるノードに対して、メッシュを平滑化するために異なるモデルを学習させた。さらに、3、4、5、6、7、8、9以外の次数のノードを扱うために、ラプラス平滑化を使用した。

合計20個のメッシュからなる2次元の三角形メッシュを用いてモデルを学習させた。データセットは、6:2:2の割合でトレーニングセット、検証セット、テストセットに分割された。各メッシュは、トレーニング前にランダムに生成された、異なるサイズと密度を持つ。



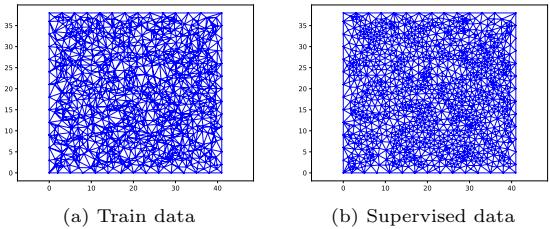
**Fig. 4:** a) For non-convex *StarPolygons*, Laplace smoothing may move nodes outside the *StarPolygon*. b) Computing the circumcenter of triangles is necessary for CVT smoothing. For highly distorted mesh element, it circumcenter is far away from the *StarPolygon*, resulting in negative volume elements. b) Shift truncation. In the training and inference phases of the model, we truncate the shift to avoid generating negative volume elements.

precisely the optimization goal of mesh smoothing. Therefore, we have adopted a line search method to handle negative volume elements, as depicted in Figure 4c. We repeatedly half the shift which introduces the negative volume elements until no negative volume element is generated. We investigate the effect of shift truncation on the model in Section 4.3.

## 4 Experiments

### 4.1 Experimental Setup

In this section, we conducted a comprehensive performance comparison of the proposed GMSNet with five baseline smoothing methods (algorithms). The baseline methods we evaluated are as follows: Laplacian smoothing [18],



**Fig. 5:** Mesh examples of the dataset for NN-Smoothing and GMSNet training. Supervised data is not utilized during the training of GMSNet.

Angle-based smoothing [13], CVT smoothing [14], Optimization-based smoothing, and NN-Smoothing [11]. Below are the implementation details for each baseline model:

- All models adopted the **asynchronous updating**, where the mesh nodes are directly updated after each optimization step (as opposed to computing the updates for all nodes and then updating them together).
- Smart Laplacian smoothing was adopted to prevent negative volume elements.
- In the Angle-based smoothing, the final node positions were obtained by averaging the optimized positions computed for each angle in the *StarPolygon*.
- The CVT smoothing employed the Equation 1 to improve the algorithm’s performance.
- The Optimization-based smoothing used MetricLoss defined in section 3.2.3 as the objective function and employed Adam [50] as the optimizer. Each node was optimized for a maximum of 20 iterations.
- The NN-Smoothing followed the implementation approach in the original paper. For nodes with different degrees, we trained different models to smooth the meshes. Additionally, Laplacian smoothing was used to handle nodes with degrees other than 3, 4, 5, 6, 7, 8, and 9.

We trained the model using two-dimensional triangle meshes, comprising a total of 20 meshes. The dataset was split into training, validation, and testing sets in a ratio of 6:2:2. Each mesh has distinct sizes and densities, randomly generated before the training. The mesh nodes are positioned randomly within the geometric domain, and the

メッシュノードは幾何学的領域内にランダムに配置され、メッシュはドロネー三角形分割[51]を用いて生成される。メッシュの例を図5aに示す。

実験では、最適化ベースのスムージングから得られた最終的な最適化結果を、図5bに示すNNSmoothingモデルの学習ラベルとして利用した。同時に、同じデータセットを用いて、学習過程でラベルを組み込まずにGMSNetを学習させた。両ニューラルネットワークモデルとも、オプティマイザとして Adam [50] を採用し、初期学習率は  $1e-2$  とした。学習過程を通して、学習率は検証セットでの性能に基づいて動的に調整された。各トレーニングエポックにおいて、各メッシュから32個のメッシュノードをランダムにサンプリングし、モデルをトレーニングした。メッシュノードの部分的なサンプリングのみで学習してもかかわらず、モデルは効果的に収束する。

#### 4.2 実験結果

モデルの平滑化性能を評価するために、4つのメッシュケースでテストを行った。モデルの性能をより詳細にテストするために、図7の最初の列に示すように、高度に歪んだ要素を含むメッシュをテストケースとして構築した。最初の2つのメッシュでは、メッシュノードは幾何学的領域内で一様サンプリングによって生成された。後者の2つのメッシュは、まずメッシュ作成ソフトウェアを用いて高品質なメッシュを生成し、次に歪んだメッシュ要素を導入するために手動で調整した。

異なるモデル間の公平な比較を容易にするため、アルゴリズムを同じフレームワークで実装した。アルゴリズム間のバリエーションは、最適化された点位置の生成方法にある。本研究では、直列アルゴリズムを採用したことを述べておく。しかし、ラプラスアン平滑化のような単純なアルゴリズムでは、すべてのノードを同時に更新することは簡単で高速である。メッシュ品質の評価指標として、最小角度、最大角度、アスペクト比の逆数を選んだ。各ケースについて10回の実験を行い、1回の実験につき最大100回の平滑化反復を行った。に基づいて、最も平滑化されたメッシュを最終結果として選択した。

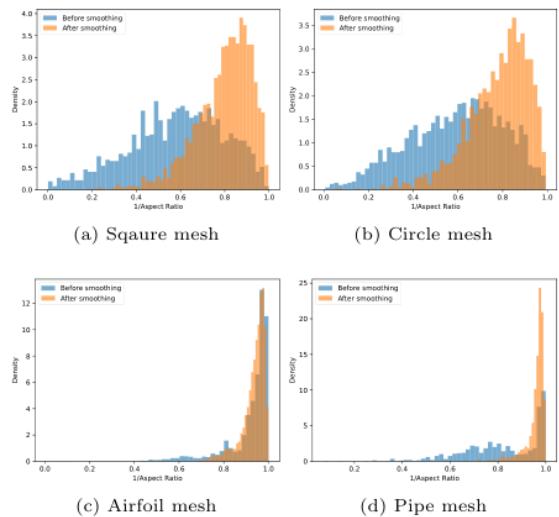


図6:GMSNetスムージング前後のメッシュ要素の品質分布。

重み付き品質メトリック、これは次のように定義される:

$$\hat{q} = \frac{1}{6} \left[ \frac{\alpha_{\text{mean}} + \alpha_{\text{min}} + 120 - \beta_{\text{max}} - \beta_{\text{mean}}}{60} + \left( \frac{1}{q} \right)_{\text{mean}} + \left( \frac{1}{q} \right)_{\text{min}} \right] \quad (11)$$

ここで、 $\alpha$ は最小角度、 $\beta$ は最大角度、 $q$ はアスペクト比である。さらに、アルゴリズムの速度は、1つのメッシュノードを処理するのにかかる時間で測定される。GMSNetを用いたメッシュスムージングの結果を図7の2列目に示し、各アルゴリズムの性能の総合的な比較を表2にまとめた。

図7から、4つのテストケースすべてにおいて、我々の提案モデルがメッシュ要素の品質を大幅に改善していることがわかる。ノードの次数が適度に分布するメッシュの場合、図7fと7hに示すように、我々のアプローチは非常に滑らかなメッシュを生成する。さらに、我々のアルゴリズムは、図7bと7dに示すように、高度に歪んだメッシュに対する頑健性を保証する。表2に示すように、我々の提案するアルゴリズムは、ほとんどのヒューリスティックメッシュ平滑化手法を凌駕し、メッシュ要素の品質メトリクスは、最適化ベースのアルゴリズムを使用して得られた結果に近い。すべてのテストケースにおいて、我々のモデルは、以下の通り、N-Smoothingモデルを概ね上回った。

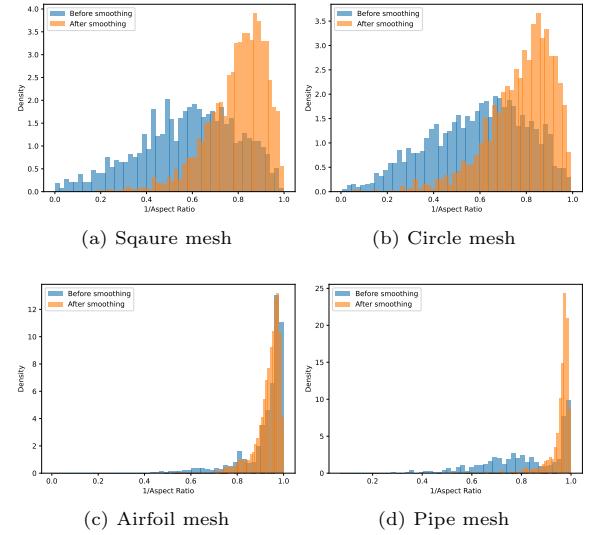
meshes are generated using Delaunay triangulation [51]. Example of the meshes is shown in Figure 5a.

In the experiment, we utilized the final optimization results obtained from the Optimization-based smoothing as the training labels for the NN-Smoothing model, which is shown in Figure 5b. Simultaneously, we trained the GMSNet without incorporating labels during the training process using the same dataset. For both neural network models, we employed Adam [50] as the optimizer with an initial learning rate of 1e-2. Throughout the training process, the learning rate was dynamically adjusted based on the performance on the validation set. In each training epoch, we randomly sampled 32 mesh nodes from each mesh to train the models. Despite training with only a partial sampling of mesh nodes, the models converge effectively.

## 4.2 Experimental results

We conducted tests on four mesh cases to evaluate the model's smoothing performance. To test the performance of the model more thoroughly, we constructed meshes containing highly distorted elements as test cases, as shown in the first column of Figure 7. For the first two meshes, mesh nodes were generated by uniform sampling within the geometric domain. The latter two meshes were initially created using meshing software to generate high-quality meshes, and then manual adjustments were made to introduce distorted mesh elements.

To facilitate a fair comparison among different models, we have implemented the algorithms within the same framework. The variations among the algorithms lie in how they generate the optimized point positions. It is essential to mention that we have employed serial algorithms in our study. However, for simpler algorithms like Laplacian smoothing, updating all nodes simultaneously is straightforward and fast. We have chosen the minimum angle, maximum angle, and the reciprocal of the aspect ratio as evaluation metrics for mesh quality. We conducted ten experimental runs for each case, with a maximum of 100 smoothing iterations per experiment. The best-smoothed mesh was selected as the final result based on the



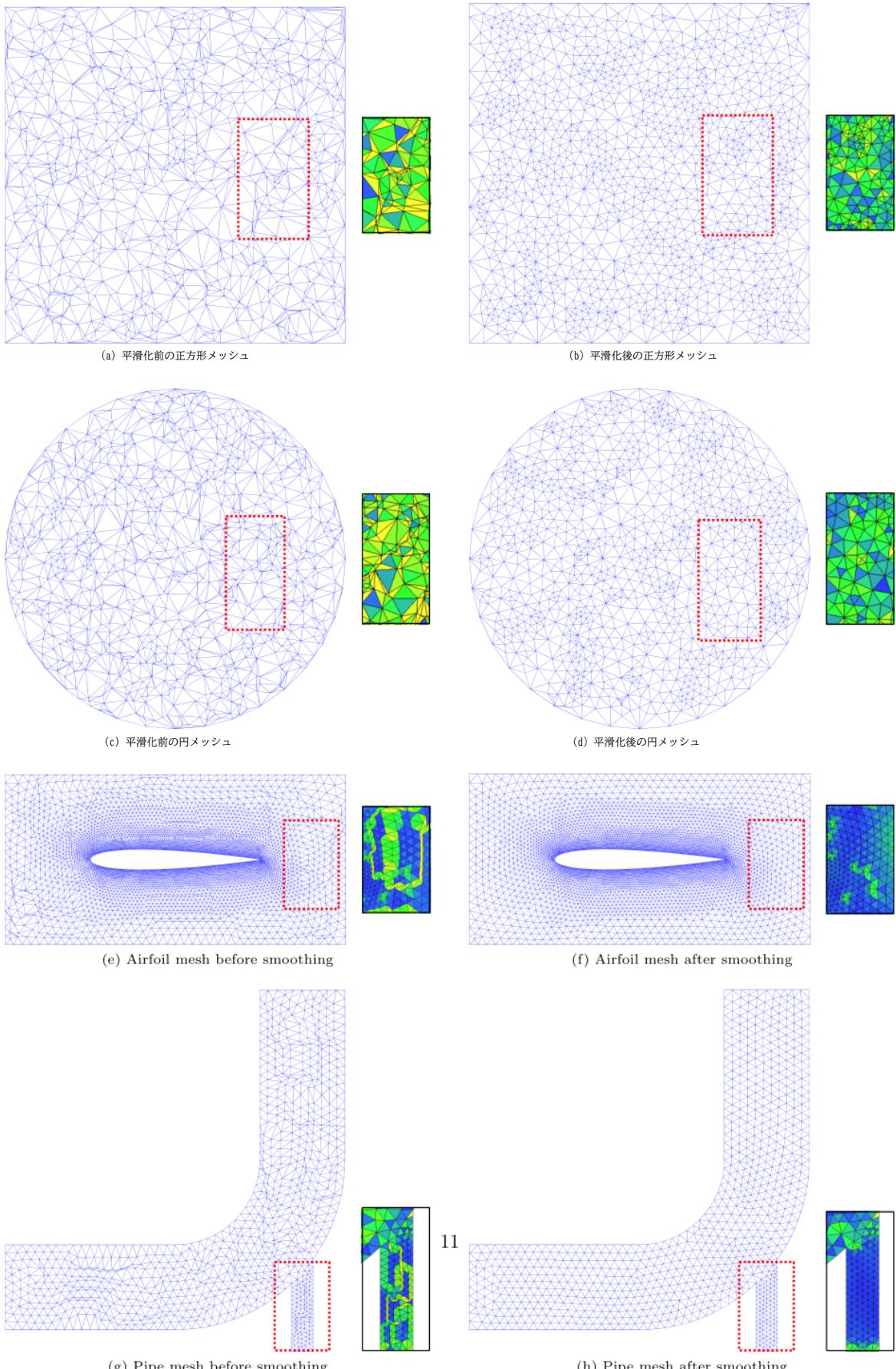
**Fig. 6:** The quality distribution of mesh elements before and after GMSNet smoothing.

weighted quality metric, which is defined as:

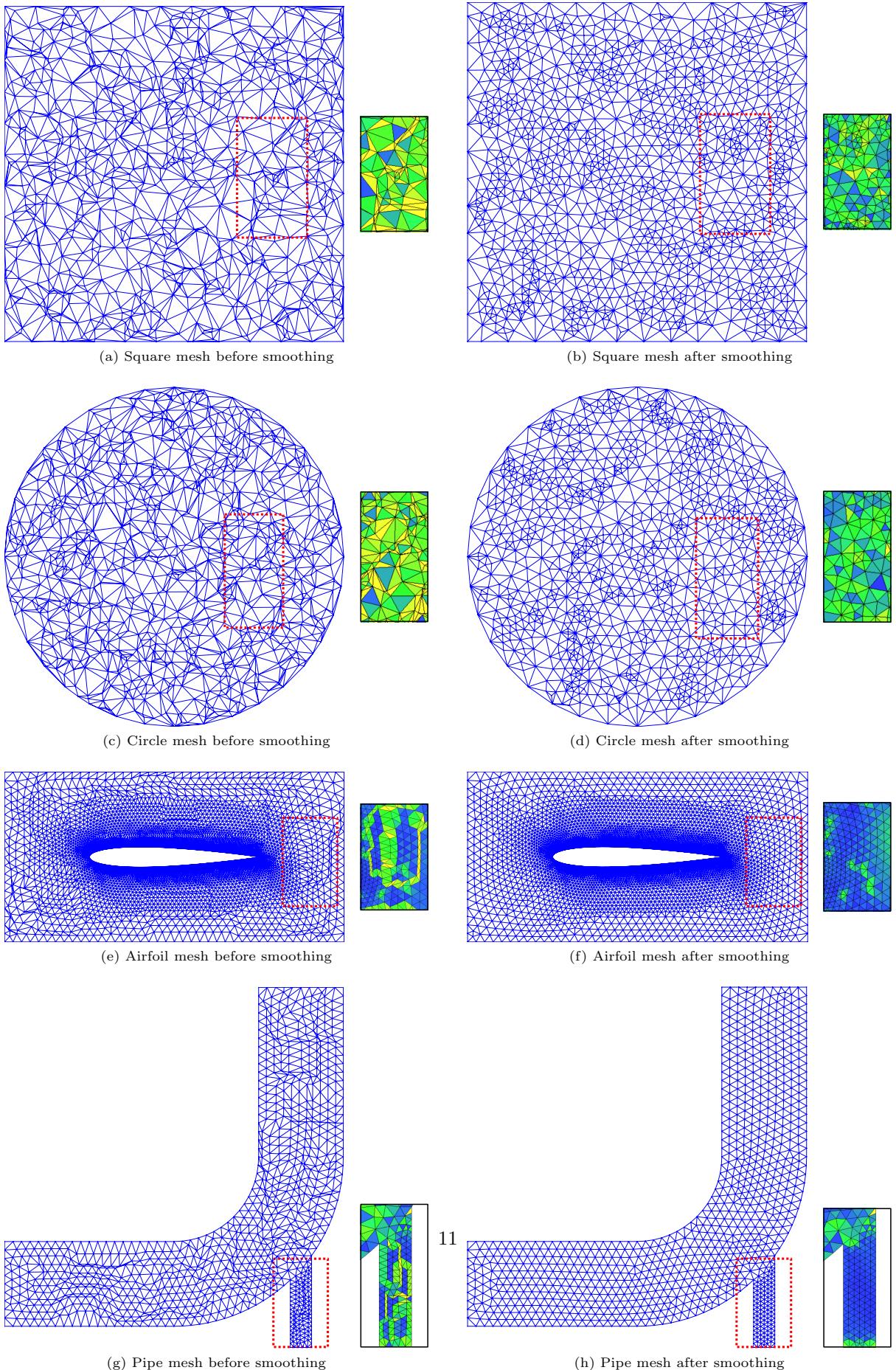
$$\hat{q} = \frac{1}{6} \left[ \frac{\alpha_{\text{mean}} + \alpha_{\text{min}} + 120 - \beta_{\text{max}} - \beta_{\text{mean}}}{60} + \left( \frac{1}{q} \right)_{\text{mean}} + \left( \frac{1}{q} \right)_{\text{min}} \right] \quad (11)$$

where  $\alpha$  is minimum angle,  $\beta$  is the maximum angle, and  $q$  is the aspect ratio. Furthermore, the algorithm speed is measured in terms of the time taken to process a single mesh node. The results of mesh smoothing using the GMSNet are depicted in the second column of Figure 7, and a comprehensive comparison of the performances of each algorithm is summarized in Table 2.

From Figure 7, it can be observed that for all four test cases, our proposed model significantly improves the mesh element quality. For meshes with reasonably distributed node degrees, our approach produces very smooth meshes, as shown in Figure 7f and 7h. Moreover, our algorithm ensures robustness for highly distorted meshes, as demonstrated in Figure 7b and 7d. As shown in Table 2, our proposed algorithm outperforms most heuristic mesh smoothing methods, and the mesh element quality metrics closely approximate the results obtained using optimization-based algorithms. For all test cases, our model generally outperforms the NN-Smoothing model, despite



**Fig. 7:** Mesh smoothing results of GMSNet on the test cases. Mesh nodes in meshes a) and c) are randomly generated in the domain, and mesh nodes in meshes e) and g) are manually adjusted to introduce distorted elements. High-quality elements are colored blue, and low-quality elements are colored yellow.



**Fig. 7:** Mesh smoothing results of GMSNet on the test cases. Mesh nodes in meshes a) and c) are randomly generated in the domain, and mesh nodes in meshes e) and g) are manually adjusted to introduce distorted elements. High-quality elements are colored blue, and low-quality elements are colored yellow.

表2: メッシュスムージングアルゴリズムの性能

Mesh	Algorithm	Min. Angle		Max. Angle		$\frac{1}{\text{Aspect ratio}}$		s/per node
		min	mean	max	mean	min	mean	
Square	Origin	0.04	29.65	179.82	95.09	0.00	0.58	-
	LaplacianSmoothing	12.38	43.97	147.67	79.44	0.25	0.79	1.26E-03
	AngleSmoothing	10.52	41.26	148.6	81.76	0.22	0.76	2.36E-03
	CVTSmoothing	2.55	40.88	170.6	84.04	0.05	0.75	4.60E-03
	OptimSmoothing	16.33	43.94	136.99	80.7	0.32	0.79	2.29E-02
	NN-Smoothing	9.14	43.58	158.97	79.79	0.16	0.79	1.97E-03
	GMSNet	13.99	43.81	151.39	79.81	0.22	0.79	2.58E-03
Circle	Origin	0.29	30.38	179.05	94.60	0.01	0.59	-
	LaplacianSmoothing	1.43	42.86	174.53	81.19	0.03	0.78	1.21E-03
	AngleSmoothing	1.95	39.47	173.97	84.23	0.04	0.73	2.30E-03
	CVTSmoothing	3.19	39.98	172.98	85.59	0.05	0.73	4.35E-03
	OptimSmoothing	6.52	43.31	154.66	81.9	0.15	0.78	2.32E-02
	NN-Smoothing	1.26	41.1	176.04	83.86	0.03	0.75	1.88E-03
	GMSNet	2.2	43	169.21	81.29	0.05	0.78	2.38E-03
Airfoil	Origin	0.25	53.25	178.91	67.84	0.01	0.92	-
	LaplacianSmoothing	26.36	54.91	111.02	65.78	0.51	0.94	1.30E-03
	AngleSmoothing	20.26	53.49	118.76	66.64	0.42	0.93	2.54E-03
	CVTSmoothing	25.49	52.08	115.79	68.19	0.48	0.91	4.64E-03
	OptimSmoothing	30.53	54.9	108.85	65.72	0.54	0.94	1.66E-02
	NN-Smoothing	27.69	54.06	113.06	66.57	0.51	0.93	2.03E-03
	GMSNet	27.17	54.5	110.47	66.08	0.52	0.94	2.52E-03
Pipe	Origin	4.10	46.28	170.48	76.81	0.07	0.82	-
	LaplacianSmoothing	27.91	56.55	112.45	63.93	0.52	0.96	1.05E-03
	AngleSmoothing	24.28	54.62	101.93	65.69	0.53	0.94	1.93E-03
	CVTSmoothing	27.78	54.22	97.8	66.28	0.62	0.93	3.68E-03
	OptimSmoothing	32.39	56.41	106.07	64.14	0.57	0.96	1.79E-02
	NN-Smoothing	28.28	53.72	112.79	66.69	0.51	0.93	1.52E-03
	GMSNet	28.23	55.76	112.27	64.71	0.52	0.95	1.93E-03

NN-Smoothingが7つのモデルを訓練する必要があるのに対して、我々は異なる度数のノードを平滑化するために1つのモデルだけを訓練したという事実。我々のモデルは、NN-Smoothingモデルと比較して、パラメータが5%しかない。効率性の観点から、提案手法は最適化ベースのアプローチと比較して、平均8.62倍の速度向上を示している。

また、メッシュ要素の品質分布を調べることで、提案アルゴリズムの有効性を検証する。スムージング前後のメッシュ要素の品質分布を図6に示す。このアルゴリズムにより、高品質なメッシュ要素の割合が大幅に増加し、

全体的なメッシュ品質が向上していることがわかる。以上より、メッシュ平滑化タスクに対する提案モデルの有効性と効率性が実験結果から実証された。

#### 4.3 シフト切り捨ての本質

モデルにおけるシフト切り捨ての有効性を比較するために、学習の最初のエポック後の出力メッシュを可視化した。図8aに示すように、ノードの更新の大部分が負のボリューム要素をもたらしたことが観察される。しかし、モデルが学習を続けると、平滑化処理を学習し、負の体積要素の数が減少することが描かれている。

**Table 2:** Performance of mesh smoothing algorithms

Mesh	Algorithm	Min. Angle		Max. Angle		$\frac{1}{\text{Aspect ratio}}$		s/per node
		min	mean	max	mean	min	mean	
Square	Origin	0.04	29.65	179.82	95.09	0.00	0.58	-
	LaplacianSmoothing	12.38	43.97	147.67	79.44	0.25	0.79	1.26E-03
	AngleSmoothing	10.52	41.26	148.6	81.76	0.22	0.76	2.36E-03
	CVTSmoothing	2.55	40.88	170.6	84.04	0.05	0.75	4.60E-03
	OptimSmoothing	16.33	43.94	136.99	80.7	0.32	0.79	2.29E-02
	NN-Smoothing	9.14	43.58	158.97	79.79	0.16	0.79	1.97E-03
Circle	GMSNet	13.99	43.81	151.39	79.81	0.22	0.79	2.58E-03
	Origin	0.29	30.38	179.05	94.60	0.01	0.59	-
	LaplacianSmoothing	1.43	42.86	174.53	81.19	0.03	0.78	1.21E-03
	AngleSmoothing	1.95	39.47	173.97	84.23	0.04	0.73	2.30E-03
	CVTSmoothing	3.19	39.98	172.98	85.59	0.05	0.73	4.35E-03
	OptimSmoothing	6.52	43.31	154.66	81.9	0.15	0.78	2.32E-02
Airfoil	NN-Smoothing	1.26	41.1	176.04	83.86	0.03	0.75	1.88E-03
	GMSNet	2.2	43	169.21	81.29	0.05	0.78	2.38E-03
	Origin	0.25	53.25	178.91	67.84	0.01	0.92	-
	LaplacianSmoothing	26.36	54.91	111.02	65.78	0.51	0.94	1.30E-03
	AngleSmoothing	20.26	53.49	118.76	66.64	0.42	0.93	2.54E-03
	CVTSmoothing	25.49	52.08	115.79	68.19	0.48	0.91	4.64E-03
Pipe	OptimSmoothing	30.53	54.9	108.85	65.72	0.54	0.94	1.66E-02
	NN-Smoothing	27.69	54.06	113.06	66.57	0.51	0.93	2.03E-03
	GMSNet	27.17	54.5	110.47	66.08	0.52	0.94	2.52E-03
	Origin	4.10	46.28	170.48	76.81	0.07	0.82	-
	LaplacianSmoothing	27.91	56.55	112.45	63.93	0.52	0.96	1.05E-03
	AngleSmoothing	24.28	54.62	101.93	65.69	0.53	0.94	1.93E-03
Pipe	CVTSmoothing	27.78	54.22	97.8	66.28	0.62	0.93	3.68E-03
	OptimSmoothing	32.39	56.41	106.07	64.14	0.57	0.96	1.79E-02
	NN-Smoothing	28.28	53.72	112.79	66.69	0.51	0.93	1.52E-03
	GMSNet	28.23	55.76	112.27	64.71	0.52	0.95	1.93E-03

the fact that we trained only one model to smooth nodes of different degrees, whereas NN-Smoothing requires training seven models. Our model has only 5% of the parameters compared to the NN-Smoothing model. From the efficiency perspective, the proposed method exhibits a average speed improvement of of 8.62 times compared to optimization-based approaches.

We also validate the effectiveness of our proposed algorithm by examining the distribution of mesh element quality. The quality distributions of mesh elements before and after smoothing are shown in Figure 6. It can be observed that the algorithm significantly increases the proportion of high-quality mesh elements and improves the

overall mesh quality. In summary, the experimental results demonstrate the effectiveness and efficiency of our proposed model for mesh smoothing tasks.

### 4.3 The essential of shift truncation

To compare the effectiveness of shift truncation in the model, we visualized the output mesh after the first epoch of training. It can be observed that a large portion of the node updates resulted in negative volume elements, as shown in Figure 8a. However, as the model continues training, it learns the smoothing process, and the number of negative volume elements decreases, as depicted

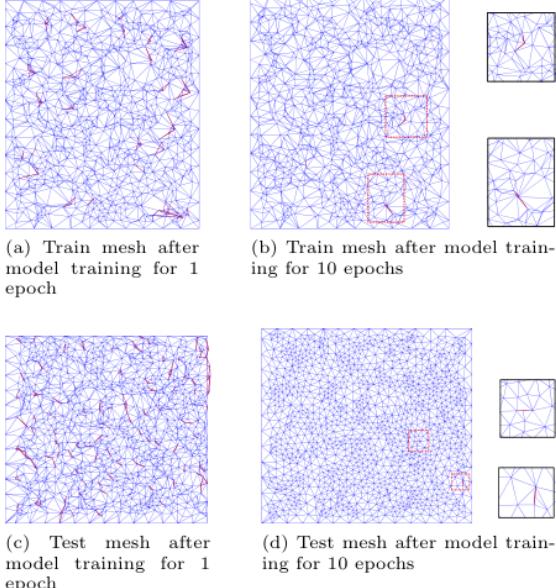


図8:シフト切り捨ての本質。図中の赤い要素は負の体積要素を表している。

を図 8b に示す。また、モデル予測におけるシフト切り捨ての影響も検討した。実験結果を図8cと図8dに示す。ある高度に歪んだ要素については、エポック学習したにもかかわらず、モデルが負の体積要素を生成する可能性があることがわかる。これは、予測段階でシフトの切り捨てが不可欠であることを示している。

#### 4.4 異なる損失関数の影響

3.2.3節では、モデル学習のための損失関数を設計した。本節では、MetricLossの有効性と適切な損失関数の選択方法について述べる。モデル学習時の損失関数として、一般的に使用されるメッシュ品質メトリクスの性能を比較する：

- 最小-最大角度損失

$$\mathcal{L}(\mathbf{x}, \mathbf{S}(\mathbf{x})) = \frac{1}{|\mathbf{S}(\mathbf{x})|} \sum_{i=1}^{|\mathbf{S}(\mathbf{x})|} \min \max(\theta_{ij}), \text{ for } j = 1, 2, 3, \text{ where } \theta_{ij} \text{ is the angle in triangle } T_i.$$

- Aspect ratio loss:

$$\mathcal{L}(\mathbf{x}, \mathbf{S}(\mathbf{x})) = \frac{1}{|\mathbf{S}(\mathbf{x})|} \sum_{i=1}^{|\mathbf{S}(\mathbf{x})|} \frac{m_i^2 + n_i^2 + l_i^2}{4\sqrt{3}S_i}, \text{ where the symbols are defined in Section 3.2.3.}$$

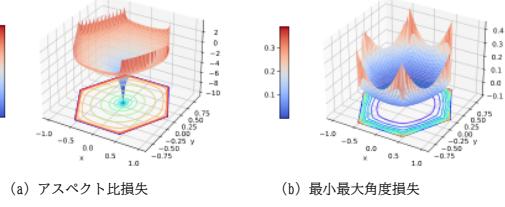


図9:異なる損失関数のプロット。損失関数の範囲と滑らかさは、モデルの収束に影響を与える。図9aでは、z座標は $\log(z)$ で与えられる。

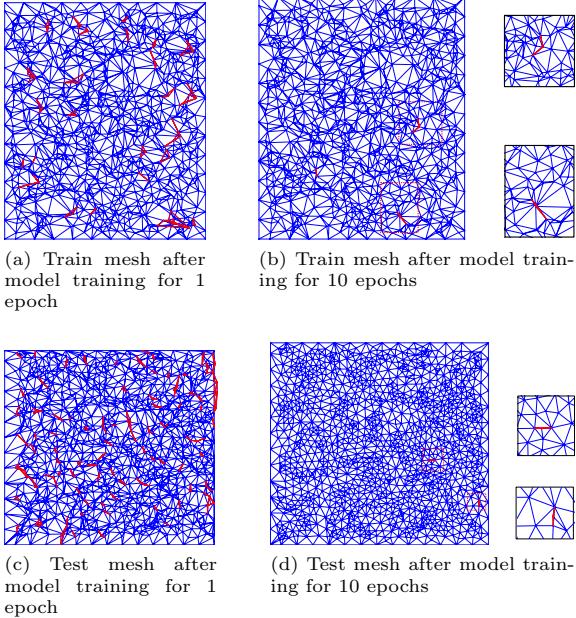
- Cosine loss:

$$\mathcal{L}(\mathbf{x}, \mathbf{S}(\mathbf{x})) = \frac{1}{3|\mathbf{S}(\mathbf{x})|} \sum_{i=1}^{|\mathbf{S}(\mathbf{x})|} (\cos(\theta_{ij}) - \frac{1}{2})^2$$

- MetricLoss:

$$\mathcal{L}(\mathbf{x}, \mathbf{S}(\mathbf{x})) = \frac{1}{|\mathbf{S}(\mathbf{x})|} \sum_{i=1}^{|\mathbf{S}(\mathbf{x})|} \left(1 - \frac{4\sqrt{3}S_i}{m_i^2 + n_i^2 + l_i^2}\right)$$

StarPolygonの中心ノードのバリエーションに関する前述の損失関数のプロットを図9に示す。我々が採用した損失関数は、入力が変化するにつれてより滑らかな変換を示し、最適化プロセスを容易にすることが観察できる。また、アスペクト比損失と最小-最大角度損失は、損失関数として採用するのに適していないことがわかる。非常に歪んだ要素に遭遇した場合、前者は数値が正の無限大に近づき、損失が急激に増加し、学習が不安定になる。一方、後者は、高度に歪んだ要素に直面すると損失が急激に変化するが、中央のノードが最適位置の近傍に位置する場合、損失関数は比較的平坦なままである。このような状況は、最適化プロセスの妨げにもなる。元の角度ベースの損失関数は学習が難しいが、図9cに示すように、余弦関数を用いて変換した損失関数はより安定している。同じモデル構成と学習構成を採用し、異なる損失関数を用いてモデルを学習させた。



**Fig. 8:** The essential of shift truncation. The red elements in the figure represent negative volume elements.

in Figure 8b. We also study the impact of shift truncation in the model predictions. The experimental results are shown in Figure 8c and 8c. It is evident that for certain highly distorted elements, the model may produce negative volume elements despite being trained for epochs. This demonstrates that shift truncation is essential during the prediction stage.

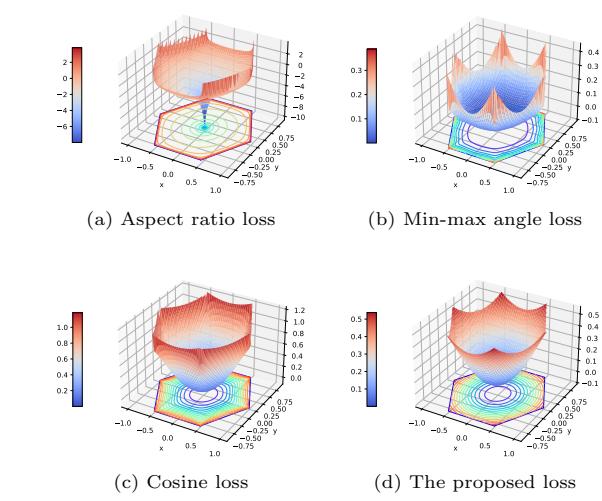
#### 4.4 The influence of different loss functions

In Section 3.2.3, we designed a loss function for model training. In this section, we discuss the effectiveness of MetricLoss and how to choose the appropriate loss function. We compare the performance of commonly used mesh quality metrics as loss functions during model training, including:

- Min-max angle loss:  

$$\mathcal{L}(\mathbf{x}, \mathbf{S}(\mathbf{x})) = \frac{1}{|\mathbf{S}(\mathbf{x})|} \sum_{i=1}^{|\mathbf{S}(\mathbf{x})|} \min \max(\theta_{ij}),$$
 for  $j = 1, 2, 3$ , where  $\theta_{ij}$  is the angle in triangle  $T_i$ .
- Aspect ratio loss:  

$$\mathcal{L}(\mathbf{x}, \mathbf{S}(\mathbf{x})) = \frac{1}{|\mathbf{S}(\mathbf{x})|} \sum_{i=1}^{|\mathbf{S}(\mathbf{x})|} \frac{m_i^2 + n_i^2 + l_i^2}{4\sqrt{3}S_i},$$
 where the symbols are defined in Section 3.2.3.



**Fig. 9:** Plots of different loss functions. The range and smoothness of the loss function influence the model’s convergence. In Figure 9a, The z-coordinate is given as  $\log(z)$ .

- Cosine loss:  

$$\mathcal{L}(\mathbf{x}, \mathbf{S}(\mathbf{x})) = \frac{1}{3|\mathbf{S}(\mathbf{x})|} \sum_{i=1}^{|\mathbf{S}(\mathbf{x})|} (\cos(\theta_{ij}) - \frac{1}{2})^2$$
- MetricLoss:  

$$\mathcal{L}(\mathbf{x}, \mathbf{S}(\mathbf{x})) = \frac{1}{|\mathbf{S}(\mathbf{x})|} \sum_{i=1}^{|\mathbf{S}(\mathbf{x})|} (1 - \frac{4\sqrt{3}S_i}{m_i^2 + n_i^2 + l_i^2})$$

The plots of the aforementioned loss functions with respect to the variations in the central node of the *StarPolygon* are shown in Figure 9. It can be observed that the loss function we employed exhibits smoother transformations as the input changes, which facilitates the optimization process. We can also see than the aspect ratio loss and min-max angle loss are not suitable to be employed as the loss function. When encountering highly distorted elements, the former leads to numerical values approaching positive infinity, causing a sharp increase in loss and inducing training instability. The latter, on the other hand, results in abrupt changes in loss when facing highly distorted elements, while the loss function remains relatively flat when the central node is situated in the neighborhood of the optimal position. This circumstance also hinders the optimization process. Although the original angle-based loss function is difficult to train, the loss function transformed using the cosine function is more stable, as shown in Figure 9c. We trained the model using different loss functions, employing the same model configuration and training configuration.

表3:パイプメッシュとスクエアメッシュにおけるコサインロスと提案ロスの比較

Mesh	Algorithm	Min. Angle		Max. Angle		$\frac{1}{\text{Aspect ratio}}$	
		min	mean	max	mean	min	mean
Pipe	Cosine loss	26.77	55.71	110.43	64.67	0.54	0.95
	MetricLoss	26.37	55.56	107.73	64.82	0.56	0.95
Square	Cosine loss	11.04	43.76	146.87	79.61	0.25	0.79
	MetricLoss	13.53	44.06	150.46	79.65	0.22	0.79

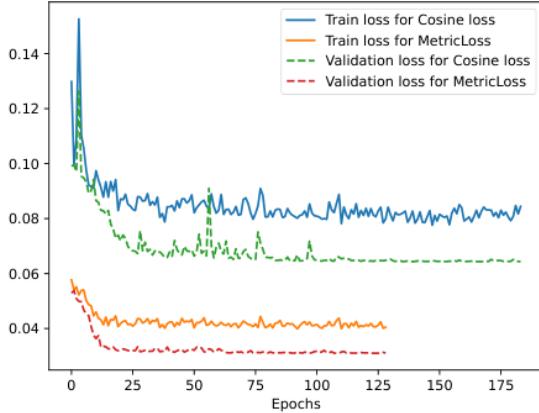


図10: 損失関数の違いによる比較。最小-最大角度とアスペクト比に基づく損失関数は、初期エポック中に学習失敗に遭遇した。一方、MetricLossと変換されたCosine損失関数は収束した結果を得た。

損失関数を変えたモデルの訓練損失と検証損失を図10に示す。MetricLossとcosine損失関数は、数回の繰り返しでモデルが急速に収束するため、モデルの学習に適していることがわかる。さらに、MetricLossは学習過程においても驚くほど安定している。異なる損失関数に基づく2つのモデル間の実験を表3に示す。両モデルは同様の性能を示していることがわかる。

他の2つの損失関数で学習する場合、モデルは収束しないか、激しい振動を示す。このことは、高度に歪んだ要素に遭遇したときに、より大きな数値範囲や数値の急激な変化を持つメッシュ品質メトリクスを採用することは、モデルの学習に不利であることを示している。要約すると、メッシュスマージングモデルの学習には様々な損失関数を用いることができる。

メッシュ品質メトリクスによって損失関数を構築する場合、歪んだ要素に遭遇したときの関数の挙動を考慮し、数値の範囲が大きくなったり、急激な変化があったりする損失関数が選択されないようにする必要がある。不適切なメッシュ品質メトリクスは、最小-最大角度損失と余弦損失によって示されるように、変換してモデル学習に採用することもできる。

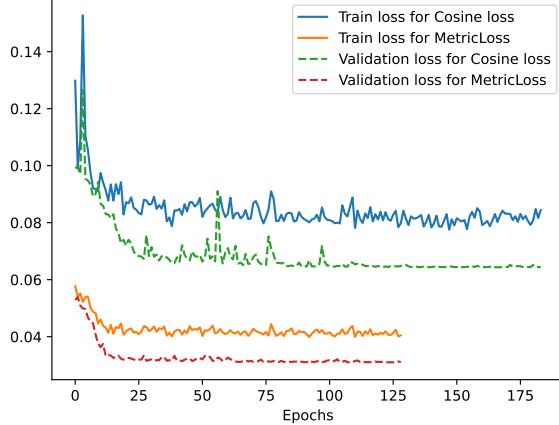
## 5 Conclusion

本論文では、インテリジェントメッシュスマージングのためのグラフニューラルネットワークモデルGMSNetを提案する。提案モデルは、メッシュノードの近傍を入力とし、メッシュノードの平滑化された位置を直接出力するように学習することで、最適化ベースの平滑化に伴う計算オーバーヘッドを回避する。また、負の値要素を防ぐために、フォールトトレランス機構であるシフトトランケーションが適用されている。軽量設計により、GMSNetは程度の異なるメッシュノードに適用でき、データ入力順序の影響を受けない。また、メッシュ品質メトリクスに基づく新しい損失関数MetricLossを導入することで、モデルを学習するための高品質なメッシュ生成コストを不要にする。2次元三角メッシュを用いた実験により、我々の提案モデルは、最適化ベースの平滑化と比較して、平均8.62倍の高速化で優れた平滑化性能を達成することが実証された。また、GMSNetはNNSmoothingモデルよりも、わずか5%のモデルパラメータで優れた性能を達成することが示された。また、MetricLossが高速で安定したモデル学習を実現することを説明し、実験を通してシフト切り捨て操作の本質を示す。

今後の課題としては、提案手法を他の種類のメッシュ要素に適用し、

**Table 3:** Comparison of cosine loss and proposed loss on Pipe and Square meshes

Mesh	Algorithm	Min. Angle		Max. Angle		$\frac{1}{\text{Aspect ratio}}$	
		min	mean	max	mean	min	mean
Pipe	Cosine loss	26.77	55.71	110.43	64.67	0.54	0.95
	MetricLoss	26.37	55.56	107.73	64.82	0.56	0.95
Square	Cosine loss	11.04	43.76	146.87	79.61	0.25	0.79
	MetricLoss	13.53	44.06	150.46	79.65	0.22	0.79



**Fig. 10:** Comparison of different loss functions. The loss functions based on min-max angle and aspect ratio encountered training failures during the initial epochs. In contrast, both MetricLoss and the transformed Cosine loss function yielded convergent results.

The train loss and validation loss of models with different loss functions are depicted in Figure 10. It can be observed that MetricLoss and cosine loss function are suitable for training the model, as the model converges rapidly after several iterations. Moreover, MetricLoss is remarkably stable during the training process. The experiments between two models based on different loss functions are shown in Table 3. It can be seen that the models show similar performance.

When training with the other two loss functions, the model either fails to converge or exhibits severe oscillations. This indicates that employing mesh quality metrics with a larger range of numerical values or rapid changes in the numerical values when encountering highly distorted elements is detrimental to the training of the model. In summary, various loss functions can be used for mesh

smoothing model training. When constructing the loss function through mesh quality metrics, it is necessary to consider the behavior of the function when encountering distorted elements, and to avoid selecting loss functions with a larger range of numerical values or abrupt changes. Inappropriate mesh quality metrics can also be transformed and employed for model training, as demonstrated by the min-max angle loss and cosine loss.

## 5 Conclusion

In this paper, we propose a graph neural network model, GMSNet, for intelligent mesh smoothing. The proposed model takes the neighbours of mesh nodes as input and learns to directly output the smoothed positions of mesh nodes, avoiding the computational overhead associated with the optimization-based smoothing. A fault-tolerance mechanism, shift truncation, is also applied to prevent negative volume elements. With a lightweight design, GMSNet can be applied to mesh nodes with varying degrees, and is not affected by the data input order. We also introduce a novel loss function, MetricLoss, based on mesh quality metrics, eliminating the need for high-quality mesh generation cost to train the model. Experiments on two-dimensional triangle meshes demonstrate that our proposed model achieves outstanding smoothing performance with an average acceleration of 8.62 times compared to optimization-based smoothing. The results also show that GMSNet achieve superior performance than NN-Smoothing model with just 5% model parameters. We also illustrate that MetricLoss achieves fast and stable model training, and show the essential of shift truncation operation through experiments.

Future work includes applying our proposed method to other types of mesh elements and

表面メッシュや体積メッシュに拡張することである。また、優れたメッシュ平滑化効果を得るために、メッシュ平滑化プロセスにエッジの反転やメッシュ密度の修正を導入することも、検討する価値のあるアプローチである。

補足情報である。

**Acknowledgments.** This research work was supported in part by the National Key Research and Development Program of China (2021YFB0300101).

## Declarations

## References

- [1] Spalart, P.R., Venkatakrishnan, V.: On the role and challenges of cfd in the aerospace industry. *The Aeronautical Journal* **120**(1223), 209–232 (2016)
- [2] Bridgeman, J., Jefferson, B., Parsons, S.A.: The development and application of cfd models for water treatment flocculators. *Advances in Engineering Software* **41**(1), 99–109 (2010)
- [3] Damjanović, D., Kozak, D., Živić, M., Ivandić, Ž., Baškarić, T.: Cfd analysis of concept car in order to improve aerodynamics. *Járműipari innováció* **1**(2), 108–115 (2011)
- [4] Samstag, R.W., Ducoste, J.J., Griborio, A., Nopens, I., Batstone, D., Wicks, J., Saunders, S., Wicklein, E., Kenny, G., Laurent, J.: Cfd for wastewater treatment: an overview. *Water Science and Technology* **74**(3), 549–563 (2016)
- [5] Darwish, M., Moukalled, F.: The Finite Volume Method in Computational Fluid Dynamics: an Advanced Introduction with OpenFOAM® and Matlab®, pp. 110–128. Springer, Berlin (2016)
- [6] Baker, T.J.: Mesh generation: Art or science? *Progress in Aerospace Sciences* **41**(1), 29–63 (2005)
- [7] Knupp, P.M.: Algebraic mesh quality metrics. *SIAM journal on scientific computing* **23**(1), 193–218 (2001)
- [8] Freitag, L.A., Ollivier-Gooch, C.: Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering* **40**(21), 3979–4002 (1997)
- [9] Freitag, L.A., Ollivier-Gooch, C.: A comparison of tetrahedral mesh improvement techniques. Technical report, Argonne National Lab.(ANL), Argonne, IL (United States) (1996)
- [10] Prasad, T.: A comparative study of mesh smoothing methods with flipping in 2d and 3d. PhD thesis, Rutgers University-Camden Graduate School (2018)
- [11] Guo, Y., Wang, C., Ma, Z., Huang, X., Sun, K., Zhao, R.: A new mesh smoothing method based on a neural network. *Computational Mechanics*, 1–14 (2021)
- [12] Herrmann, L.R.: Laplacian-isoparametric grid generation scheme. *Journal of the Engineering Mechanics Division* **102**(5), 749–756 (1976)
- [13] Zhou, T., Shimada, K.: An angle-based approach to two-dimensional mesh smoothing. *IMR* **2000**, 373–384 (2000)
- [14] Du, Q., Gunzburger, M.: Grid generation and optimization based on centroidal voronoi tessellations. *Applied mathematics and computation* **133**(2-3), 591–607 (2002)
- [15] Lloyd, S.: Least squares quantization in pcm. *IEEE transactions on information theory* **28**(2), 129–137 (1982)
- [16] Du, Q., Faber, V., Gunzburger, M.: Centroidal voronoi tessellations: Applications and algorithms. *SIAM review* **41**(4), 637–676 (1999)
- [17] Parthasarathy, V., Kodiyalam, S.: A constrained optimization approach to finite element mesh smoothing. *Finite Elements in*

extending it to surface and volume meshes. Introducing edge flipping and mesh density modification into the mesh smoothing process to achieve superior mesh smoothing effects is also an approach worth exploring.

#### Supplementary information.

**Acknowledgments.** This research work was supported in part by the National Key Research and Development Program of China (2021YFB0300101).

## Declarations

## References

- [1] Spalart, P.R., Venkatakrishnan, V.: On the role and challenges of cfd in the aerospace industry. *The Aeronautical Journal* **120**(1223), 209–232 (2016)
- [2] Bridgeman, J., Jefferson, B., Parsons, S.A.: The development and application of cfd models for water treatment flocculators. *Advances in Engineering Software* **41**(1), 99–109 (2010)
- [3] Damjanović, D., Kozak, D., Živić, M., Ivandić, Ž., Baškarić, T.: Cfd analysis of concept car in order to improve aerodynamics. *Járműipari innováció* **1**(2), 108–115 (2011)
- [4] Samstag, R.W., Ducoste, J.J., Griborio, A., Nopens, I., Batstone, D., Wicks, J., Saunders, S., Wicklein, E., Kenny, G., Laurent, J.: Cfd for wastewater treatment: an overview. *Water Science and Technology* **74**(3), 549–563 (2016)
- [5] Darwish, M., Moukalled, F.: The Finite Volume Method in Computational Fluid Dynamics: an Advanced Introduction with OpenFOAM® and Matlab®, pp. 110–128. Springer, Berlin (2016)
- [6] Baker, T.J.: Mesh generation: Art or science? *Progress in Aerospace Sciences* **41**(1), 29–63 (2005)
- [7] Knupp, P.M.: Algebraic mesh quality metrics. *SIAM journal on scientific computing* **23**(1), 193–218 (2001)
- [8] Freitag, L.A., Ollivier-Gooch, C.: Tetrahedral mesh improvement using swapping and smoothing. *International Journal for Numerical Methods in Engineering* **40**(21), 3979–4002 (1997)
- [9] Freitag, L.A., Ollivier-Gooch, C.: A comparison of tetrahedral mesh improvement techniques. Technical report, Argonne National Lab.(ANL), Argonne, IL (United States) (1996)
- [10] Prasad, T.: A comparative study of mesh smoothing methods with flipping in 2d and 3d. PhD thesis, Rutgers University-Camden Graduate School (2018)
- [11] Guo, Y., Wang, C., Ma, Z., Huang, X., Sun, K., Zhao, R.: A new mesh smoothing method based on a neural network. *Computational Mechanics*, 1–14 (2021)
- [12] Herrmann, L.R.: Laplacian-isoparametric grid generation scheme. *Journal of the Engineering Mechanics Division* **102**(5), 749–756 (1976)
- [13] Zhou, T., Shimada, K.: An angle-based approach to two-dimensional mesh smoothing. *IMR* **2000**, 373–384 (2000)
- [14] Du, Q., Gunzburger, M.: Grid generation and optimization based on centroidal voronoi tessellations. *Applied mathematics and computation* **133**(2-3), 591–607 (2002)
- [15] Lloyd, S.: Least squares quantization in pcm. *IEEE transactions on information theory* **28**(2), 129–137 (1982)
- [16] Du, Q., Faber, V., Gunzburger, M.: Centroidal voronoi tessellations: Applications and algorithms. *SIAM review* **41**(4), 637–676 (1999)
- [17] Parthasarathy, V., Kodiyalam, S.: A constrained optimization approach to finite element mesh smoothing. *Finite Elements in*

Analysis and Design **9**(4), 309–320 (1991)

- [18] Field, D.A.: Laplacian smoothing and delaunay triangulations. Communications in applied numerical methods **4**(6), 709–712 (1988)
- [19] Canann, S.A., Tristano, J.R., Staten, M.L.: 三角メッシュ、四辺形メッシュ、四辺形メッシュのためのラプラシアンと最適化に基づく平滑化の組み合わせへのアプローチ。IMR 1 , 479–94 (1998). 出版社 Citeseer
- [20] フィールド、D.A.: ラプラシアン平滑化とデラウネイ三角形分割。応用数値計算法における通信 **4**(6), 709–712 (1988)
- [21] Chen, X., Liu, J., Pang, Y., Chen, J., Chi, L., Gong, C.: Developing a new mesh quality evaluation method based on convolutional neural network. Engineering Applications of Computational Fluid Mechanics **14**(1), 391–400 (2020)
- [22] Wang, Z., Chen, X., Li, T., Gong, C., Pang, Y., Liu, J.: Evaluating mesh quality with graph neural networks. Engineering with Computers **38**(5), 4663–4673 (2022)
- [23] Chen, X., Liu, J., Gong, C., Li, S., Pang, Y., Chen, B.: Mve-net: An automatic 3-d structured mesh validity evaluation framework using deep neural networks. Computer-Aided Design **141**, 103104 (2021)
- [24] Zhang, Z., Wang, Y., Jimack, P.K., Wang, H.: Meshingnet: A new mesh generation method based on deep learning. In: International Conference on Computational Science, pp. 186–198 (2020). Springer
- [25] Zhang, Z., Jimack, P.K., Wang, H.: Meshingnet3d: Efficient generation of adapted tetrahedral meshes for computational mechanics. Advances in Engineering Software **157**, 103021 (2021)
- [26] Daroya, R., Atienza, R., Cajote, R.: Rein: Flexible mesh generation from point clouds. In: Proceedings of the IEEE/CVF Conference
- コンピュータビジョンとパターン認識ワークショップ, pp. 352–353 (2020)
- [27] Papagiannopoulos, A., Clausen, P., Avellan, F.: How to teach neural networks to mesh: Application on 2-d simplicial contours. Neural Networks **136**, 152–179 (2021)
- [28] Chen, X., Li, T., Wan, Q., He, X., Gong, C., Pang, Y., Liu, J.: Mgnet: a novel differential mesh generation method based on unsupervised neural networks. Engineering with Computers **38**(5), 4409–4421 (2022)
- [29] Bohn, J., Feischl, M.: Recurrent neural networks as optimal mesh refinement strategies. Computers & Mathematics with Applications **97**, 61–76 (2021)
- [30] Paszyński, M., Grzeszczuk, R., Pardo, D., Demkowicz, L.: ディープラーニング駆動の自己適応hp有限要素法。In: 計算科学国際会議, pp. 114–121 (2021). シュプリンガー
- [31] Tingfan, W., Xuejun, L., Wei, A., Huang, Z., Hongqiang, L.: A mesh optimization method using machine learning technique and variational mesh adaptation. Chinese Journal of Aeronautics **35**(3), 27–41 (2022)
- [32] Wallwork, J.G., Lu, J., Zhang, M., Pigott, M.D.: E2n: error estimation networks for goal-oriented mesh adaptation. arXiv preprint arXiv:2207.11233 (2022)
- [33] Fidkowski, K.J., Chen, G.: Metric-based, goal-oriented mesh adaptation using machine learning. Journal of Computational Physics **426**, 109957 (2021)
- [34] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems **32**(1), 4–24 (2020)
- [35] Vollmer, J., Mencl, R., Mueller, H.: Improved laplacian smoothing of noisy surface meshes. In: Computer Graphics Forum, vol. 18, pp. 131–138 (1999). Wiley Online Library

- Analysis and Design **9**(4), 309–320 (1991)
- [18] Field, D.A.: Laplacian smoothing and delaunay triangulations. Communications in applied numerical methods **4**(6), 709–712 (1988)
- [19] Canann, S.A., Tristano, J.R., Staten, M.L.: An Approach to Combined Laplacian and Optimization-Based Smoothing for Triangular, Quadrilateral, and Quad-Dominant Meshes. IMR **1**, 479–94 (1998). Publisher: Citeseer
- [20] Field, D.A.: Laplacian smoothing and delaunay triangulations. Communications in applied numerical methods **4**(6), 709–712 (1988)
- [21] Chen, X., Liu, J., Pang, Y., Chen, J., Chi, L., Gong, C.: Developing a new mesh quality evaluation method based on convolutional neural network. Engineering Applications of Computational Fluid Mechanics **14**(1), 391–400 (2020)
- [22] Wang, Z., Chen, X., Li, T., Gong, C., Pang, Y., Liu, J.: Evaluating mesh quality with graph neural networks. Engineering with Computers **38**(5), 4663–4673 (2022)
- [23] Chen, X., Liu, J., Gong, C., Li, S., Pang, Y., Chen, B.: Mve-net: An automatic 3-d structured mesh validity evaluation framework using deep neural networks. Computer-Aided Design **141**, 103104 (2021)
- [24] Zhang, Z., Wang, Y., Jimack, P.K., Wang, H.: Meshingnet: A new mesh generation method based on deep learning. In: International Conference on Computational Science, pp. 186–198 (2020). Springer
- [25] Zhang, Z., Jimack, P.K., Wang, H.: Meshingnet3d: Efficient generation of adapted tetrahedral meshes for computational mechanics. Advances in Engineering Software **157**, 103021 (2021)
- [26] Daroya, R., Atienza, R., Cajote, R.: Rein: Flexible mesh generation from point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 352–353 (2020)
- [27] Papagiannopoulos, A., Clausen, P., Avellan, F.: How to teach neural networks to mesh: Application on 2-d simplicial contours. Neural Networks **136**, 152–179 (2021)
- [28] Chen, X., Li, T., Wan, Q., He, X., Gong, C., Pang, Y., Liu, J.: Mgnet: a novel differential mesh generation method based on unsupervised neural networks. Engineering with Computers **38**(5), 4409–4421 (2022)
- [29] Bohn, J., Feischl, M.: Recurrent neural networks as optimal mesh refinement strategies. Computers & Mathematics with Applications **97**, 61–76 (2021)
- [30] Paszyński, M., Grzeszczuk, R., Pardo, D., Demkowicz, L.: Deep learning driven self-adaptive hp finite element method. In: International Conference on Computational Science, pp. 114–121 (2021). Springer
- [31] Tingfan, W., Xuejun, L., Wei, A., Huang, Z., Hongqiang, L.: A mesh optimization method using machine learning technique and variational mesh adaptation. Chinese Journal of Aeronautics **35**(3), 27–41 (2022)
- [32] Wallwork, J.G., Lu, J., Zhang, M., Pigott, M.D.: E2n: error estimation networks for goal-oriented mesh adaptation. arXiv preprint arXiv:2207.11233 (2022)
- [33] Fidkowski, K.J., Chen, G.: Metric-based, goal-oriented mesh adaptation using machine learning. Journal of Computational Physics **426**, 109957 (2021)
- [34] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y.: A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems **32**(1), 4–24 (2020)
- [35] Vollmer, J., Mencl, R., Mueller, H.: Improved laplacian smoothing of noisy surface meshes. In: Computer Graphics Forum, vol. 18, pp. 131–138 (1999). Wiley Online Library

- [36] Xu, K., Gao, X., Chen, G.: Hexahedral mesh quality improvement via edge-angle optimization. *Computers & Graphics* **70**, 17–27 (2018)
- [37] Reddy, D.R.: Speech recognition by machine: A review. *Proceedings of the IEEE* **64**(4), 501–531 (1976)
- [38] LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *nature* **521**(7553), 436–444 (2015)
- [39] Pak, M., Kim, S.: A review of deep learning in image recognition. In: 2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT), pp. 1–3 (2017). IEEE
- [40] Chowdhary, K., Chowdhary, K.: 自然言語処理. 人工知能の基礎, 603–649 (2020)
- [41] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
- [42] Song, W., Zhang, M., Wallwork, J.G., Gao, J., Tian, Z., Sun, F., Piggott, M., Chen, J., Shi, Z., Chen, X., et al.: M2n: mesh movement networks for pde solvers. *Advances in Neural Information Processing Systems* **35**, 7199–7210 (2022)
- [43] Lino, M., Cantwell, C., Bharath, A.A., Fotiadis, S.: Simulating continuum mechanics with multi-scale graph neural networks. arXiv preprint arXiv:2106.04900 (2021)
- [44] Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., Battaglia, P.W.: Learning mesh-based simulation with graph networks. arXiv preprint arXiv:2010.03409 (2020)
- [45] Han, X., Gao, H., Pfaff, T., Wang, J.-X., Liu, L.-P.: Predicting physics in mesh-reduced space with temporal attention. arXiv preprint arXiv:2201.09113 (2022)
- [46] Peng, W., Yuan, Z., Wang, J.: 亂流シミュレーションのための注意強化型ニューラルネットワークモデル. 流体の物理 34(2) (2022)
- [47] Li, G., Muller, M., Thabet, A., Ghanem, B.: Deepgcns: Can gcns go as deep as cnns? In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9267–9276 (2019)
- [48] 蔡、T.、羅、S.、徐、K.、何、D.、劉、T.-y.、王、L.: Graphnorm: グラフニューラルネットワークの学習を加速するための原理的なアプローチ。In: 機械学習国際会議, pp. 1204–1215 (2021). PMLR
- [49] Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)
- [50] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
- [51] Lee, D.-T., Schachter, B.J.: 脱則的三角測量を構築するための2つのアルゴリズム。国際コンピュータ情報科学ジャーナル 9(3), 219–242 (1980)

- [36] Xu, K., Gao, X., Chen, G.: Hexahedral mesh quality improvement via edge-angle optimization. *Computers & Graphics* **70**, 17–27 (2018)
- [37] Reddy, D.R.: Speech recognition by machine: A review. *Proceedings of the IEEE* **64**(4), 501–531 (1976)
- [38] LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. *nature* **521**(7553), 436–444 (2015)
- [39] Pak, M., Kim, S.: A review of deep learning in image recognition. In: 2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT), pp. 1–3 (2017). IEEE
- [40] Chowdhary, K., Chowdhary, K.: Natural language processing. Fundamentals of artificial intelligence, 603–649 (2020)
- [41] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
- [42] Song, W., Zhang, M., Wallwork, J.G., Gao, J., Tian, Z., Sun, F., Piggott, M., Chen, J., Shi, Z., Chen, X., et al.: M2n: mesh movement networks for pde solvers. *Advances in Neural Information Processing Systems* **35**, 7199–7210 (2022)
- [43] Lino, M., Cantwell, C., Bharath, A.A., Fotiadis, S.: Simulating continuum mechanics with multi-scale graph neural networks. arXiv preprint arXiv:2106.04900 (2021)
- [44] Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A., Battaglia, P.W.: Learning mesh-based simulation with graph networks. arXiv preprint arXiv:2010.03409 (2020)
- [45] Han, X., Gao, H., Pfaff, T., Wang, J.-X., Liu, L.-P.: Predicting physics in mesh-reduced space with temporal attention. arXiv preprint arXiv:2201.09113 (2022)
- [46] Peng, W., Yuan, Z., Wang, J.: Attention-enhanced neural network models for turbulence simulation. *Physics of Fluids* **34**(2) (2022)
- [47] Li, G., Muller, M., Thabet, A., Ghanem, B.: Deepgcns: Can gcns go as deep as cnns? In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9267–9276 (2019)
- [48] Cai, T., Luo, S., Xu, K., He, D., Liu, T.-y., Wang, L.: Graphnorm: A principled approach to accelerating graph neural network training. In: International Conference on Machine Learning, pp. 1204–1215 (2021). PMLR
- [49] Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)
- [50] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
- [51] Lee, D.-T., Schachter, B.J.: Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences* **9**(3), 219–242 (1980)