IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 18, 2020, accepted April 1, 2020, date of publication April 8, 2020, date of current version April 27, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2986488

A Data-Driven Approach for Simultaneous Mesh Untangling and Smoothing Using Pointer Networks

JIBUM KIM[1], JUNHYEOK CHOI[2], AND WOOCHUL KANG[3], (Member, IEEE)

[1]Department of Computer Science and Engineering, Incheon National University, Incheon 22012, South Korea
[2]SK Hynix Inc., Icheon 17336, South Korea
[3]Department of Embedded Systems Engineering, Incheon National University, Incheon 22012, South Korea

Corresponding author: Woochul Kang (wchkang@inu.ac.kr)

This work was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korea Government (MSIT) under Grant NRF-2020R1A2C1007917, and in part by the Incheon National University Research Grant, in 2019.

ABSTRACT Poorly-shaped and/or inverted elements negatively affect numerical simulation accuracy and efficiency. Most current approaches consider mesh quality improvement and mesh untangling problems as numerical optimization problems and solve them using nonlinear optimization. However, these optimization-based approaches require users to set and solve complex numerical optimization problems with no guarantee that the output meshes are valid meshes with good element qualities. Therefore, this paper proposes a data-driven approach for simultaneous mesh untangling and smoothing using a Pointer network. The proposed approach generates various triangular meshes and employs a novel sequence generation algorithm to train the Pointer network and predicts competitive approximate solutions using the trained network. The strength of the proposed framework lies in its simplicity to predict the best free vertex candidate, providing high-quality output meshes, since it does not require solving complex numerical optimization for prediction. Experimental results show that the proposed framework successfully eliminates inverted elements on the meshes and improved average and worst element quality up to 85.9% and 97.8%, respectively, compared to current optimization-based methods.

INDEX TERMS Mesh smoothing, mesh untangling, pointer network, deep learning, recurrent neural network.

I. INTRODUCTION

Mesh quality significantly affects both solution accuracy and speed for numerical simulations [1]–[3]. Poor-quality mesh elements have large condition numbers of the resulting systems, which negatively affect the accuracy of the numerical solution [1], [3]–[5]. Mesh smoothing is a common method to improve mesh quality by simply moving the vertices while fixing the mesh topology.

One of the simplest and most popular mesh smoothing methods is Laplacian smoothing, which moves the position of a vertex by the average of its neighborhood positions [6], [7]. However, Laplacian smoothing can produce inverted elements for non-convex polygons [4], [7], where inverted elements have negative area. Meshes with inverted element(s) are called tangled.

Optimization-based smoothing is often used as an alternative to Laplacian smoothing [8]–[10], and generally produces better mesh quality. It uses the mesh quality metric to evaluate element quality and formulates a numerical optimization problem that guides vertex movement to improve the quality [11]. Various nonlinear solvers, such as nonlinear steepest descent and conjugate gradient methods, have been developed to solving the numerical optimization problem [10], [12].

Tangled meshes produce invalid numerical solutions. Therefore, inverted elements are removed using mesh untangling methods [13]–[15]. Previous mesh untangling methods were mainly based on numerical optimization, similar to mesh smoothing [15]. However, optimization-based mesh untangling is not guaranteed to remove all inverted elements,

The associate editor coordinating the review of this manuscript and approving it for publication was Chao Tong.

and output mesh quality is often poor after removing all inverted elements [16]–[20]. To date, there does not exist an algorithm, which guarantees an untangled mesh, even if it is known to have an untangling solution [18].

Many previous studies have considered optimization methods to simultaneously eliminate inverted elements and improve mesh qualities [4], [16], [21], but to the best of our knowledge, none of the previous works successfully remove inverted elements for all input meshes.

Optimization-based mesh untangling (also, optimization-based simultaneous mesh untangling and smoothing) methods fail mainly due to two reasons. First, the mesh cannot be untangled via vertex movement [20]. Second, an untangled mesh exists when the global minimum of the objective function is achieved, but the mesh optimization often only finds a local minimum, not a global minimum [20].

Therefore, this paper propose an alternative data-driven approach for simultaneous mesh untangling and smoothing using Pointer networks (Ptr-Net) [22]. In contrast with previous approaches, the proposed Ptr-Net based framework adopts a data-driven approach and does not require the user to solve complex numerical optimization problems to predict the best free vertex location and produces high-quality output meshes. We define a free vertex as an interior vertex in the mesh that needs be relocated to improve element quality.

Ptr-Net is a recurrent neural network (RNN) based deep learning model, and have received considerable recent attention, changing traditional sequence-to-sequence (seq2seq) networks such that output dimensions can be flexibly changed depending on the input length. Thus, Ptr-Nets can be used for problems with discrete outputs corresponding to input sequence positions [22], using attention as a pointer to choose a member of the input sequence as the target [22]. They have been successfully applied to solve many complex problems and provide approximate solutions [23]–[25]. Vinyals *et al.* used data-driven Ptr-Net approaches for three challenging geometric combinatorial problems [22]: finding planar convex hulls, generating Delaunay triangulation, and planar traveling salesman problems. They showed that an attention mechanism could solve those problems and used Ptr-Nets to learn approximate solutions. Gu *et al.* applied Ptr-Nets to solve the $0-1$ knapsack problem, successfully finding the approximate solution [24], and they have also been employed to sort numbers [25]. Mottini *et al.* used a Ptr-Net for airline itinerary prediction with promising results [23].

We employ a Ptr-Net deep learning model for our problems since the Ptr-Net output dimensions can be flexibly changed depending to input length and the model learn solutions that have output dictionary size depending on the number of input sequence elements. Moreover, the model can learn optimal free vertex locations and predict competitive approximate solutions.

Our proposed framework includes a training phase and a prediction phase. Training datasets are generated and the Ptr-Net trained to learn the best free vertex location among various vertex candidates. The subsequent prediction phase is performed for input test meshes to produce the high-quality output meshes with no inverted elements. Input test meshes are decomposed into multiple patches and the prediction phase is repeatedly performed for each patch until all patches in the mesh are visited. The best free vertex candidate is then chosen from the Ptr-Net output sequence.

The main contributions of our work are summarized as follows:

- To the best of our knowledge, this work is the first simultaneous mesh untangling and smoothing framework using a data-driven approach with Ptr-Nets.
- The Ptr-Net based framework can learn competitive approximate solutions compared with optimization-based approaches, and produce high-quality meshes with no inverted elements.
- We propose a novel sequence generation algorithm to train the Ptr-Net to simultaneous untangle and smooth the input mesh, and a simple algorithm to choose the best free vertex candidate among various vertex candidates.

The remainder of paper is organized as follows. The prior works are summarized in Section II. Section III presents the existing mesh smoothing and untangling methods. Ptr-Net is described in Section IV. The proposed framework for simultaneous mesh untangling and smoothing is described in Section V. Numerical experiments are presented in Section VI. Discussions and limitations are presented in Section VII. Finally, we conclude the paper in Section VIII.

II. RELATED WORK

Mesh smoothing is a well-studied problem in various research fields. Most previous mesh smoothing studies have employed numerical optimization to improve average element qualities [4], [10], [26]. However, improving average mesh quality can often retain a few poor-quality mesh elements due to geometric constraints. Even a few poor-quality elements negatively affect numerical simulation solution efficiency and accuracy. Thus, many previous studies have considered how to improve the worst quality element in a given mesh, which forms a non-smooth unconstrained optimization problem. Freitag and Plassman proposed an active set algorithm to improve the worst quality element by maximizing the minimum element area in the mesh [9], whereas Sastry *et al.* proposed a log-barrier interior point method [16], and Park *et al.* and Kim *et al.* investigated several derivative-free mesh quality improvement methods [12], [27].

Several previous studies have also investigated how to remove inverted elements in the mesh. Modern simulation packages, such as finite element analysis software, cannot handle tangled meshes, producing erroneous results. Therefore, all inverted elements must be eliminated before numerical simulations are conducted. Bhowmick and Shontz proposed a force-directed mesh untangling algorithm [28] and Knupp proposed a penalty-based mesh untangling method using numerical optimization [15]. However, these methods often produce tangled meshes and/or output meshes with poor element quality.

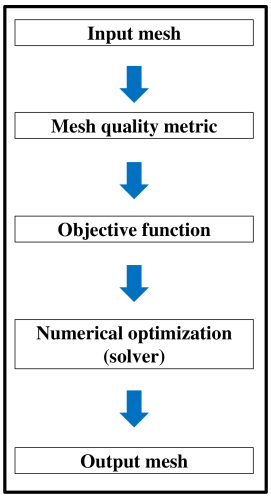


FIGURE 1. Optimization-based mesh smoothing and untangling.

Various simultaneous mesh untangling and smoothing methods have been proposed to overcome these limitations. Escobar *et al.* proposed simultaneous mesh untangling and smoothing methods using a modified objective function [21]. Kim *et al.* proposed a multi-objective mesh optimization framework to simultaneously untangle inverted elements and improve mesh quality [4], combining two or more competing objective functions into a single objective function and solving it using nonlinear optimization. Geometric element transformation was recently proposed for 2D and 3D simultaneous mesh untangling and smoothing [29]. However, simultaneous mesh untangling and smoothing methods sometimes produce tangled meshes and require for the user to solve complex optimization problems with robust solvers.

On the other hand, a novel population-based optimization algorithm based on balancing composite motions was proposed [30]. This approach was proved to be a potential tool in association with deep learning. However, to the best of our knowledge, none of the previous approaches applied deep learning-based data-driven approaches for simultaneous mesh untangling and smoothing.

III. EXISTING APPROACH: OPTIMIZATION-BASED MESH SMOOTHING AND UNTANGLING

Numerical optimization is widely used for mesh smoothing and untangling. For example, Figure 1 shows a typical approach using optimization-based mesh smoothing and untangling. Mesh smoothing and untangling follow the same procedure.

First, the user defines a desired mesh quality metric and the corresponding ideal element. The ideal element is generally defined as an equilateral triangle or tetrahedron in 2D or 3D, respectively. Then mesh optimization makes actual element shapes as close as possible to the ideal element shape. The next step is to formulate an objective function with respect to mesh vertex coordinates to describe overall mesh (element) quality [11]. Then numerical optimization uses this objective function to find local optima and hence the optimal mesh.

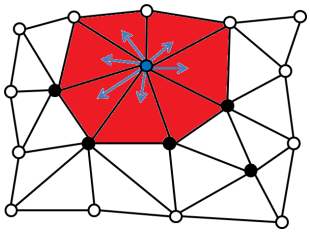


FIGURE 2. A simple mesh with 6 free vertices. A free vertex (blue) and the layer of elements adjacent to a free vertex constitute a single patch (red). The free vertex moves to a new vertex location for improving the local mesh quality.

Numerical optimization schemes commonly include steepest descent or conjugate gradient methods.

Users must first eliminate inverted elements in tangled meshes, i.e., untangle the mesh, and subsequently smooth the mesh to improve mesh qualities as a two-step process. Another approach is to perform simultaneous mesh untangling and smoothing methods [4], [16], [21], [29].

A. MESH SMOOTHING

Mesh smoothing improves mesh (element) quality by relocating vertex locations, while keeping the original mesh topology. Global or local mesh smoothing strategies are commonly employed. Global mesh smoothing moves all free (interior) vertices simultaneously to improve element quality, using a global objective function to simultaneously improve element quality. In contrast, local mesh smoothing (Gauss-Seidel type) subdivides the entire mesh into discrete patches for each free vertex and iteratively performs mesh smoothing for each patch until user-defined termination criteria are met [11]. For local mesh smoothing, the order free vertices are moved can affect final output mesh quality and/or efficiency [31].

Figure 2 shows that each patch includes the layer of elements adjacent to a free vertex. Boundary vertices are fixed, i.e., cannot be moved during mesh smoothing, hence a free vertices is an interior vertex that is able to move freely. The example in Fig. 2 considers 6 free vertices. Global mesh smoothing generally requires considerably more computation time and memory than local mesh smoothing [29]. We employ local mesh smoothing for this study.

Mesh quality is a measure of how the shape or the size of the element is close to the ideal element. In this paper, we use a 2D triangular element and use the inverse mean ratio (IMR) quality metric, which considers both the shape and size of the element [26]. Let the three vertex coordinates of the element be (a, b, c). Then, the incidence matrix, A, is denoted as follows:

$$A = [b - a, \text{c-a}]\,. \tag{1}$$

We assume that the ideal element is an equilateral triangle. Then, the incidence matrix of the ideal element is follows:

$$W = \begin{pmatrix} 1 & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} \end{pmatrix}. \tag{2}$$

For an i^{th} element, the IMR quality metric is computed as follows:

$$q_{i,IMR} = \frac{\left\|AW^{-1}\right\|_F^2}{2\left|\det\left(AW^{-1}\right)\right|}, \tag{3}$$

where $\|\cdot\|_F$ is a Frobenius norm. When the current element and ideal element has same shape and size, the AW^{-1} has the identity matrix. The range of the IMR quality metric ranges between one and infinity and a smaller value close to one indicates a better element quality [4]. We use the square root of sum of the squared values (i.e., L^2 norm), $F = \sum_{i\in E}\left(q_{i,IMR}\right)^2$, to improve the average quality of the mesh. Nonlinear optimization solvers (e.g., steepest descent or conjugate gradient) can be used to find the local optimal point.

B. MESH UNTANGLING

The inverted elements in the mesh are defined as elements whose signed areas are non-positive. The goal of mesh untangling is to eliminate inverted elements in the mesh. Most existing optimization-based mesh untangling methods give a high penalty (cost) to the inverted elements. Similar to the local mesh smoothing, local mesh untangling is performed by iteratively visiting over the set of patches. For an i^{th} element, the untangling quality metric in [12], [15] is computed as follows:

$$q_{i,UNT} = \frac{1}{2}\left(|A-\beta| - (A-\beta)\right), \tag{4}$$

where A is a signed area of i^{th} element and $\beta\,(>0)$ is a user-defined parameter. We use the L^2 norm to incorporate the element qualities and formulate the objective function as follows:

$$F = \sum_{i\in E}\left(q_{i,UNT}\right)^2, \tag{5}$$

where E is an element set in the patch. Similar to the mesh smoothing problem, nonlinear optimization methods can be used to find the local optimal point.

IV. POINTER NETWORK

The proposed framework employs a Ptr-Net-based deep learning model for simultaneous mesh untangling and smoothing. The framework generates input and output sequences to train the Ptr-Net and predict new free vertex locations using the output sequence from the trained Ptr-Net. This section briefly reviews traditional seq2seq and content-based input attention models as baselines for Ptr-Nets, and we then describe the Ptr-Net we employed for the proposed framework.

A. SEQUENCE-TO-SEQUENCE NETWORK

Sequence-to-sequence networks use two different recurrent neural networks (RNNs) as an encoder and decoder [32]. Seq2seq networks use basic RNN, long short-term memory (LSTM) [33], and gated recurrent unit (GRU) [34] RNN cells. The encoder receives time series input data sequentially, and outputs a thought vector. The decoder receives both the thought vector and its previous output as input and produces the final output.

Let the input sequence and corresponding output sequence data pair be (I, O), then, the seq2seq network computes the conditional probability as follows:

$$p\left(O|I;\theta\right) = \prod_{i=1}^{m(I)} p_\theta\left(O_i|O_1,\ldots,O_{i-1},I;\theta\right), \tag{6}$$

where $I = \{I_1, I_2, \ldots, I_n\}$ is a time series value for n input data and $O = \left\{O_1, O_2, \ldots, O_{m(I)}\right\}$ is a sequence of $m(I)$ indices between $[1, n]$. The parameter θ of the seq2seq model is trained to maximize the conditional probability of the training input and output data pairs as follows:

$$\theta^* = \underset{\theta}{\operatorname{argmax}} \sum_{I,O} \log\left(p\left(O|I;\theta\right)\right), \tag{7}$$

where the sum is over training examples. The output dimensionality is fixed by the problem dimensionality and is the same during training and inference [22].

B. CONTENT-BASED INPUT ATTENTION

The content-based input attention encoder is identical to the seq2seq network, but it transforms the output process in the decoder. The content-based input attention model compares the decoder output with each encoder hidden state to create a new context vector, and combines it with the hidden decoder state to calculate the final output [35], [36].

Let $(e_1, \ldots, e_n)$ and $\left(d_1, \ldots, d_{m(I)}\right)$ be the encoder and decoder hidden states, respectively. Then, the attention vector at each output time i is computed as follows:

$$\begin{aligned} u_j^i &= v^T \tanh\left(W_1 e_j + W_2\, d_i\right), j \in (1,\ldots,n) \\ \alpha_j^i &= \operatorname{softmax}\left(u_j^i\right), j \in (1,\ldots,n), \end{aligned} \tag{8}$$

where softmax normalizes the vector u^i to be the attention over the inputs [22]; and v, W_1, and W_2 are learned parameters. Then, a context vector is created as the weighted sum of attention weights and the hidden states of the encoder, i.e.

$$c_i = \sum_{j=1}^{n} \alpha_j^i e_j. \tag{9}$$

The final output attention vector is calculated through the activation function. The content-based input attention model performs significantly better than the pure seq2seq model on convex hull problems, but it is not applicable to problems where the output dictionary size depends on the input [22].

C. POINTER NETWORK

The seq2seq network uses a softmax distribution over a fixed sized output dictionary and cannot be used for problems where the output dictionary size is equal to the length of the input sequence [22]. In contrast, Ptr-Nets adapt the attention mechanism to create pointers to elements in the

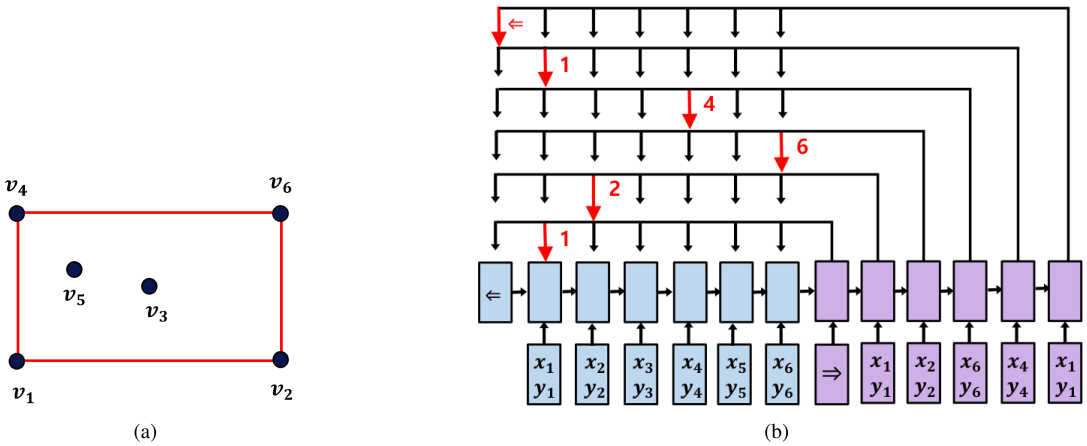


FIGURE 3. (a) Convex hull problem (b) Ptr-Net model. The inputs of the Ptr-Net are planar points, $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, where $v^i = (x_i, y_i)$ are the Cartesian coordinates of the points and the output sequence represents the solution of the convex hull problem, $\{\Rightarrow, 1, 2, 6, 4, 1, \Leftarrow\}$.

input sequence [22], [23], [37], using the attention vector, u_j^i, as pointers to input elements for each output time i using the attention mechanism:

$$u_j^i = v^T \tanh\left(W_1 e_j + W_2 d_i\right), j \in (1, \cdots, n)$$
$$p\left(O_i | O_1, \ldots, O_{i-1}, I\right) = \text{softmax}\left(u^i\right), \quad (10)$$

where softmax normalizes u^i to be an output distribution over the dictionary of inputs; and v, W_1, and W_2 are learned parameters of the output model [22].

This has the feature of choosing an element of the input sequence to output. Many combinatorial optimization problems are NP-hard and brute-force algorithms are often not tractable. Therefore, approximate solutions using data-driven approaches are often preferred to find near-optimal solutions to hard problems. However, these data-driven approaches require huge supervised labeled datasets (e.g. 1M training examples) for training [22].

Previous studies employ the convex hull problem as a baseline to develop the model as a data-driven approach [22]. Figure 3 shows an example Ptr-Net to find the convex hull of a finite number of points (vertices). Ptr-Net inputs are planar points, $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, where $v^i = (x_i, y_i)$ are Cartesian coordinates of the vertices; the output sequence is the solution for the convex hull problem, $\{\Rightarrow, 1, 2, 6, 4, 1, \Leftarrow\}$; and $\Rightarrow$ and $\Leftarrow$ represent the sequence beginning and end, respectively. The attention mechanism output is a softmax distribution with dictionary size equal to the input length [22].

To represent the output as a sequence and reduce ambiguity, the vertex with the lowest index is selected as the starting vertex and proceeds counter-clockwise from there. The encoder feeds the input to the sequence end ($\Rightarrow$), then the model switches to decoder mode, generating the output sequence one at a time until $\Leftarrow$.

V. PROPOSED FRAMEWORK

The proposed framework includes training and prediction phases. During the training phase, training datasets are generated and the Ptr-Net learns best free vertex locations, producing the high-quality output meshes with no inverted elements. The subsequent prediction phase is only performed for input test meshes. The input mesh is decomposed into patches and the prediction phase is applied locally by repeatedly iterating over the patches, allowing each free vertex to move and relocate. Different from the original Ptr-Net, a simple post-processing step is added to find the best free vertex candidate from the Ptr-Net output sequence during the prediction phase.

A. TRAINING PHASE

1) DATASETS GENERATION

The training phase comprises two steps, as shown in Fig. 4. Various polygon shapes are generated for training, and we perform polygon triangulation by decomposing the polygons into multiple triangles, generating a triangular mesh with one interior (free) vertex and connecting edges between each boundary and free vertex. The free vertex is randomly located inside a patch.

The optimal free vertex location, i.e., the location that provides the best average element quality of triangular elements, is computed using Mesquite software [11]. It applies optimization-based mesh smoothing to improve mesh quality. The proposed framework assumes that the ideal element shape is an equilateral triangle; and we use the IMR quality metric in (3) as the mesh quality metric, the L^2 norm to formulate the objective function, and nonlinear steepest descent to find the optimal output mesh.

After completing mesh quality improvement, the final step generates random vertices inside the polygon, and optimal free vertex and randomly generated vertices inside the

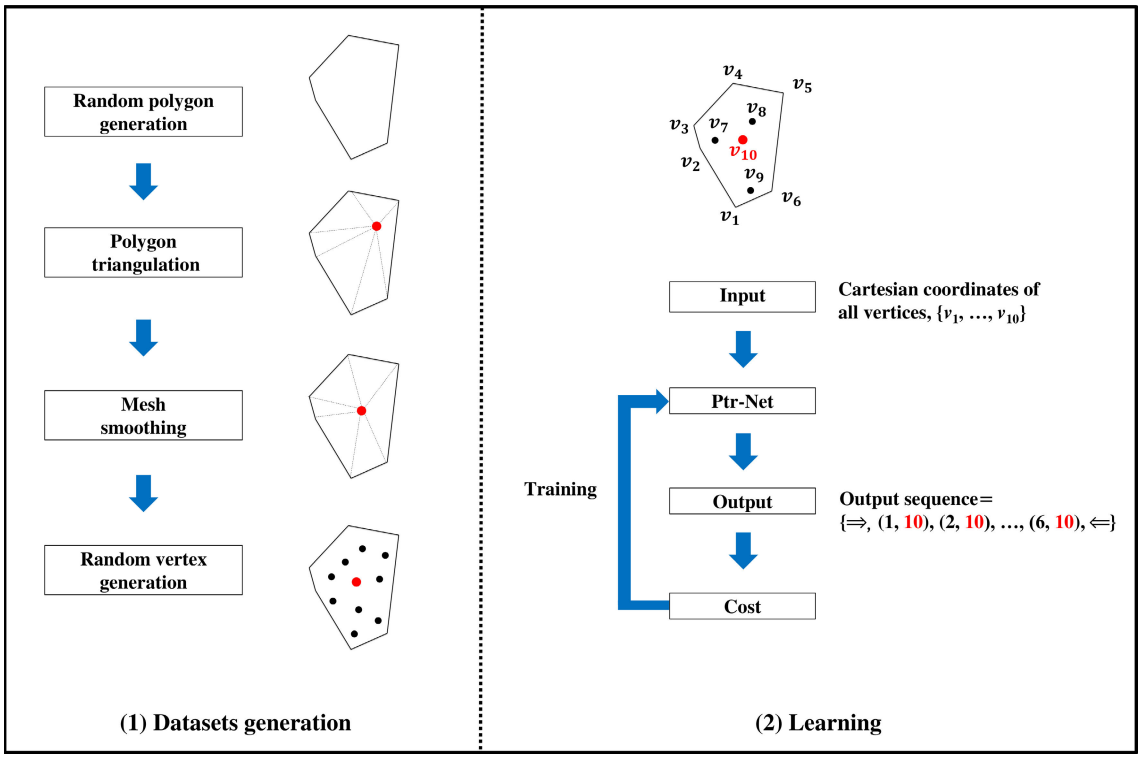


FIGURE 4. Training phase.

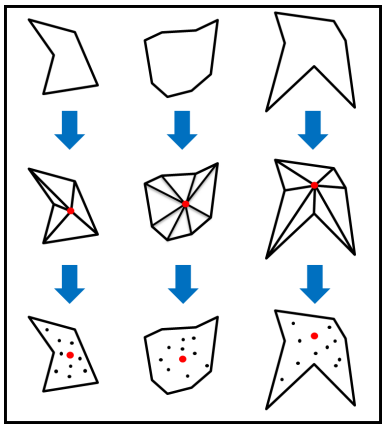


FIGURE 5. Dataset generation with various polygon shapes. Here, red and black vertices are the optimal free vertex location and randomly generated vertices inside a polygon, respectively.

polygon are used to train the Ptr-Net. We generate random vertices using barycentric coordinates by circulating the triangular elements inside each polygon. Given three vertices of a triangular element, P_0, P_1, and P_2, vertex P inside the triangle can be expressed as:

$$P = \lambda_0 P_0 + \lambda_1 P_1 + \lambda_2 P_2, \lambda_0 + \lambda_1 + \lambda_2 = 1, \qquad (11)$$

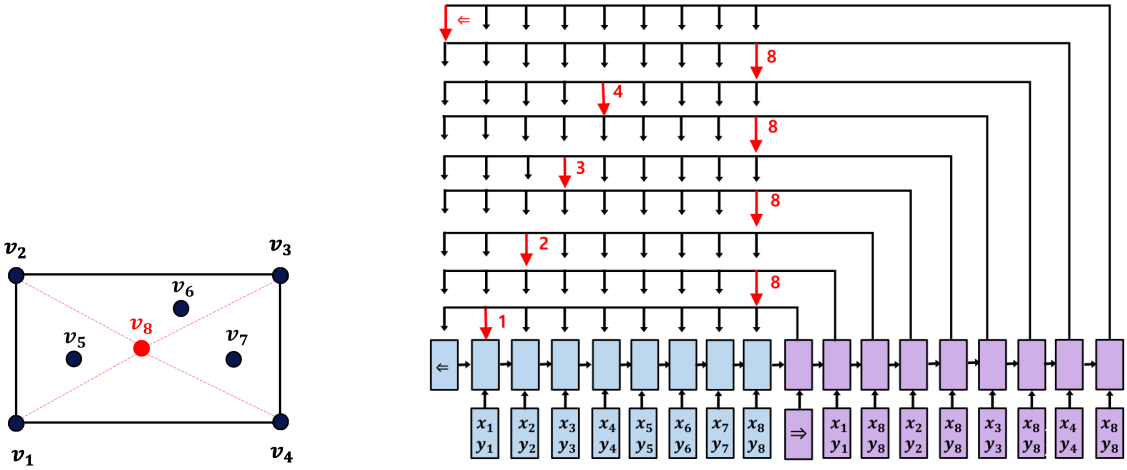
where λ_0, λ_1, and $\lambda_2 > 0$ are barycentric coordinates for P. Figure 5 shows several generated polygons, triangulation, optimal free vertex (red), and randomly generated vertex examples.

2) LEARNING

The Ptr-Net is trained on the generated datasets to learn the best free vertex position to provide the best output element quality for each patch. Ptr-Net inputs for each polygon are the Cartesian coordinates for all vertices, including boundary and interior vertices, where the interior vertices included the optimal free vertex found by Mesquite and randomly generated vertex candidates. For input data normalization, we normalize the input data x and y coordinates over [0, 1] for each random polygon. Input sequence length may vary depending on the polygon shape, which has different numbers of boundary vertices.

We create a sequence that crossed the boundary vertex indices and the optimal free vertex as the Ptr-Net output sequence. Let I_b^i be the index for the i^{th} boundary vertex and I_O be the index for the optimal free vertex in the polygon. Then, the created output sequence is $\{\Rightarrow, (I_b^1, I_O), (I_b^2, I_O), \cdots, (I_b^N, I_O), \Leftarrow\}$, where N is the number of boundary vertices in the polygon. By repeatedly placing boundary vertices and the optimal free vertex, we expect that the model will learn that even-numbered indices from the output sequence except the beginning/end of sequence tokens correspond to the optimal free vertex.

Figure 6 shows example input/output for training the Ptr-Net for a simple polygon with 8 vertices. Interior vertex candidates are $\{v_5, v_6, v_7, v_8\}$ with optimal free vertex v_8. For this example, four triangular elements using v_8 are closer to ideal elements and hence produced better element quality



(a) Polygon and the optimal free vertex location (v_8, red). The triangular mesh using v_8 is represented as a dotted line.

(b) Input $P = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\}$ and the output sequence $\{\Rightarrow, (1, 8), (2, 8), (3, 8), (4, 8), \Leftarrow\}$.

FIGURE 6. Training example for a simple polygon with 8 vertices. (a) Polygon (b) input and output representation, for training the Ptr-Net. The tokens ⇒ and ⇐ represent the sequence beginning and end, respectively.

than the other vertex candidates. Ptr-Net input is the Cartesian coordinates of the vertices, $\{v_1, \cdots, v_8\}$, and output sequence is $\{\Rightarrow, (1, 8), (2, 8), (3, 8), (4, 8), \Leftarrow\}$.

The input/output sequence ordering affects Ptr-Net performance. Previous studies have indicated that the model suffers when awkward ordering is chosen [25]. We follow clockwise ordering to avoid ambiguity and randomly allocated index numbers to avoid over-fitting.

B. PREDICTION PHASE

Figure 7 shows the prediction phase. The input mesh is decomposed into a set of patches [11] and the prediction phase repeatedly visits the set of patches, performing each patch separately. This decomposition process is similar to the local mesh smoothing (Section III). First, random free vertex location candidates are generated for each patch using barycentric coordinates. Then we use the trained Ptr-Net to predict the new free vertex location among the generated candidates. The Ptr-Net inputs are the Cartesian coordinates for all vertices in the patch, and the trained Ptr-Net produced an output sequence, which is subsequently used to predict the new free vertex location.

Excluding the beginning/end of sequence tokens, the trained Ptr-Net model automatically learns that the best free vertex index occurs in the even-numbered position of the output sequence. We count the occurrence of even-numbered integers from the output sequence except the beginning/end of sequence tokens and choose the most frequent index, i, from the output sequence.

Algorithm 1 shows the proposed algorithm to select the best free vertex index from the output sequence for a single patch. For example, suppose the Ptr-Net output sequence $= \{\Rightarrow, (1, 8), (2, 6), (3, 8), (4, 8), \Leftarrow\}$. Then, the most frequent

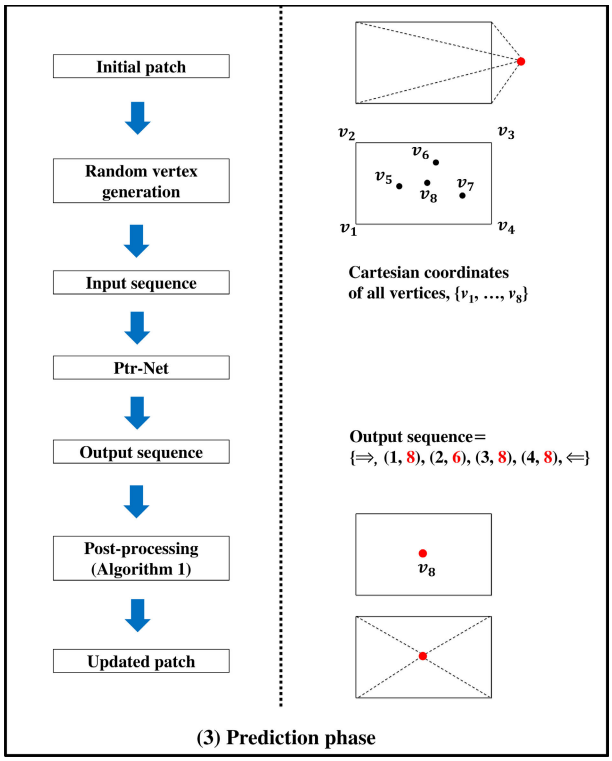


FIGURE 7. Prediction phase. For an input mesh, the prediction phase is repeated for all patches in the mesh. The figure on the right shows an example.

index in the even-numbered locations $= 8$ except the beginning/end of sequence tokens. Therefore, we choose v_8 as the best free vertex index and update the new free vertex location as v_8. Element connectivity is maintained during the prediction phase and only the free vertex location is updated. The prediction phase is repeated for all patches in the mesh.

Algorithm 1 Choose the Best Free Vertex Index

Input: Output sequence of the Ptr-Net

Output: The vertex index i

1: Count the occurrence of even-numbered integers from the output sequence except the beginning/end of sequence tokens

2: Find the integer, i, with highest occurrence

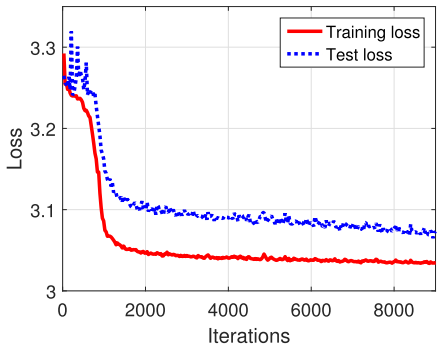


FIGURE 8. Learning curve.

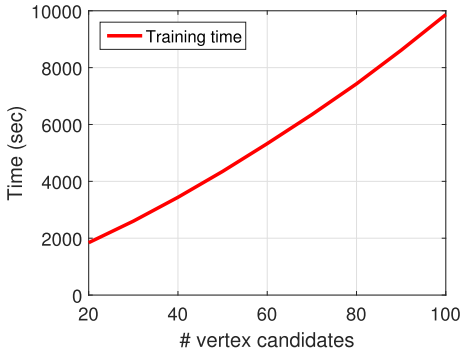


FIGURE 9. Training time.

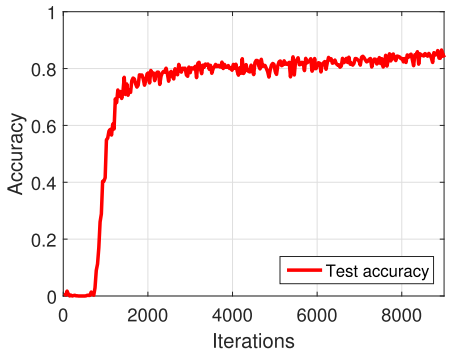


FIGURE 10. Test accuracy.

VI. EXPERIMENTS

A. EXPERIMENTAL DETAILS

1) TRAINING

We use a Tensorflow library to implement the proposed framework, and use the same architecture throughout all the experiments. All our models use 512 LSTM hidden units, and we initialize the weights with the uniform random distribution in the range [0, 1]. The Adam optimizer is used for learning with learning rate $= 0.001$. We set the batch size to 50 such that the Ptr-Net is trained by putting 50 patches of various polygon shapes in each iteration. A total of 20,000 polygons (10,000 convex and 10,000 non-convex) is generated for training various polygon shapes. Preliminary experiments show there is no significant performance difference even after increasing the number of polygons to more than 20,000.

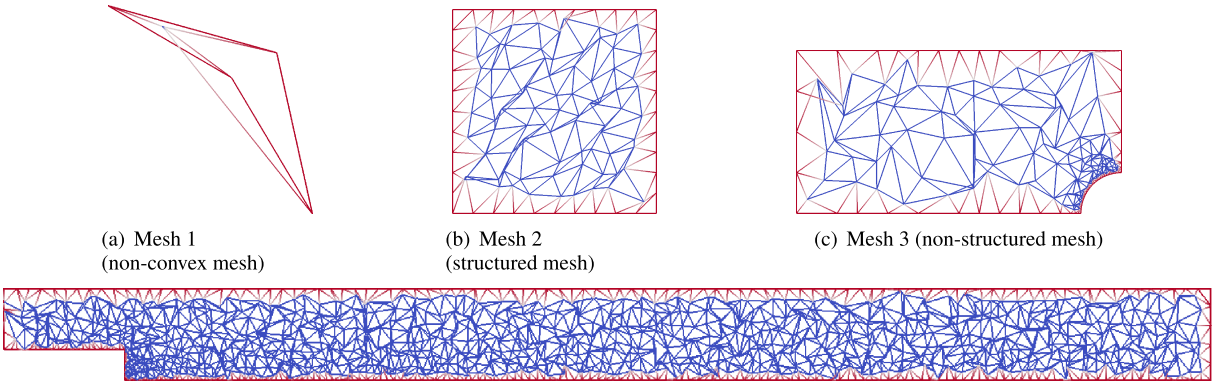
Our preliminary experiments show that increasing the capacity of the network by adding more hidden units slightly improves the accuracy but increases the training time. We also observe that tuning the learning rate by using learning rate annealing can improve the convergence speed. We do not tune all hyper-parameters of the Ptr-Net model, such as the learning rate, regularization, and batch size, since our goal is to show the possibility of using a data-driven approach for performing simultaneous mesh untangling and smoothing.

Figure 8 shows the learning curve for the proposed framework. Each iteration includes training performed once by putting a batch in the model. Maximum training iterations is set to 9K since loss converged sufficiently when iteration reached 9K. Figure 9 shows the training time with respect to the number of vertex candidates in each polygon during training. Training time increases linearly with increasing number of vertex candidates. Increasing the number of vertex candidates also tends to improve overall mesh quality (average and worst), but there are no significant differences when number of vertex candidates > 50. Therefore, we set the number of vertex candidates in each patch $= 50$ during training and prediction phases.

Figure 10 shows the proposed framework test accuracy using labeled data. Test accuracy increases to approximately 85% as iterations increase. Therefore, the model chooses the best vertex candidate approximately 85% of the time after training.

We observe that the output sequence generation method during the prediction phase highly affects the overall training time and accuracy. We propose the novel output sequence generation method to learn the optimal free vertex location that crosses the boundary vertex indices and the optimal free vertex. If we place all boundary vertex indices and the optimal free vertex index at the beginning and the end of the output sequence, respectively, then the accuracy is significantly reduced compared with the proposed output sequence generation method. Similar results are observed when the vertices are placed in reverse order.

The advantage of the Ptr-Net model is that it can produce variable-sized output dictionaries when the input size is variable length, while the seq2seq models require the size of the output dictionary to be fixed *a priori* [22]. During the prediction phase, depending on the shape of the patch (polygon) in the mesh, it may be necessary to produce the output sequence even for the untrained input length. Ptr-Net model does extrapolate to lengths that it has never seen during training [22]. Our experimental results show that the accuracy



(a) Mesh 1 (non-convex mesh)

(b) Mesh 2 (structured mesh)

(c) Mesh 3 (non-structured mesh)

(d) Mesh 4 (non-structured mesh)

FIGURE 11. Test meshes.

TABLE 1. The test mesh configurations. The IMR quality metric is used for measuring the element quality.

Mesh name	# vertices	# elements	# inverted elements	Average element quality	Worst element quality
Mesh 1 (non-convex)	4	4	1	5.62	14.30
Mesh 2 (structured)	121	200	12	1.86	34.20
Mesh 3 (non-structured)	200	333	21	1.80	55.88
Mesh 4 (non-structured)	1,451	2,612	153	2.47	319.38

is significantly reduced (even failed) when we train using the seq2seq or seq2seq with attention models compared with the Ptr-Net model.

2) TEST MESHES

Figure 11 shows the four 2D triangular meshes that are tested. All four test meshes are initially tangled meshes, i.e., include inverted elements and must perfectly eliminate inverted elements before numerical simulations are conducted.

Mesh 1 is a simple non-convex mesh with four elements, mesh 2 is structured, and meshes 3 and 4 are non-structured. Mesh 3 is a locally refined mesh, with more elements around the circle. Table 1 summarizes test mesh properties and mesh quality statistics. We use the IMR quality metric in (3) to measure element quality, where the ideal element has value one, smaller value indicates better element quality, and inverted elements have negative values.

3) EVALUATION METRICS

Several common metrics are used to evaluate output mesh quality.

- The number of inverted elements in the output mesh.
- Average element quality, i.e., the sum of all element quality in the mesh divided by the total number of elements.
- Worst element quality, i.e., the maximum element quality value for all elements.
- The distribution of element quality values.

The worst element quality is important since a few poor-quality elements can severely affect numerical solution accuracy and efficiency [12].

4) COMPARISON

For validation, we compare the proposed framework with two mesh optimization algorithms. First method is a two-step (TS) method. The TS method first uses optimization-based mesh untangling (OMU) [12], [15] to eliminate all inverted elements, and then optimization-based mesh smoothing to improving mesh quality (see Section III). OMU is a simple and popular mesh untangling method, implemented in Mesquite software. The OMU β value is a user-defined parameter greater than zero. Preliminary experiments show that the selected β has little impact on OMU performance. Therefore, we set $\beta = 0.001$ for all experiments.

The second step of the TS method is not performed when the OMU (first step) fails to untangle the mesh. The nonlinear steepest descent method is unable to return valid gradient and function values for optimization-based mesh smoothing (second step of the TS method) for the initial patch when the mesh is tangled. We use the L^2 norm for the TS method to incorporate element qualities, and nonlinear steepest descent method to find the local optimum

Another optimization-based method we compare is a derivative-free (DF) mesh optimization method [12]. Different from the TS method, it does not require to compute the function derivatives or a Hessian, but only uses function evaluations to find the local optimal point [12]. Readers refer to [12] for more details on the DF method. The DF method finds the local optimal point by improving the worst quality element in the mesh.

We focus on improving the worst quality element in the mesh and employ the DF method to find the local optimal point. The objective function we minimize can be denoted

TABLE 2. Number of inverted elements in initial and output meshes.

Mesh name	Initial mesh	Output mesh (TS method)	Output mesh (DF method)	**Output mesh (proposed framework)**
Mesh 1	1	0	0	**0**
Mesh 2	12	0	0	**0**
Mesh 3	21	2	0	**0**
Mesh 4	153	1	4	**0**

follows:

$$\max_{i \in E} (q_i) , \tag{12}$$

where E is an element set in the patch. Since the optimization problem is a non-smooth function due to the maximum function, the DF method employs a downhill simplex method to solve the optimization problem. The downhill simplex method requires a user-defined parameter, simplex diameter, to set. It affects the initial simplex size around the free vertex. Similar to [12], a simplex diameter in the downhill simplex method is set to 0.1 for all experiments. We employ the untangling quality metric, $q_{i,UNT}$, in Section III and use the same β value as the OMU method.

B. RESULTS

1) INVERTED ELEMENTS

Table 2 shows the number of inverted elements in initial and output meshes. The proposed framework successfully eliminates all inverted elements for all test meshes, whereas both the TS and DF methods fail to untangle inverted elements for unstructured meshes. Although OMU decreases the number of inverted elements, output meshes 3 and 4 retained 2 and 1 inverted elements, respectively. The DF method successfully eliminates meshes 1-3, but fails to eliminate inverted elements for the output mesh 4.

2) OUTPUT MESH QUALITY

Figure 12 shows output meshes using the TS [12], [15] and DF [29] methods and the proposed framework, respectively. OMU (the first step of TS) fails to untangle inverted elements for meshes 3 and 4, hence the second step is not performed. Mesh 1 is a simple non-convex mesh, and both TS and the proposed framework provide nearly optimal output meshes, although the proposed framework produces a slightly better output mesh. The output mesh using the DF method produces more elements with high aspect ratio compared with other two methods. One can apply Laplacian smoothing for this simple mesh, such that the free vertex moved the average of its neighboring vertex positions. However, this leads to a tangled mesh for the non-convex mesh, since the new free vertex position is located outside the mesh.

Mesh 2 is structured, and widely used for partial differential equation (PDE)-based simulations. The TS and proposed framework both produce high-quality output meshes with most elements being nearly equilateral. The DF method shows the worst performance in terms of the output element quality among the three compared methods.

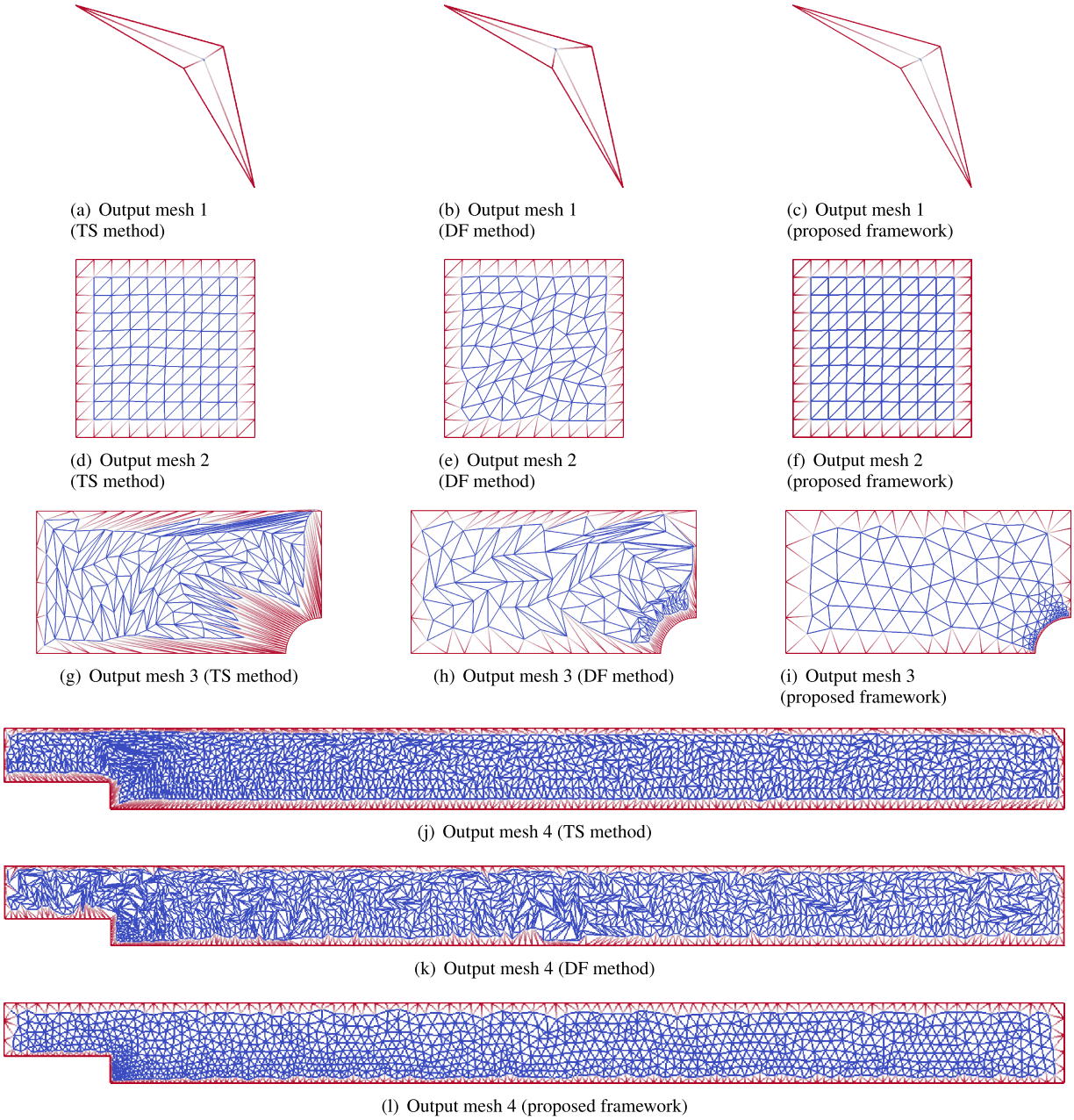
Meshes 3 and 4 are unstructured meshes. For these two meshes, the output mesh quality using the proposed framework significantly outperforms the TS and DF methods. OMU method fails to untangle inverted elements and the resultant TS output meshes include skinny or skewed elements with poor element qualities.

Table 3 shows output mesh average and worst element qualities. Average element qualities using the proposed framework improve by up to 15.4%, 8.7%, 85.9%, and 44.7% for meshes 1, 2, 3, and 4, respectively, compared with the optimization-based methods. Overall output element quality for mesh 1 is worse than other meshes due to its geometric constrains and non-convexity. Similar results are exhibited for the worst element qualities. The worst output mesh element qualities using the proposed framework improve by 34.9%, 52.6%, 97.7%, and 80.0%, for meshes 1, 2, 3, and 4, respectively, compared with the optimization-based methods. Considerable higher mesh quality improvement is achieved for unstructured meshes.

Figure 13 shows the output mesh quality distributions and the probability distribution function. For all test meshes, the proposed framework produces output meshes with better element qualities compared with optimization-based methods. Due to the concavity, the element qualities for output mesh 1 are worse than other output meshes. The proposed framework produces output meshes, whose element qualities close to ideal elements for output meshes 2, 3, and 4. In general, the TS method shows better output element qualities compared with the DF method, but for the output mesh 3, the DF method outperforms the TS method in terms of the output element qualities. Both the TS and DF methods fail to eliminate inverted elements for output mesh 4.

VII. DISCUSSIONS AND LIMITATIONS

The proposed framework uses the optimization-based mesh smoothing using Mesquite software to find and learn the best free vertex location during the training phase since the optimization-based mesh smoothing can produce high-quality output meshes for most initially valid meshes. But our test meshes are all initially tangled meshes with poor element qualities. If the initial mesh is tangled with poor element qualities, we observe that the optimization-based untangling (also, optimization-based mesh untangling and smoothing) methods often fail to eliminate inverted elements in the mesh. Therefore, we employ a deep learning-based approach to perform simultaneous mesh untangling and smoothing for initially tangled meshes with poor element qualities.



(a) Output mesh 1 (TS method)
(b) Output mesh 1 (DF method)
(c) Output mesh 1 (proposed framework)
(d) Output mesh 2 (TS method)
(e) Output mesh 2 (DF method)
(f) Output mesh 2 (proposed framework)
(g) Output mesh 3 (TS method)
(h) Output mesh 3 (DF method)
(i) Output mesh 3 (proposed framework)
(j) Output mesh 4 (TS method)
(k) Output mesh 4 (DF method)
(l) Output mesh 4 (proposed framework)

FIGURE 12. **Output meshes using the TS method, DF method and the proposed framework, respectively.**

TABLE 3. **Output mesh quality statistics.**

Mesh name	Mesh quality	Initial mesh	Output meshes			
			First step of the TS method	Second step of the TS method	DF method	**Proposed framework**
Mesh 1	Average	5.62	93.37	4.15	4.86	**4.11**
	Worst	14.30	187.31	4.62	7.09	**4.61**
Mesh 2	Average	1.86	1.15	1.15	1.26	**1.15**
	Worst	34.20	1.30	1.17	2.47	**1.17**
Mesh 3	Average	1.80	7.61	7.61	2.96	**1.07**
	Worst	55.88	67.48	67.48	33.43	**1.50**
Mesh 4	Average	2.47	1.59	1.59	1.90	**1.05**
	Worst	319.38	8.45	8.45	13.45	**2.69**

During the training phase, the optimal free vertex index and the boundary vertex indices are placed in the even-numbered and odd-numbered indices, respectively, as the Ptr-Net output sequence. We perform additional experiments by

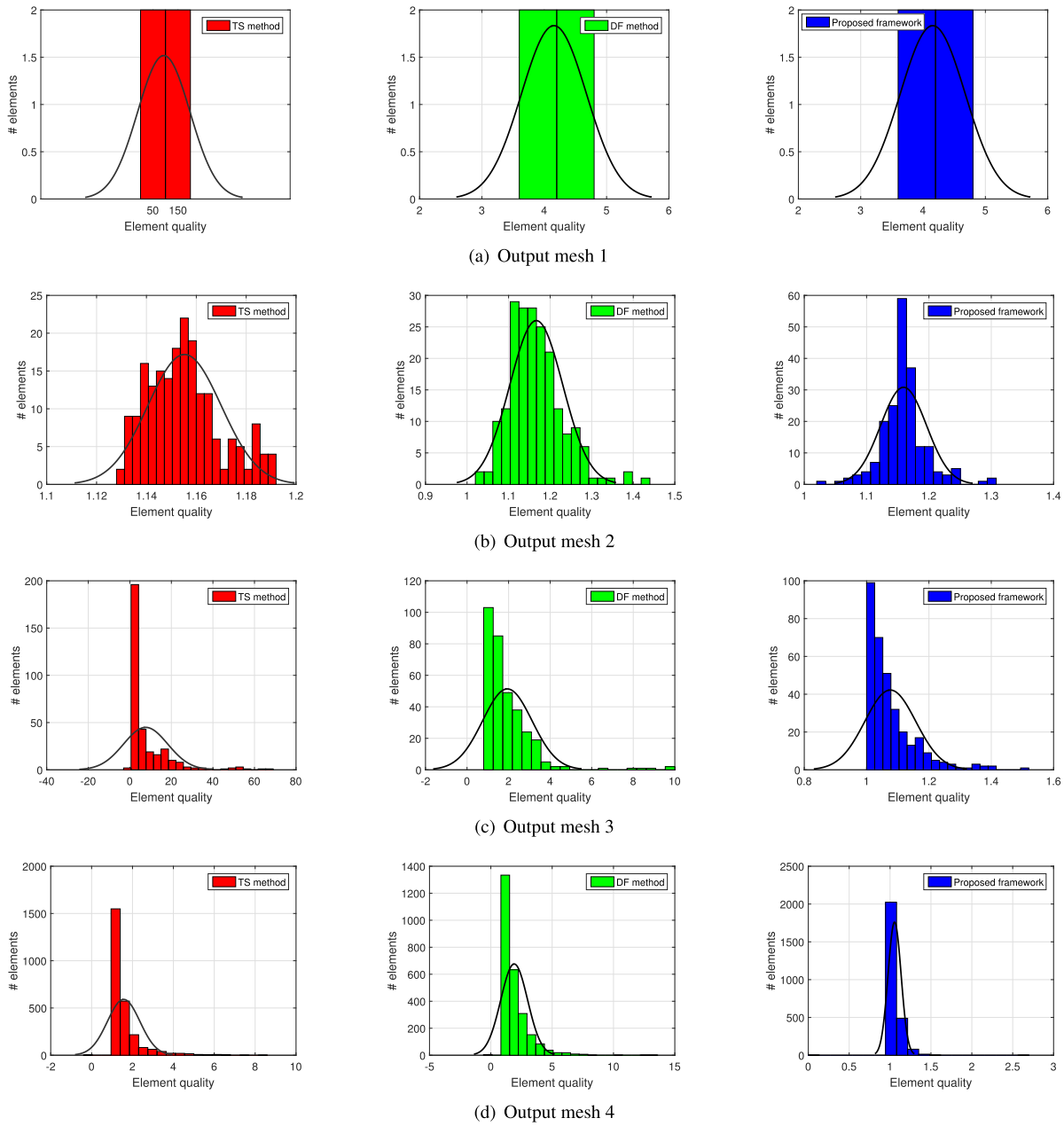


FIGURE 13. Mesh quality distributions of output meshes. The IMR quality metric is used to measure the element quality. A smaller value indicates a better element quality and the inverted elements have negative values. The ideal element has a value of one.

changing the order of the output sequence such that the created output sequence during the training phase is $\{\Rightarrow, (I_O, I_b^1), (I_O, I_b^2), \cdots, (I_O, I_b^N), \Leftarrow\}$, where N is the number of boundary vertices in the polygon. Then, during the prediction phase, we choose the most frequent index in the odd-numbered locations from the output sequence as the best free vertex index except the beginning/end of sequence tokens. We observe similar results using this ordering.

The proposed framework learns and predicts the best free vertex candidate among various vertex candidates. Therefore, output mesh elements are close, but not identical, to equilateral triangles. The user can provide more vertex candidates during prediction to improve output mesh quality, but this may also increase prediction time. Another possible strategy to improve output element quality is to repeat the prediction phase until output mesh quality converges. In particular, free vertex movement can be allowed only if the new vertex candidates improve output mesh quality.

For most PDE-based applications, skinny or skewed elements produce large errors compared with equilateral triangles. In this paper, we assume that the ideal element is an equilateral triangle and use the IMR quality metric to find the

optimal free vertex location during the training phase. But other mesh quality metrics can be used to train the Ptr-Net. We train and test another mesh quality metric, radius ratio quality metric, to validate the results. It is defined as the ratio of the inradius to the circumradius of a triangle [27]. It prefers equilateral triangles to triangles with high aspect ratios [27]. We observe similar results with those in Tables 2 and 3.

The main limitations for the proposed framework are that it requires high-quality labeled data for training and finds approximate rather than optimal solutions. Creating high-quality labeled data can be somewhat expensive. Moreover, the proposed framework requires additional training time compared with optimization-based methods.

VIII. CONCLUSION

This paper proposed a Ptr-Net based framework for simultaneous mesh untangling and smoothing. The proposed framework is the first data-driven approach to perform simultaneous mesh untangling and smoothing, and offers advantages by predicting competitive approximate solutions without requiring to solve complex numerical optimization problems. Numerical experiments verified that the proposed framework successfully eliminates all inverted elements and improved average and worst element quality up to 85.9% and 97.8%, respectively, compared with the optimization-based methods.

The proposed framework predicts competitive approximate solutions from random vertex candidates. Future studies will investigate how to improve mesh quality by generating better vertex candidates. We observe that both input and output sequence order presented to the Ptr-Net impacts overall performance. We plan to investigate better input/output sequence ordering for improving the performance. Recently, polygonal and polyhedral meshes received lots of attention, since they require a fewer number of elements and also well-suited for complex geometries. Another direction for future research is to extend the proposed framework for other element types such as polygonal and polyhedral meshes [13], [38].

REFERENCES

[1] J. Shewchuk, "What is a good linear element? Interpolation, conditioning, and quality measures," in *Proc. Int. Meshing Roundtable*, New York, NY, USA: Ithaca, 2002, pp. 115–126.

[2] M. Berzins, "Mesh quality—Geometry, error estimates, or both," in *Proc. Int. Meshing Roundtable*, Dearborn, MI, USA, 1998, pp. 229–237.

[3] J. Kim, S. P. Sastry, and S. M. Shontz, "A numerical investigation on the interplay amongst geometry, meshes, and linear algebra in the finite element solution of elliptic PDEs," *Eng. Comput.*, vol. 28, no. 4, pp. 431–450, Oct. 2012.

[4] J. Kim, T. Panitanarak, and S. M. Shontz, "A multiobjective mesh optimization framework for mesh quality improvement and mesh untangling," *Int. J. Numer. Methods Eng.*, vol. 95, no. 13, pp. 1113–1114, Sep. 2013.

[5] M. Berzins, "Solution-based mesh quality for triangular and tetrahedral meshes," in *Proc. Int. Meshing Roundtable*, New York, NY, USA: Ithaca, 1997, pp. 427–436.

[6] L. Freitag, "On combining Laplacian and optimization-based mesh smoothing techniques," *AMD Trends Unstructured Mesh Gener.*, vol. 220, pp. 37–43, Jul. 1997.

[7] L. Freitag and P. Plassmann, "An efficient parallel algorithm for mesh smoothing," in *Proc. Int. Meshing Roundtable*, Albuquerque, NM, USA, Dec. 1995, pp. 47–58.

[8] L. Freitag Diachin, P. Knupp, T. Munson, and S. Shontz, "A comparison of two optimization methods for mesh quality improvement," *Eng. Comput.*, vol. 22, no. 2, pp. 61–74, Sep. 2006.

[9] L. A. Freitag and P. Plassmann, "Local optimization-based simplicial mesh untangling and improvement," *Int. J. Numer. Methods Eng.*, vol. 49, nos. 1–2, pp. 109–125, Sep. 2000.

[10] S. P. Sastry and S. M. Shontz, "Performance characterization of nonlinear optimization methods for mesh quality improvement," *Eng. Comput.*, vol. 28, no. 3, pp. 269–286, Jul. 2012.

[11] M. Brewer, L. Freitag Diachin, P. Knupp, T. Leurent, and D. Melander, "The mesquite mesh quality improvement toolkit," in *Proc. Int. Meshing Roundtable*, Santa Fe, NM, USA, 2003, pp. 239–250.

[12] J. Kim, M. Shin, and W. Kang, "A derivative-free mesh optimization algorithm for mesh quality improvement and untangling," *Math. Problems Eng.*, vol. 2015, pp. 1–10, Mar. 2015.

[13] R. Garimella, J. Kim, and M. Berndt, "Polyhedral mesh generation and optimization for non-manifold domains," in *Proc. Int. Meshing Roundtable*, Orlando, FL, USA, 2013, pp. 313–330.

[14] J. Danczyk and K. Suresh, "Finite element analysis over tangled simplicial meshes: Theory and implementation," *Finite Elements Anal. Des.*, vols. 70–71, pp. 57–67, Aug. 2013.

[15] P. M. Knupp, "Hexahedral and tetrahedral mesh untangling," *Eng. Comput.*, vol. 17, no. 3, pp. 261–268, Oct. 2001.

[16] S. P. Sastry, S. M. Shontz, and S. A. Vavasis, "A log-barrier method for mesh quality improvement and untangling," *Eng. Comput.*, vol. 30, no. 3, pp. 315–329, Jul. 2014.

[17] P. Knupp, "Updating meshes on deforming domains: An application of the target-matrix paradigm," *Commun. Numer. Methods Eng.*, vol. 24, no. 6, pp. 467–476, 2007.

[18] D. Vartziotis and B. Himpel, "Laplacian smoothing revisited," 2014, *arXiv:1406.4333*. [Online]. Available: http://arxiv.org/abs/1406.4333

[19] J. W. Franks and P. M. Knupp, "A new strategy for untangling 2D meshes via node-movement," CSRI Summer, Sandia Nat. Lab., Albuquerque, NM, USA, Tech. Rep., 2010.

[20] J. W. Franks and P. M. Knupp, "The maximum value method: A node-movement strategy for simultaneous mesh untangling and improvement," CSRI Summer, Sandia Nat. Lab., Albuquerque, NM, USA, Tech. Rep., 2011.

[21] D. Benitez, E. Rodriguez, J. Escobar, and R. Montenegro, "Performance evaluation of a parallel algorithm for simultaneous untangling and smoothing of tetrahedral meshes," in *Proc. Int. Meshing Roundtable,* Orlando, FL, USA, 2013, pp. 579–598.

[22] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2015, pp. 2692–2700.

[23] A. Mottini and R. Acuna-Agost, "Deep choice model using pointer networks for airline itinerary prediction," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining KDD*, Halifax, NS, Canada, 2017, pp. 1575–1583.

[24] S. Gu and T. Hao, "A pointer network based deep learning algorithm for 0–1 knapsack problem," in *Proc. 10th Int. Conf. Adv. Comput. Intell. (ICACI)*, Xiamen, China, Mar. 2018, pp. 473–477.

[25] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," 2015, *arXiv:1511.06391*. [Online]. Available: http://arxiv.org/abs/1511.06391

[26] T. Munson, "Mesh shape-quality optimization using the inverse mean-ratio metric," *Math. Program.*, vol. 110, no. 3, pp. 561–590, May 2007.

[27] J. Park and S.M. Shontz, "Two derivative-free optimization algorithms for mesh quality improvement," in *Proc. Int. Conf. Comput. Sci.*, Amsterdam, The Netherlands, 2010, pp. 387–396.

[28] S. Bhowmick and S. M. Shontz, "Towards high-quality, untangled meshes via a force-directed graph embedding approach," in *Proc. Int. Conf. Comput. Sci.* Amsterdam, The Netherlands, 2010, pp. 357–366.

[29] D. Vartziotis and M. Papadrakakis, "Improved GETMe by adaptive mesh smoothing," *Comput. Assist. Methods Eng. Sci.*, vol. 20, no. 1, pp. 55–71, Jan. 2017.

[30] T. Le-Duc, Q.-H. Nguyen, and H. Nguyen-Xuan, "Balancing composite motion optimization," *Inf. Sci.*, vol. 520, pp. 250–270, May 2020.

[31] J. Choi, H. Kim, S. P. Sastry, and J. Kim, "A deviation-based dynamic vertex reordering technique for 2D mesh quality improvement," *Symmetry*, vol. 11, no. 7, p. 895, 2019.

[32] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Montreal, Quebec, Canada, 2014, pp. 3104–3112.
[33] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[34] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*. [Online]. Available: http://arxiv.org/abs/1412.3555
[35] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014, *arXiv:1409.0473*. [Online]. Available: http://arxiv.org/abs/1409.0473
[36] J. Weston, S. Chopra, and A. Bordes, "Memory networks," 2014, *arXiv:1410.3916*. [Online]. Available: http://arxiv.org/abs/1410.3916
[37] T. Guo, C. Han, S. Tang, and M. Ding, "Solving combinatorial problems with machine learning methods," *Nonlinear Combinatorial Optimization*. Cham, Switzerland: Springer, 2019, pp. 207–229.
[38] H. Nguyen-Xuan, K. N. Chau, and K. N. Chau, "Polytopal composite finite elements," *Comput. Methods Appl. Mech. Eng.*, vol. 355, pp. 405–437, Oct. 2019.

JIBUM KIM received the B.S. and M.S. degrees in electrical engineering from Yonsei University, Seoul, South Korea, in 2003 and 2005, respectively, and the Ph.D. degree in computer science and engineering from Pennsylvania State University, University Park, PA, USA, in 2012. He was a Postdoctoral Fellow with the Los Alamos National Laboratory (Group T-5), in 2013. He is currently an Associate Professor with the Department of Computer Science and Engineering, Incheon National University, South Korea. His current research interests include computational science, artificial intelligence, and high-performance computing.

JUNHYEOK CHOI received the B.S. and M.S. degrees from the Department of Computer Science and Engineering, Incheon National University, South Korea, in 2017 and 2019, respectively. He is currently a Research Engineer with SK Hynix Inc. His current research interests include deep learning and high-performance computing.

WOOCHUL KANG (Member, IEEE) received the Ph.D. degree in computer science from the University of Virginia, in 2009. He was a Senior Researcher with the Electronics and Telecommunications Research Institute, South Korea, from 2000 to 2004, and from 2009 to 2012. He was a Postdoctoral Research Associate with the University of Illinois at Urbana-Champaign, USA, from 2012 to 2013. He is currently an Associate Professor with the Department of Embedded Systems Engineering, Incheon National University, Incheon, South Korea. His research interests include real-time systems, distributed middleware, feedback control of computing systems, and deep learning for embedded systems.