




Frame Field Prediction For Quadrilateral Domain Partition

Tobias Rentschler* , Alexander Tismer , Stefan Riedelbauch .

Institute of Fluid Mechanics and Hydraulic Machinery
University of Stuttgart
DE, Germany

*tobias.rentschler@ihs.uni-stuttgart.de

DATE

Abstract—The generation of high-quality quadrilateral meshes is an essential requirement for a variety of applications, including Computational Fluid Dynamics (CFD). Conventional approaches generally compromise between speed and quality, with the computational demands increasing in response to higher quality requirements. Cross-field-based mesh generation has the potential to produce high-quality block-structured meshes. However, this approach relies on the solution of Partial Differential Equations (PDEs), which can be computationally intensive, particularly for large or high-resolution domains. This study presents a Graph Neural Network (GNN) capable of learning and approximating these PDEs, allowing the efficient partitioning of geometric domains into quadrilateral subregions. The primary benefit of this method is its capacity to partition geometric domains into quadrilateral subregions with good scalability, while eliminating the need to solve expensive PDEs. The GNN predicts a boundary-aligned frame field that serves as the foundation for the domain partitioning. Boundary conditions are defined based on normal and tangent vectors at the domain boundaries. They are propagated smoothly across the interior using the GNN, eliminating the need for computationally expensive PDEs. In a subsequent post-processing step, singularities within the frame field are identified, and streamlines originating from these singularities are computed. These streamlines are strategically aligned with the geometry of the domain, leading to the decomposition of the domain into quadrilateral regions. The resulting regions are well-suited for fast algebraic meshing techniques, such as bilinear transfinite interpolation, ensuring the generation of high-quality meshes. This study utilizes a simple 2-dimensional NACA profile as a test case to train the GNN.

Keywords-component—Domain Partitioning, Graph Neural Network, Frame Field, Quadrilateral Mesh, Grid Generation

I. INTRODUCTION

High-quality meshes are essential because they influence numerical stability and solution accuracy [8], therefore min-

imize numerical errors, and enable faster convergence of simulations compared to low-quality meshes.

In CFD simulations, which require high precision, the meshes are often manually created by experts, not only to ensure the quality of the mesh, but also to capture physically relevant details. However, the manual generation of high-quality and high-resolution meshes, which can easily exceed one million faces, is a time-consuming process, even for experienced professionals. In optimization processes, where a large number of designs are generated, manually created meshes become impractical. Consequently, an automated approach is essential. However, existing automated methods often present a trade-off. They are either fast but lack the desired quality, or they produce high-quality meshes at the expense of significant computational effort. This often results in the mesh generation process demanding more computational effort than the subsequent CFD simulation, thus potentially causing a significant bottleneck in the overall workflow.

The development of quadrilateral meshing techniques has been accompanied by comprehensive studies that categorize and evaluate these methods for various applications [1]. Cross-field-based approaches are widely adopted for their effectiveness in generating high-quality block-structured meshes among the various methods. These approaches utilize cross fields to partition a domain into quadrilateral regions, suitable for quadrilateral meshing. However, they rely on solving (PDEs), which is computationally expensive, especially for large or high-resolution domains. Recent studies have explored machine learning techniques as alternatives to address this computational bottleneck. For instance, [10] proposed a Convolutional Neural Network (CNN)-based framework for quadrilateral mesh generation. While Convolutional Neural

Identify applicable sponsor/s here. (*sponsors*)

Networks (CNNs) are effective at predicting patterns in structured grid data, they struggle with the complex and non-uniform mesh geometries, which restricts their functionality to basic two-dimensional test cases. Graph Neural Networks (GNNs) on the other hand, are uniquely suited for processing the relational and topological information inherent in mesh structures [7]. This study builds on these advancements by replacing CNNs with GNNs to predict frame fields, which form the basis for quadrilateral meshing. Using GNNs significantly reduces computational costs while offering a more flexible and robust framework for mesh generation in CFD applications. Additionally, recent advancements in Artificial Intelligence (AI)-driven, transformer-based mesh generation methods [3] have demonstrated the capability to produce high-quality meshes for three-dimensional surfaces. These methods excel in applications such as computer animation, where meshes with relatively low face counts are sufficient. However, due to their architectural design, transformer-based approaches are constrained by a maximum sequence length, which limits the size of the generated meshes [9]. This constraint poses challenges for CFD applications, where high-resolution meshes with significantly larger element counts are required. This limitation makes them unsuitable for engineering applications like CFD, which require volumetric meshing with significantly higher face counts. A promising solution to this limitation is subdividing the entire domain into smaller, more manageable regions, enabling these methods to process complex geometries while maintaining their high-quality output. This paper proposes a novel GNN-based approach for domain partitioning, which efficiently divides a geometric domain into quadrilateral subregions. By combining the strengths of GNNs and traditional cross-field methods, the framework enables scalable, high-quality meshing for CFD applications while also addressing the limitations of existing transformer-based techniques.

II. OVERALL APPROACH

In the initial phase of the process, the geometric domain is meshed using a conventional triangulation algorithm. In this research, the geometric domain is a simple flow field around a NACA airfoil. At each boundary node of the triangulated mesh, normal and tangent vectors are computed, forming a set of four orthogonal vectors. This set of vectors is referred to as a cross. Cross fields are defined as smoothly varying fields of such crosses over the whole domain. The objective of this study is to generate such a cross field by propagating the crosses from the boundary to the interior. Conventional approaches to this problem rely on the solution of PDEs. In this research, a GNN was trained to approximate, and consequently replace these PDEs. Rather than propagating four orthogonal vectors per node, the crosses are mapped to a single vector, reducing the problem's complexity. The propagation of the boundary vectors generates a frame field. Subsequently, this frame field is remapped to a cross field, assigning four orthogonal vectors to each node. It is important to note that there are locations within the domain where

the assignment of four smooth orthogonal vectors is not possible. These locations are designated as singularities and require special treatment. In particular, at each singularity an odd number of separatrices, which are trajectories of the frame field that pass through the singularity, are computed [6]. Following the trajectory of these separatrices generates streamlines that partition the domain into quadrilateral regions, thereby eliminating the presence of singularities and ensuring a smooth cross field.

In this section, the proposed method is described and illustrated in detail in a sequence of steps.

A. Triangulated Mesh Generation

The process is initiated by generating a triangulated mesh, displayed in figure 1, which serves as a computational tool rather than forming the geometric foundation of the final mesh. In the numerical approach, this mesh is used to compute the frame field, while in the GNN approach, it facilitates graph convolution for the prediction of the frame field. This research utilized the open-source meshing tool Gmsh to generate a Delaunay Triangulation [2]. Notably, as highlighted in [5], the quality of the triangulated mesh has minimal impact on the resulting domain partition.

B. Cross Computing

At each boundary node, a cross is computed, comprising four orthogonal vectors formed by two pairs of tangent and normal vectors. At zero-continuity boundary nodes, where no exact normal and tangent can be defined, the mean direction is used as an approximation, as illustrated in figure 1. These vectors define directionality based on the local boundary geometry.

C. Cross to Vector Mapping

These boundary crosses are mapped to a single representative vector per node, with non-boundary nodes initialized to zero. Using only normal or tangent vectors at boundary nodes as representative vectors of the crosses, fails to achieve a smooth frame field if there are discontinuous boundaries.

This issue is evident in our test case and is a common challenge in most fields. In our test case, 0-continuity is observed at specific points within the domain, particularly at corner nodes where two boundary lines intersect at an angle of $\frac{\pi}{2}$ [rad], as well as at the trailing edge of the blade. The discontinuities, present at these points, affect the smooth propagation of cross-field vectors throughout the domain. Identifying and handling these discontinuities is essential to ensure smooth transitions and maintain the quality of the resulting frame field. Instead, we leverage the concept of directionality from [5] by mapping the computed crosses at each boundary node to a single representative vector, which ensures a smooth cross field even at sharp boundary corners, thus aligning better with the structure of a quadrilateral mesh.

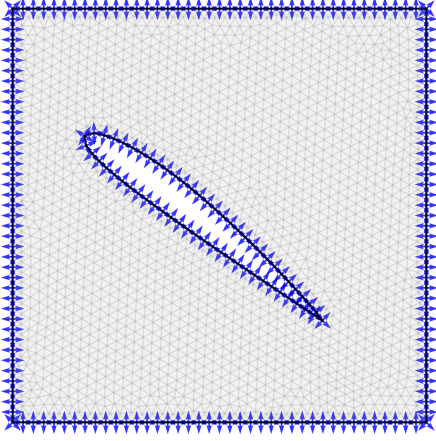


Figure. 1. Illustration of the first and second step. An initial triangulated mesh of the test case, consisting of a two-dimensional uniform flow channel surrounding a NACA profile is generated. Subsequently, the cross vectors, composed of tangent and normal vectors, are computed at the boundary nodes of the mesh.

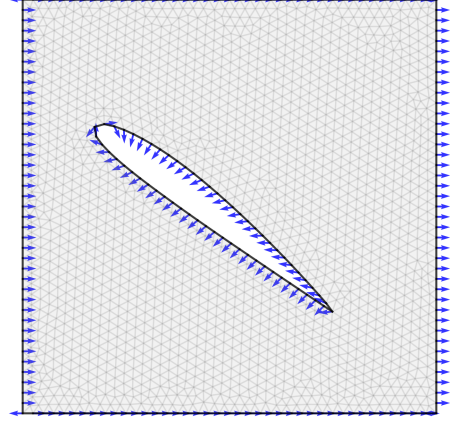


Figure. 2. Illustration of the mapped crosses to a single representative vector.

For non-boundary nodes, this vector is initialized to zero. This mapping is essential to achieve a compact representation of directional information across the domain. Each cross vector is defined as:

$$C_\phi = \vec{v}_k = \left(\cos \left(\phi + \frac{k\pi}{2} \right), \sin \left(\phi + \frac{k\pi}{2} \right) \right), \quad k = 0, 1, 2 \quad (1)$$

The vectors of a cross v_k are mapped to a single representative vector u using a reference axis. In this case, the x -axis is used as the reference. Notably, the choice of the reference axis does not affect the resulting frame field. The angles between the cross vectors and the reference axis are calculated, and the smallest angle is selected. The computed representative vectors have angles constrained to the range $[0, \frac{\pi}{2}]$ [rad]. However, since this range is discontinuous, it is unsuitable for smooth field interpolation. Consequently, the angles are multiplied by four, transforming the range to $[0, 2\pi]$ [rad], which ensures continuity and smoothness. Formally, this mapping is expressed as:

$$\vec{u} = 4 \cdot \min \theta(\vec{v}_k), \quad k = 0, 1, 2 \quad (2)$$

D. Approximating the Partial Differential Equation (PDE) using a GNN.

The graph representing the mesh consists of two components: the node feature tensor and the connectivity tensor. The node feature tensor stores information about each mesh node, including its spatial coordinates and details about its representative vector. If the node is located on the domain boundary, the graph node contains the precomputed x and y values of the representative vector. If the node is in the interior of the domain, no additional information about the representative vector is available, and the node is initialized with $x = 0$ and $y = 0$. The connectivity tensor stores the start and end node indices for each edge in the graph. Since the

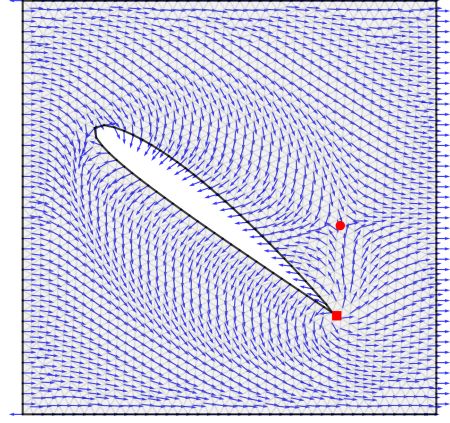


Figure. 3. Visualization of the propagated frame field, with singularities highlighted in red. The five-valent singularity is shown as a red circle, while the three-valent singularity is marked with a red square, as detailed in section II-E.

graph is undirected, each edge is included twice, once in each direction. The graph is then passed to the GNN to predict the frame field of the entire domain. A detailed description of the GNN is provided in section III.

E. Singularities Detection

The identification of singularities is the next step in the decomposition of a domain into quadrilateral regions. In this step, singularities are detected within a frame field. The detection relies on the computation of the Poincaré index for each triangle in the mesh. This index quantifies the rotation of the frame vectors around the perimeter of the triangle, allowing for the identification of singularities and their type. Based on the index value, it is determined whether three or five streamlines must converge at the singularity, which plays a key role in guiding the decomposition of the domain

into quadrilateral subregions. To compute the Poincaré index of a vector field defined over a triangulated mesh, a single triangle $T = (P_1, P_2, P_3)$ is considered, with the vertices $P(P_1, P_2, P_3)$ oriented counterclockwise. For each vertex P_i of the triangle, where $i \in 1, 2, 3$, the angle between the vector at P_i and the positive X-axis is denoted as θ_i . The angular change of the vector is computed as it moves from one vertex to the next to capture the rotation of the vector field along the edges of the triangle. For any two vertices P_i and P_j (with $i, j \in 1, 2, 3$ and $i \neq j$), the angular change along the edge $E = (P_i, P_j)$ is defined as:

$$\Delta\theta_j^i = ((\theta_j - \theta_i + \pi) \bmod 2\pi) - \pi \quad (3)$$

Here, $\Delta\theta_j^i$ represents the angular difference between the direction of the vector at (P_j) and the vector (P_i) within the range $[-\pi, \pi]$, representing the smallest signed rotation between the two vectors [5]. The value (I_T) , equation 4, represents the total winding or rotation of the vector field as it moves along the boundary of the triangle. If (I_T) is an integer, it represents the number of full rotations the vector field completes in its trajectory along the boundary of the triangle. It is worth noting that if $(\theta_i = -\theta_j)$ for any two consecutive vertices, the interpolation of the angle along that edge is discontinuous, and special care is needed to handle this case.

$$I_T = \frac{1}{2\pi} (\Delta\theta_2^1 + \Delta\theta_3^2 + \Delta\theta_1^3) \quad (4)$$

If the Poincaré index (I_T) is +1, it corresponds to a 3-valent singularity, meaning that three trajectories of the frame field pass through this singularity. Conversely, an index of -1 corresponds to a 5-valent singularity [5], as shown in Figures 3 and 5.

F. Mapping Frame Field to Cross Field

To generate a cross field, the vectors of the frame field are transformed into crosses. This process effectively reverses the approach described in Step 3.

G. Streamline Initialization

The generation of streamlines, to divide the domain into quadrilateral regions, is initiated by computing the trajectories of the cross field that passes through the singularities, known as separatrices. The following process is applied to each triangle where a singularity is detected as described in the subsection II-E. The process begins by initializing points P located along the edges of a triangle defined by the vertices (P_1, P_2, P_3) to generate the separatrices. Each of these vertices (P_1, P_2, P_3) is associated with a corresponding cross vector $\vec{u}_1, \vec{u}_2, \vec{u}_3$. The direction of the vector $\vec{S}\vec{P}$, which points from the singularity S to the point P , must be aligned with the interpolated vector of the cross field at P [6]. This approach is illustrated in Figure 5.

The point P on the edge $E = (P_i, P_j)$ is parameterized as:

$$P = (1 - t)P_i + tP_j \quad \text{where } t \in [0, 1] \quad (5)$$

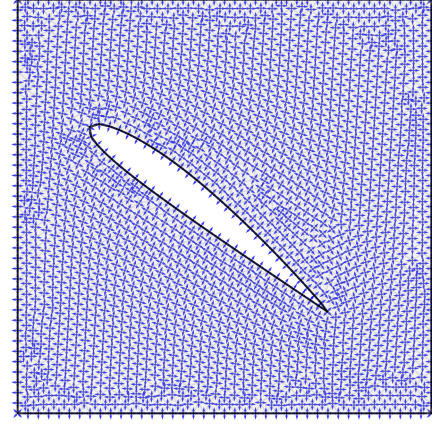


Figure 4. Illustration of the resulting cross field.

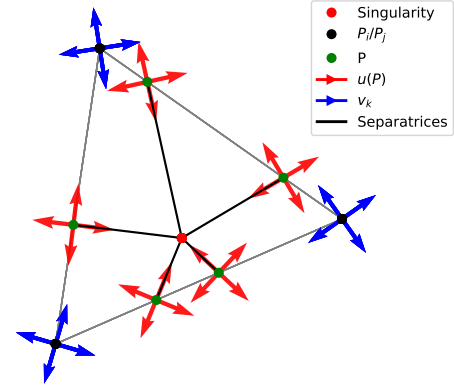


Figure 5. Illustration of the process to compute the separatrices of the singularity. The red dot inside the triangle represents the singularity. The cross-field vectors are shown as blue arrows at each vertex of the triangle, while the interpolated cross vectors at points along the triangle's edges, where the direction of the cross aligns with the line connecting the point to the singularity, are represented by red arrows.

To compute the interpolated vector $\vec{u}(P)$ at P , the frame field is interpolated using the vectors \vec{u}_i and \vec{u}_j at P_i and P_j , respectively. This interpolation is given by:

$$\vec{u}(P) = (1 - t)\vec{u}_1 + t\vec{u}_2 \quad (6)$$

This provides a smooth transition of the vector field along the edge $E = (P_i, P_j)$ as t varies from 0 to 1. However, it is important to note that this interpolated vector belongs to the frame field, not the cross field. While the singularities of the frame field are the same as those of the cross field, the streamlines are different [5]. Since the goal is to generate streamlines of the cross field, the vector $\vec{u}(P)$ must be mapped to a cross vector \vec{c}_k with $k \in [0, 1, 2, 3]$. Each of the four vectors \vec{c}_k is checked to see if one of them aligns with the direction of the vector $\vec{S}\vec{P}$. The alignment is tested by ensuring that the cross product of $\vec{S}\vec{P} \times \vec{c}_k$ is zero. This condition guarantees that the two vectors point in the same direction.

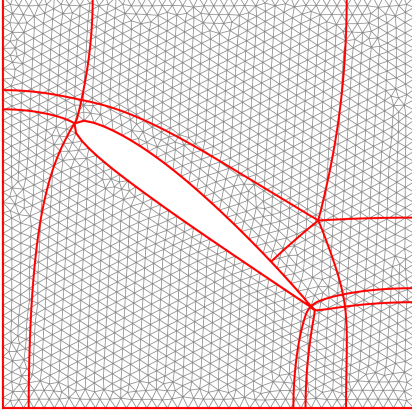


Figure. 6. Illustration of the flow field subdivided into four-sided regions created by streamlines originating from the singularities of the frame field and following the trajectories defined by the cross-field.

H. Streamline Generation

All points P that satisfy the condition $\vec{S}\vec{P} \times \vec{e}_k = 0$ are selected as the second points of the streamlines, with the singularity S serving as the starting point. In addition, each node at the boundary with zero continuity must be considered a starting point. Since the direction at these locations is known, streamline integration can be initiated directly. To accurately trace the streamline trajectory within the cross-field, a second-order Runge-Kutta Heun [4] integration method is employed. The integration of a streamline terminates if either the streamline reaches the boundary of the domain or enters a triangle containing another singularity. In certain cases, multiple streamlines are generated in close proximity and follow almost parallel trajectories. As these streamlines do not provide additional information to the domain partition, they are combined into a single representative streamline. The resulting streamlines subdivide the domain into four-sided regions with no singularities. These regions are well-suited for applying algebraic transfinite interpolation methods, as the absence of singularities ensures smooth and continuous parameterization across the entire sub-domains.

III. GRAPH NEURAL NETWORK

The proposed GNN architecture is designed to predict a frame field within a two-dimensional computational domain. The model employs three graph convolutional layers to propagate information from the boundary to the interior nodes of the graph, enabling the estimation of the frame field vectors at each graph node. The input Graph $G = (N, E)$ to the network is a node feature tensor N where each node is characterized by a feature vector n and the edge connectivity tensor E . Each feature vector consists of the two-dimensional coordinates $[x, y]$ of the node, a binary node label l , and the components of a 2D representative vector $[u_x, u_y]$.

$$n_i = [x_i, y_i, l_i, u_{i,x}, u_{i,y}] \quad (7)$$

The binary label l indicates the type of node, where $l = 0$ identifies boundary nodes and $l = 1$ identifies interior nodes. For boundary nodes, the representative vector $[u_x, u_y]$ is derived from the boundary conditions. For interior nodes, the components $[u_x, u_y]$ are initialized as $[0, 0]$, as no prior information about the frame field is available for these nodes. During the forward propagation, the input features N are successively processed through three convolutional layers. The output of each layer is transformed using the hyperbolic tangent (\tanh) activation function. The final output of the network, designated as $\hat{U} = GCN(G(N, E))$, encompasses two features $[\hat{u}_x, \hat{u}_y]$ for each node.

A. Training

The training procedure for the GNN model is an iterative process with the objective to minimize the difference between the predicted frame field \hat{U} and the ground-truth frame field U . For each epoch, the training data is shuffled, and the model processes each graph, generating node-level frame field predictions. The loss is calculated as the Mean Squared Error (MSE) between the predicted and ground-truth frame fields. Gradients are accumulated over multiple training graphs before an optimizer step is performed. The gradient accumulation process stabilizes updates and emulates a larger batch size.

Validation is performed at regular intervals by evaluating the model on a randomly selected validation graph. In the event that the validation loss is less than the optimal value observed to date, the model weights are saved as a checkpoint. At the conclusion of each epoch, the learning rate is modified via a scheduler to facilitate convergence. The training and validation losses are recorded throughout the process, ensuring that the model exhibiting the best performance is saved for future use.

B. Dataset

The GNN was trained, validated, and tested using a dataset consisting of 10,000 graphs representing the meshes of flow channels around various NACA airfoil profiles. In order to generate a graph for this particular dataset, a random selection is made from one of four 4-digit NACA profiles: 0012, 0015, 2412, or 6412. The selected profile is computed and normalized. Each airfoil is then, based on random coordinates, positioned in a 2D unit flow channel. The initial triangular mesh is generated using the open-source meshing tool Gmsh and converted into a torch graph data structure, where the node coordinates and mesh connectivity are stored. Additionally, binary node and edge labels are assigned to identify nodes and edges located on the domain boundary. An essential property of this graph structure is the frame field. The propagated frame field, which is based on the representative vectors at the boundary, is computed by solving a PDE under a norm constraint to prevent vector magnitudes from vanishing. This approach ensures that the frame field maintains consistent vector magnitudes throughout the domain, facilitating smooth and continuous field propagation. The method follows the formulation proposed in [5].

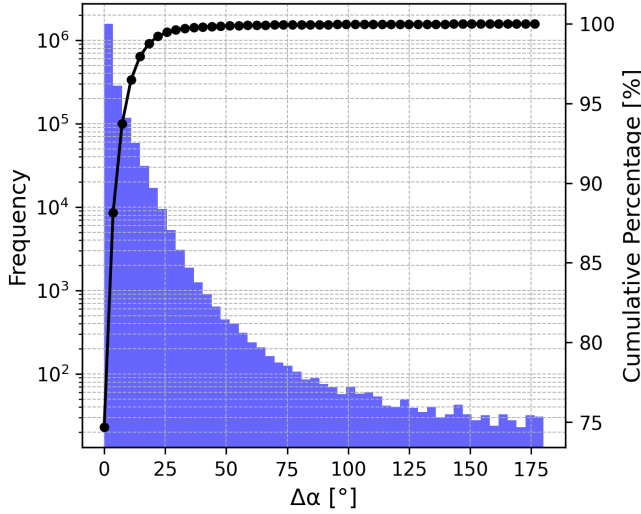


Figure. 7. The histogram illustrates the frequency distribution of angular deviations in the predicted frame field generated by the graph neural network, offering a detailed view of the model performance by showcasing the prevalence of small deviations and the extent of outliers.

C. Validation

Out of the 10,000 meshes, 7,000 were used for training, 2,000 were assigned for model validation during training, and 1,000 were reserved for testing the trained GNN. The evaluation examines the distribution of angular deviations between the predicted frame field vectors and the ground truth across these 1,000 test meshes. Since only the direction of the vectors is considered, magnitude differences are intentionally disregarded, as the primary objective is to assess the alignment accuracy of the predicted frame field. The median angular deviation is 1.53° , indicating that half of the predictions deviate by no more than this amount. Additional statistical insights include a mean deviation of 3.32° , a standard deviation of 5.71° , and a maximum deviation of 179.96° . The histogram (Figure 7) displays the distribution of angular deviations in degrees along the x-axis. The angular deviations are grouped into bins, with a maximum deviation of 180° and 49 bins, each containing angles within a range of approximately 3.67° . The left y-axis, which is logarithmically scaled, provides a representation of the number of occurrences of each specified deviation range, while the right y-axis displays the cumulative percentage of the total number of predicted angles that fall within the corresponding bins. The cumulative analysis of the angular deviations reveals that 74.68% of the predicted vectors deviate by less than 3.68° , showcasing a high degree of alignment with the ground truth. Moreover, 93.7% of the predictions are within the range of 11.2° . Beyond 25° , deviations become significantly less frequent, with fewer than 1% of predictions exceeding this threshold. This observation indicates that a substantial percentage of predictions remain within a reasonably constrained error range, with only a negligible number exhibiting substantial deviations.

IV. SUMMARY AND DISCUSSION

This study introduces a GNN-based approach for quadrilateral domain partitioning, replacing traditional PDE-based methods. The results demonstrate that the predicted frame fields closely align with the ground truth, with the majority of angular deviations remaining small. This confirms the effectiveness of the GNN in capturing the underlying structure of the cross field while significantly reducing computational costs. The method is scalable and adaptable to complex geometries, making it well-suited for CFD applications. However, its accuracy depends on the quality of training data, and singularities and streamlines require additional processing. While merging redundant streamlines improves clarity, it introduces an extra post-processing step. An interesting outlook is the extension of this approach to predict three-dimensional cross fields, where each node is assigned six orthogonal vectors to subdivide a volumetric domain. Compared to PDE-based methods, extending the proposed GNN for this task is relatively straightforward, particularly given the complexity of generating high-quality training data using a PDEs method. Additionally, investigating an AI-based approach for streamline generation, tracing trajectories in the field from given starting points, could further enhance automation and efficiency in the meshing process.

V. FUNDING

This work was funded by the German Research Foundation (DFG) under the special priority program SPP-2353: 501932169.

REFERENCES

- [1] David Bommes, Bruno Levy, Nico Pietroni, Enrico Puppo, Cláudio Silva, and Denis Zorin. Quad-mesh generation and processing: A survey. *Computer Graphics Forum*, 32, 09 2013.
- [2] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre- and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.
- [3] Zekun Hao, David W. Romero, Tsung-Yi Lin, and Ming-Yu Liu. Meshtron: High-fidelity, artist-like 3d mesh generation at scale, 2024.
- [4] Karl Heun. Neue methode zur approximativen integration der differentialgleichungen einer unabhängigen variablen. *Zeitschrift für Mathematik und Physik*, 45:23–38, 1900.
- [5] Nicolas Kowalski, Franck Ledoux, and Pascal Frey. A pde based approach to multidomain partitioning and quadrilateral meshing. In Xi-angmin Jiao and Jean-Christophe Weill, editors, *Proceedings of the 21st International Meshing Roundtable*, pages 137–154, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- [6] Jonathan Palacios and Eugene Zhang. Rotational symmetry field design on surfaces. *ACM Trans. Graph.*, 26(3):55–es, July 2007.
- [7] Ugo Pelissier, Augustin Parret-Fréaud, Felipe Bordeu, and Youssef Mesri. Graph neural networks for mesh generation and adaptation in structural and fluid mechanics. *Mathematics*, 12(18), 2024.
- [8] Ideen Sadrehaghghi. Mesh assessment & quality issues. *CFD Open Series*, 2(40), 2023.
- [9] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.
- [10] Yuxiang Zhou, Xiang Cai, Qingfeng Zhao, Zhoufang Xiao, and Gang Xu. Quadrilateral mesh generation method based on convolutional neural network. *Information*, 14(5), 2023.