The 8th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2022
5-–9 June 2022, Oslo, Norway

MACHINE LEARNING MODEL FOR GAS-LIQUID INTERFACE RECONSTRUCTION IN CFD NUMERICAL SIMULATIONS

T. Nakano[1], M. A. Bucci[1], J-M. Gratien[2], T. Faney[2], G. Charpiat[1]

[1] INRIA, LISN, bât 660, Université Paris-Saclay, 91405 Orsay cedex, France

[2] IFPEN, 1 et 4, avenue du Bois-Préau, 92500 Rueil-Malmaison Cedex, France
http://www.ifpenergiesnouvelles.fr

Key words: Graph neural network, interface reconstruction, VoF method

Abstract. The volume of fluid (VoF) method is widely used in multi-phase flow simulations to track and locate the interface between two immiscible fluids. A major bottleneck of the VoF method is the interface reconstruction step due to its high computational cost and low accuracy on unstructured grids. We propose a machine learning enhanced VoF method based on Graph Neural Networks (GNN) to accelerate the interface reconstruction on general unstructured meshes. We first develop a methodology to generate a synthetic dataset based on paraboloid surfaces discretized on unstructured meshes. We then train a GNN based model and perform generalization tests. Our results demonstrate the efficiency of a GNN based approach for interface reconstruction in multi-phase flow simulations in the industrial context.

1 Introduction

Gas-liquid inter-facial flow can be found in a large variety of industrial problems, such as cooling systems for electrical engines, chemical reactors or pore-scale flow in porous media. Among the many numerical methods developed to simulate such gas-liquid flows, an important issue is to track the motion of the free interface. Two classes of methods have emerged in the last decades: front tracking methods and interface capturing methods. The second class is more appropriate for industrial applications as it can more easily deal with topological changes of the free interface (e.g. break up or coalescence) in complex flows. The volume of fluid (VoF) method ([4], [5]) is such an interface capturing method that recovers interface properties (i.e., normal, location, curvature) from the volume fraction of each fluid. One bottleneck of this method is the cost of the accurate computation of the local interface curvature which is essential for the evaluation of the surface tension force at the interface. Currently, the convolution method ([9], [10]) and the height function method ([1], [5]) are the most standard and the most accurate approaches. Nevertheless, these methods are computational expensive and while robust on regular structured grids, they are the source of unphysical fluid motions ([3]) when applied to unstructured tetrahedral grids.

Several attempts to apply machine learning approaches have been done recently. Qi et al. [7] proposed a two-layer neural network to predict the curvature of a 2D surface, trained with a synthetic data-set. The trained model was then implemented into a CFD solver and tested

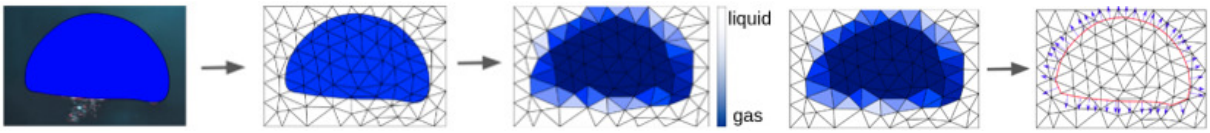
on an unseen data-set. The prediction was accurate enough to consider this approach as a valid solution. Patel et al. [6] extended this approach to 3D geometry using a two-layer neural network as well. Their model was trained considering a synthetic data-set generated from a spherical surface. They then studied the performance of their trained model on analytical and CFD based test cases. Their approach outperformed the convolution method and even matched the accuracy of the height function method [6]. Svyetlichnyy et al. [8] also applied the neural network approach to predict the interface properties for a 2D and 3D surface reconstruction. Their model showed a good performance on the prediction of the normal but not on the curvature and surface location predictions. The major bottleneck of the common VoF methods and the proposed data-driven approaches is the low accuracy on unstructured grids. The VoF performs well on structured-like grids while it becomes unstable on an unstructured grid inducing artificial numerical residual currents. On the other hand the proposed neural network solutions in Qi et al. [7], Patel et al. [6] and Svyetlichnyy et al. [8] cannot be employed with unstructured grids since they do not take into account the variability in the neighbouring cells for a given mesh element. Below is a tabular summary of the involved problems for each available method today. It is worth noting that in industrial problems involving multi-phase-flow simulations, unstructured grids are usually employed. To overcome this problem, we propose to use Graph Neural Networks (GNN) models to deal with complex data structures such as non-euclidean or graph-based inputs.

	Structured grid	Unstructured grid	Computational cost
VoF	Good	Instability	Expensive
Neural Network	Good	Not available	Less expensive

The objective of this work is to propose a machine learning-based framework in place of conventional algorithms in order to model interface dynamics. This is expected to accelerate the interface reconstruction while preserving an accurate prediction. We start by presenting in section 2 a quick review on the VoF method. We then present in section 3 a GNN architecture to recover interface properties in each mesh element from the discretized concentration field. In section 4, we describe how to generate a training dataset that will allow the model to make accurate predictions on a large range of unseen test data representative of industrial applications. Section 5 details the model training procedure on the generated dataset. Finally, we validate our methodology in section 6 using several synthetic test cases.

2 Volume of Fluid method

There are a wide variety of numerical methods to solve multi-phase flow problems. Most of them are based on the resolution of the mass and momentum conservation equations leading to the Navier-Stokes (NS) equations. If we consider for instance two fluids A and B partitioning the simulation domain Ω, a standard approach consists in representing each type of fluid as a single fluid with spatial physical properties and with the characteristic function $\chi_A(x)$ of fluid A, defined as follows: $\chi(x) = 1$ if x in fluid A, else $\chi(x) = 0$. The interface between the two fluids induces a surface tension force that is modeled as a source term in the NS equation. The methods dealing with fluid interfaces differ in the way such interface is represented and computed. Among



(a) Discretization of continuous interfaces (b) Interface reconstruction

Figure 1: Gas-liquid interface representation with volume fraction

the various methods to solve such problems, the VoF method is an Eulerian-Eulerian approach that is capable of accurately capturing the interface from the phase indicator field $\alpha = \int_\tau \chi_A$ where τ is a cell of a mesh discretization Ω_h of Ω. α represents then for each cell τ the volume fraction of fluid A. Fig.1 illustrates a bubble defined by a characteristic function χ_A in blue and the discretization on a mesh with cells colored depending on the value of the discrete phase indicator α. Physical quantities such as the fluid density, viscosity and velocity are expressed as volume fraction weighted sums. For instance, let ρ_A and ρ_B be the density of fluid A and fluid B, the fluid density is then defined as $\rho = \chi_A\rho_A + (1 - \chi_A)\rho_B$. The Navier-Stokes momentum and mass conservation equations can be written as:

$$\frac{\partial \rho \mathbf{U}}{\partial t} + \nabla \cdot (\rho \mathbf{U}\mathbf{U}) = -\nabla p + \rho \mathbf{g} + \nabla \cdot (\mu \nabla \mathbf{U}) + \rho \mathbf{S} + \mathbf{F}_\sigma$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0$$

where t, $\mathbf{U}$, p, ρ and μ are time, the velocity, the pressure, the density and the dynamic viscosity. $\mathbf{g}$ represents body accelerations acting on the fluid, for example gravity. $\mathbf{F}_\sigma$ represents the surface tension force. The surface tension force term at the interface of two fluids is usually modeled as follows:

$$\mathbf{F}_\sigma = \sigma H \mathbf{n} \delta (x - x_s)$$

where x_s are the points on the interface and σ, H and $\mathbf{n}$ are the surface tension coefficient, the mean curvature of the interface and the normal direction to the interface. δ is the Kronecker-δ that allows to account for the surface tension only for points belonging to the interface. An extra advection equation is written to compute the evolution of χ_A discretized on the mesh with α the volume fraction.

3 Graph Neural Network Model

3.1 GNN models

Neural Networks (NN) are the most popular method used to recover data driven models. The most basic neural network, namely the Multi Layer Perceptron (MLP), is composed of a few layers of neurons (input, hidden and output layers). Data is fed to the input layer and predictions are made at the output layer. Neurons are connected between successive layers with multiplicative weights and non-linear activation functions. Convolutional Neural Networks (CNN) perform convolution operations with parameterized filters in order to capture spatial patterns on structured data on regular grids, such as pixel or voxel grids for images and videos.

MLPs and CNNs cannot be applied to irregularly-organized data such as irregular meshes, which can be represented as graphs.

A graph consists of nodes and edges connecting the nodes as illustrated in Fig.2a. We denote a graph by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is a set of $n \in \mathbb{N}$ nodes v_i and $\mathcal{E}$ is a set of $(e_{ij})_{i,j \in n}$ edges. e_{ij} defines the connection between nodes v_i and v_j. The graph connectivity can be represented by the adjacency matrix $\mathbf{A}$ which is a $n \times n$ matrix with $A_{ij} = 1$ if $e_{ij} \in \mathcal{E}$ and $A_{ij} = 0$ if $e_{ij} \notin \mathcal{E}$. Nodes and edges can be equipped with features vectors $\mathbf{x}_v \in \mathbb{R}^d$ (see Fig.2a) and $\mathbf{x}_e \in \mathbb{R}^m$ respectively. Likewise, $\mathbf{X}_v \in \mathbb{R}^{n \times d}$ is the node feature matrix, $\mathbf{X}_e \in \mathbb{R}^{n \times m}$ is the edge feature matrix.

Graph Neural Networks (GNNs) are designed to treat data in the form of graph structures. A GNN takes a graph as its input and returns the same graph with updated features as the output. *Message passing* (MP) is the main mechanism that allows the nodes to communicate through connections in order to update the node features. The MP mechanism on the i node can be summarised in the following three steps:

1. message computation: $\phi_v(\mathbf{x}_{v_j}) \rightarrow \tilde{\mathbf{x}}_{v_j}$ for all nodes j in the neighbour of the i node
2. message propagation: $\phi_e(\tilde{\mathbf{x}}_{v_j}, \mathbf{x}_{e_{ij}}) \rightarrow \bar{\mathbf{x}}_{v_j}$
3. message aggregation: $\phi_a(\mathbf{x}_{v_i}, \underset{j}{\square}(\bar{\mathbf{x}}_{v_j})) \rightarrow \mathbf{x}_{v_i}$ where $\underset{j}{\square}$ is any aggregation function with permutation invariant properties.

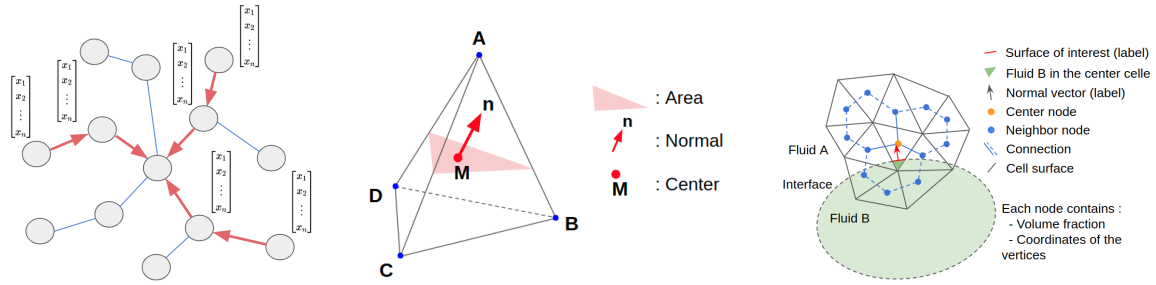
ϕ_v, ϕ_e, ϕ_a are learnable functions, generally MLPs. The same steps are computed for all nodes in the graph in an operation similar to convolution. In our work we employ SAGEConv which is a variant of graph convolution network proposed by Hamilton et al. [2]. We refer the reader to [11], [12] and [2] for a thorough review on the topic.

3.2 GNN architecture design

3.2.1 Data modeling

The objective is to predict the normal, the curvature, the center and the area of the reconstructed surface for each tetrahedral cell from a graph containing the information from neighbouring tetrahedral cells (including the current cell). Therefore, each interface cell is associated to its own graph and interface properties, representing a single data point in the training process. The nodes of the graph are the centroids of each cell, and the edges are defined by the cell connectivity through shared faces. The dimension of the graph is defined by all elements sharing at least one vertex with the current cell. The inputs are provided to the graph as node features for each cell: coordinates of cell vertices and volume fraction. Fig.2b sketches the reconstructed surface properties for a three dimensional tetrahedral cell, and Fig.2c shows the graph for a given interface cell in two dimensions.

In the training algorithm, all the coordinates are normalized by $L = \max(L_x, L_y, L_z)$, where L_x, L_y and L_z are the maximum dimensions in the x, y and z directions over all mesh cells, such that each tetrahedral cell belongs to the unitary cube. The corresponding adimensional curvature H is $H = H'L$ where H' is the original curvature, and the corresponding adimensional area of the reconstructed surface S is $S = S'/L^2$ where S' is the original area. The coordinates



(a) Schematic diagram of a graph

(b) Surface properties

(c) Graph structure

Figure 2: Gas-liquid interfaces representation with GNN graphs

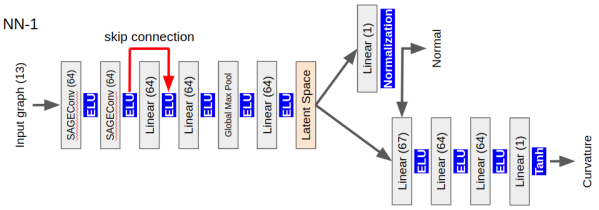
of the interface center M are expressed in the barycentric system as $M = p_1P_1 + p_2P_2 + p_3P_3 + p_4P_4$, where P_1, P_2, P_3, P_4 are the coordinates of each vertex of the tetrahedral cell (see Fig.2b). p_1, p_2, p_3, p_4 are the barycentric coordinates of M and they satisfy the constraint $p_1 + p_2 + p_3 + p_4 = 1$.

3.2.2 Neural network architecture

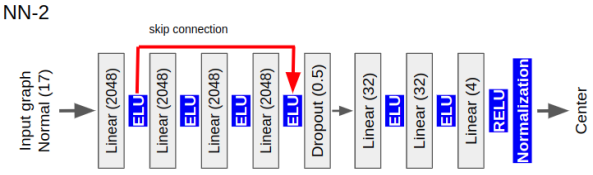
Fig.3 shows the architecture of our model. The design of the architecture is based on a causality criterion: first the normal and the curvature of the interface are computed, then the interface location is defined through the computation of the center by moving the interface in the computed normal direction. Finally, the area of the interface is evaluated as a function of the normal, the center and the vertices of the considered cell. The whole model then consists in staking three different sub-models.

NN-1 is a graph neural network that simultaneously predicts the normal and the curvature from the input graph associated to a given interface cell. The node features are the coordinates of the cell vertices and the volume fraction α of the cells. More specifically, at each node we have the 3 dimensional coordinates for the 4 vertices of a tetrahedron $\{\mathbf{P}_v \in \mathbb{R}^{12}, \forall v \in \mathcal{V}\}$. Here the coordinates of vertices are flattened in a manner that $\mathbf{P}_v = (P_1, P_2, ..., P_4) = (x_1, y_1, z_1, x_2, y_2, \ldots, z_4)$ where $1, 2, 3, 4$ are the number of the vertices of the tetrahedron. We then concatenate $\mathbf{P}_v$ with the volume fraction of the same node as $\mathbf{x}_v = (\alpha, \mathbf{P}_v) \in \mathbb{R}^{13}$. Therefore, an input feature of NN-1 for a graph having n nodes can be denoted as $\mathbf{X}_1 \in \mathbb{R}^{13\times n}$. The first two layers of NN-1 are SAGEConv layers that process the input graph and return a new graph with updated features. Node features are further processed by two linear layers. Skip connections are employed to prevent the vanishing gradient problem. The following global-mean-pool layer returns graph averaged node features. A latent representation of the initial graph is then recovered. The model splits into two branches of linear layers for the prediction of the normal $\mathbf{n} \in \mathbb{R}^3$ and the curvature H.

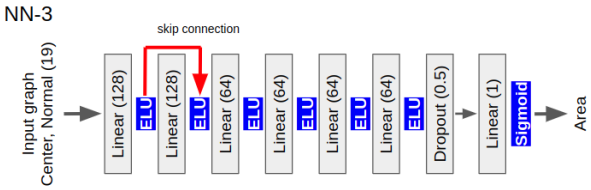
NN-2 is a simple MLP network that predicts the interface center M from the local information available in the current interface cell, including the normal $\mathbf{n}$ predicted by NN-1. We concatenate the volume fraction α, the coordinates of the vertices $\mathbf{P}_{cell} \in \mathbb{R}^{3\times4}$ and the normal $\mathbf{n}$, which is fed in the form of the dot product with $\mathbf{P}_{cell}$, that is, $\mathbf{n}\cdot\mathbf{P}_{cell} \in \mathbb{R}^4$. This helps the neural network



(a) NN-1: Normal/Curvature



(b) NN-2: Center



(c) NN-3: Area

Figure 3: Schematic diagram of the architecture, the numbers in the figure mean the number of node in the layer or the probability of dropout

discriminate vertices that are above or below the interface. Finally, a concatenated form of them $\mathbf{X}_2 = \left(\alpha, \mathbf{n} \cdot \mathbf{P}_{cell}, \hat{\mathbf{P}}_{cell}\right) \in \mathbb{R}^{17}$ is used as input, where $\hat{\mathbf{P}}_{cell} = (P_{cell1}, P_{cell2}, ..., P_{cell4}) \in \mathbb{R}^{12}$ is the flattened $\mathbf{P}_{cell}$ and $P_{cell1}, P_{cell2}, ...$ are the coordinates of the vertices. At the output the barycentric coordinates $p \in \mathbb{R}^4$ are recovered. A final $L1$ normalisation is applied on the output to enforce $\sum_{j=1}^{4} p_j = 1$.

NN-3 is also a simple MLP that predicts the interface area from the available cell quantities $\mathbf{X}_3 = \left(\mathbf{n} \cdot \mathbf{P}_{cell}, \hat{\mathbf{P}}_{cell}, p\right) \in \mathbb{R}^{19}$. A sigmoid function is used after the final layer. Due to the non-dimensionalization by L of the coordinates, the maximum possible area is $\sqrt{2}/2$ which is less than 1, allowing us to skip normalization of the target area.

4 Dataset

4.1 Dataset generation

Previous studies such as Patel et al. [6] uses a spherical surface to generate their synthetic dataset. They performed a prediction of the normal on a sinusoidal surface as shown in Fig.4. We observe that the largest errors are located on saddle point regions, where the surface presents discordant principal curvature sign. It is likely that the model trained on a sphere based dataset is biased and can't generalize to surfaces with non-constant curvatures. To avoid this issue, we use a paraboloidal surface as shown in Fig.4 to generate our dataset. A paraboloid can be described as $\frac{x^2\kappa_1}{2} + \frac{y^2\kappa_2}{2} = z$, where κ_1 and κ_2 correspond to the two curvatures along the principal planes. By sampling explicitly κ_1 and κ_2, we can obtain a dataset representative of a larger range of 2-dimensional surfaces in 3-dimensional space. Our dataset sampling process is as follows:

1. Define the curvature range as $\kappa_1, \kappa_2 \sim -[0.001, 0.5] \cup +[0.001, 0.5]$ and let $u_1 \sim \mathcal{U}(-1, 1)$

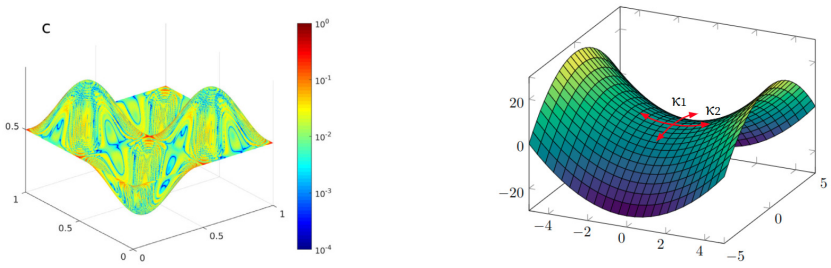


Figure 4: Curvature prediction by Patel et al.[6] (left) and Paraboloid (right)

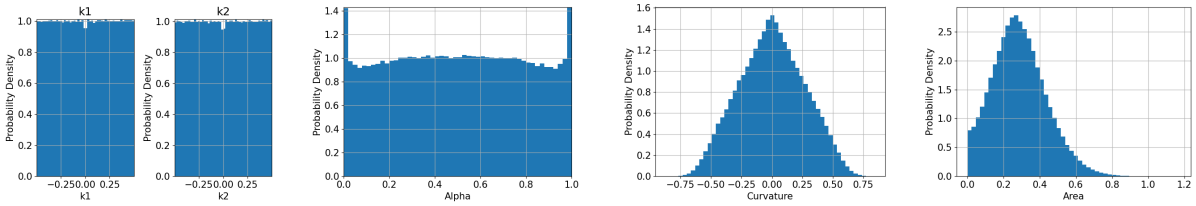


Figure 5: Histograms on the generated dataset: κ_1, κ_2, α, Curvature and Area (left to right)

and $u_2 \sim \mathcal{U}(0,1)$ be uniformly distributed random variables. This paraboloidal surface mimics the interface between two fluids.

2. Form a cubic space around the interface and discretize the inside with a tetrahedron mesh. The dimension of the cube is $(12+u_1)\times(12+u_1)\times(12+u_1)$, its center is given as (u_1, u_1, u_1). The grid resolution is constantly $\Delta = 1$. The curvature in a dimensionless form will be: $\kappa_1\Delta, \kappa_2\Delta \sim \pm[0.001, 0.5]$.
3. Generate a graph around the center of the cube (u_1, u_1, u_1) as illustrated in Fig.2c. As a reminder, we collect the following data from this graph. The label variables (normal, curvature, center and area) are collected at the center cell (see Fig.2c for its definition), and the feature variables (the coordinates of the vertices of a tetrahedron, edges, volume fraction) are collected at all nodes in the graph.
4. Rotate the graphs randomly by $(rot_x, rot_y, rot_z) = (\cos^{-1}(2u_2 - 1), \pi u_2, 2\pi u_2)$. Those formula are known to give a uniform 3D rotation.
5. Repeat this step from 1 to 5 for N_{all} times, N_{all} being the dimension of the desired dataset.

4.2 Dataset analysis

We generate $N_{all} = 500,000$ paraboloids for the dataset. Fig.5 shows the histograms of some properties of the generated dataset. Both κ_1 and κ_2 have a uniform distribution. The mean curvature is defined as $H = \frac{\kappa_1+\kappa_2}{2}$. The area and the mean curvature is non-uniform and non-controllable a priori. Nevertheless, we expect such a dataset generation procedure to better represent general surfaces than a dataset solely based on spherical surfaces.

5 Model training

We divide our 500,000 graphs dataset into a 7:2:1 ratio for training, validation and test. To train more efficiently, some exact label values can also be used as inputs: in NN-2, the input normal can be the normal predicted by NN-1 or the exact normal. This means that two different loss functions need to be defined. The same is true for the the input center in NN-3. Each individual loss is a L2 normalized MSE loss. The total loss is then:

$$\text{Loss}_{\text{total}} = \text{Loss}_{\text{normal}} + \text{Loss}_{\text{curv}} + \text{Loss}^{\text{wl}}_{\text{center}} + \text{Loss}^{\text{wol}}_{\text{center}} + + \text{Loss}^{\text{wl}}_{\text{area}} + \text{Loss}^{\text{wol}}_{\text{area}}$$

where $\text{Loss}_{\text{normal}}$, $\text{Loss}_{\text{curv}}$, $\text{Loss}_{\text{center}}$ and $\text{Loss}_{\text{area}}$ are the loss functions of the normal, curvature, center and area predictions. "wl" means that the label values are used for the input (wl: with-label). "wol" means that the predicted values are used for the input (wol: without-label).

The pytorch-integrated adaptive learning rate is used and set between 1e-2 and 1e-6, reducing it by a factor of 0.9 when the validation loss function doesn't improve for 5 consecutive epochs. The batch size is 512. The *early stopping patience* parameter is set to 80 epochs in order to avoid overfitting. Finally, gradient clipping is set to 1e-2 and the Adam optimizer is used.

Fig.6 shows the learning curves of the total and individual train/validation losses during training. No improvement was observed for the total validation loss after 508^{th} epoch and the training was stopped at the 588^{th} epoch. All types of losses converge without major overfitting. For the center and the area, the loss with label gives a lower loss function than without. This is expected since the input variables in the without label case are predicted variables by another NN and their variables can already contain errors. Those total and individual loss functions at the 508^{th} epoch for the training and the validation are compared in the Table 1. It also shows the prediction on the test dataset by the model trained above. All types of losses stay in the same range as the ones in the training and the validation. This result confirms the generalization of the model trained above.

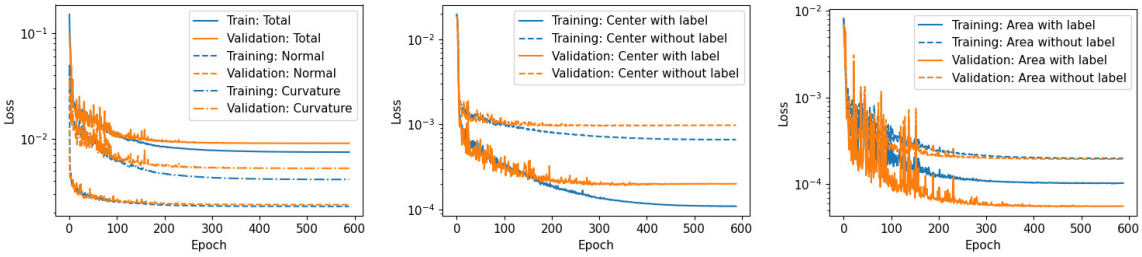


Figure 6: Learning curves: total, normal, curvature (left), center (middle) and area (right)

To analyse the results in-depth, the errors in L_1 norm for each predicted property, as defined in Table 2, are plotted in Fig 7. The histograms show a unimodal shape centered on zero with a long tail for large errors. Table 2 also shows the median of the errors. Fig.8 shows the R^2 score between the prediction and their labels in the test dataset for the scalar variables, *i.e.*, the curvature and the area. The scattered points are colored by the volume fraction α of the cell where the prediction was done. R^2 value is larger than 0.9 for both of them. The value of α

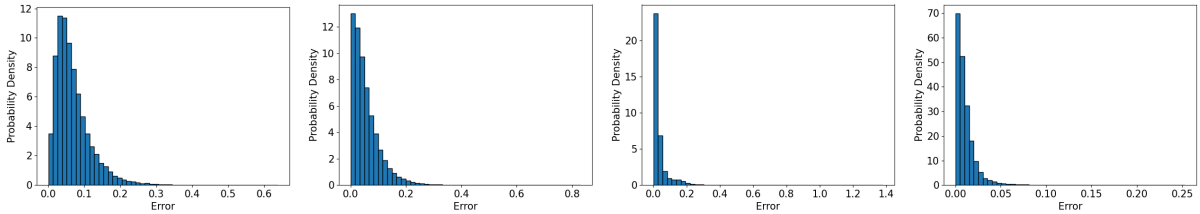


Figure 7: Histogram of the errors: Normal, Curvature, Center, Area

doesn't seem to have a clear relation with the error: high and low α values appear homogeneously in the figure regardless of wether the prediction under- or over- estimates the label.

Table 1: Model performance at the 508^{th} epoch

Type of loss	Training	Validation	MSE
$\text{Loss}_{\text{total}}$	7.50e-3	9.07e-3	9.30e-3
$\text{Loss}_{\text{normal}}$	2.30e-3	2.39e-3	2.41e-3
$\text{Loss}_{\text{curv}}$	4.13e-3	5.26e-3	5.40e-3
$\text{Loss}^{\text{wl}}_{\text{center}}$	1.10e-4	2.01e-4	1.84e-4
$\text{Loss}^{\text{wol}}_{\text{center}}$	6.66e-4	9.76e-4	1.05e-3
$\text{Loss}^{\text{wl}}_{\text{area}}$	1.03e-4	5.53e-5	5.51e-5
$\text{Loss}^{\text{wol}}_{\text{area}}$	1.99e-4	1.95e-4	2.03e-4

Table 2: Model performance on test set

Predicted variable	Median
$n_{error} = \|\mathbf{n}_{label} - \mathbf{n}_{pred}\|$	5.61e-2
$H_{error} = \|H_{label} - H_{pred}\|$	4.16e-2
$M_{error} = \|M_{label} - M_{pred}\|$	1.98e-2
$A_{error} = \|A_{label} - A_{pred}\|$	7.57e-3

6 Model validation

In order to assess the model generalization, we perform a prediction on a 3D sinusoidal surface as shown in Fig 9. The surface is defined by $f(x) = 0.1 \sin 9x \cos 9y$. The space surrounding the surface was voxelized (generation of a tetrahedral mesh) with a constant discretization by $\Delta = 0.05$. We then generate graphs in each of those voxels. Each voxel contains the center point of a graph. Its neighbor voxels contain other nodes of the same graph. This means that a voxel can be a container of a center point as well as a neighbor point. The maximum absolute non-dimensionalized curvature of the test surface is $|H|_{max} = 0.703$. This is in the range of the training data-set ($H_{training} \approx [-0.75, 0.75]$). Each interface cell where the volume fraction is in the range $0.01 < \alpha < 0.99$ is input to the model, while interface cells with very high or low volume fractions are filtered out since we consider these as out of the range of applicability of our model. Fig.10 shows the errors of the prediction on the four variables and their histograms. The error for each variable is defined as Table.3. We see that errors don't have a clear relation to the geometry for any variable. The histograms show a unimodal shape and no abnormal behavior is observed. Table.3 also show the median of the error.

This test surface doesn't have a uniform distribution about α as Fig.11 shows. More samples are found at α close to 0 and 1 (hereby let us call them "marginal α"). This can have an impact on the prediction result. In Fig.11 the prediction is compared to the label colored by α. The marginal α cases have clearly a larger error. For a small marginal α case the model tend to predict smaller than the label, while it tends to predict larger to a large marginal α. This

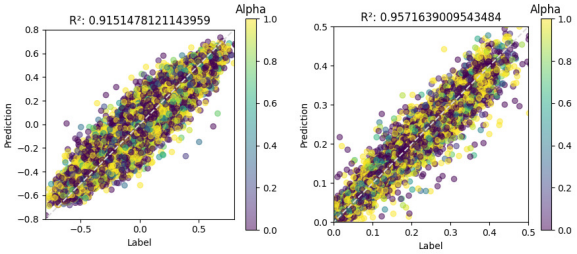


Figure 8: R^2 score on the prediction: Curvature (left) and Area (right)

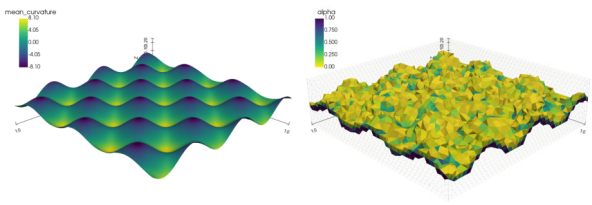


Figure 9: Test surface: colored by the mean curvature (left), voxelized and colored by α (right)

influence of α has not been observed in the test case in the training where α was quasi-uniform. It implies the limited capacity of the model on the marginal α and it should be taken into consideration in case of implementation in a real CFD solver and performing on real use cases.

Table 3: Median of the error for the entire surface

Predicted variable	value	
$n_{error} = \|\mathbf{n}_{label} - \mathbf{n}_{pred}\|$	6.05e-2	$[min(n_{error}), max(0.3n_{error})]$
$H_{error} = \|H_{label} - H_{pred}\|$	4.32e-2	$[min(H_{error}), max(0.2H_{error})]$
$M_{error} = \|M_{label} - M_{pred}\|$	1.84e-2	$[min(M_{error}), max(0.1M_{error})]$
$A_{error} = \|A_{label} - A_{pred}\|$	6.84e-3	$[min(A_{error}), max(0.2A_{error})]$

7 Conclusions and perspectives

In this work, we proposed a machine learning-based framework to model interface reconstruction. To do so, we generated a synthetic dataset built with paraboloid surfaces discretized on unstructured meshes. Then we introduced a GNN based method to compute the interface reconstruction on general unstructured meshes. With the generalization test, we validated our new method. We demonstrated that 1/ GNNs could be an alternative to the conventional surface reconstruction methods, 2/ GNNs could be an effective reconstruction approach to unstructured grids. This can make our method particularly interesting in an industrial context. Further work includes 1/ implementation of the model in a CFD solver, 2/ assessment of time performance in a real CFD simulation, 3/ treatment of boundary conditions.

8 Acknowledgements

This research was supported by DATAIA convergence Institute as part of the “Programme d’Investissement d’Avenir”, (ANR- 17-CONV-0003) operated by INRIA and IFPEN.

References

[1] Sharen J Cummins, Marianne M Francois, and Douglas B Kothe. “Estimating curvature from volume fractions”. In: *Computers & structures* 83.6-7 (2005), pp. 425–434.

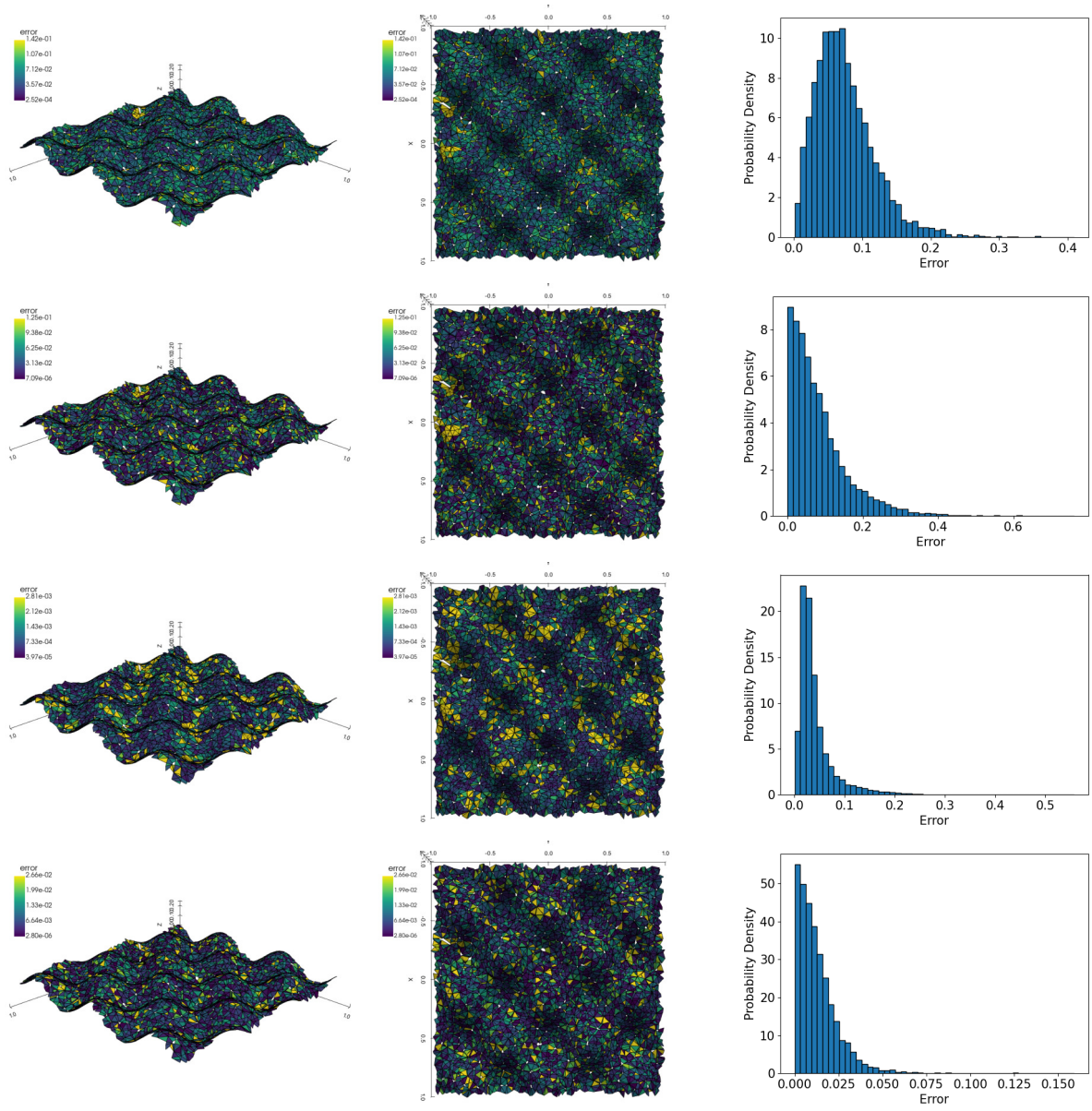


Figure 10: Error plotted on the test surface (left and middle) and its histogram (right) of n_{error}, H_{error}, M_{error}, A_{error} (top to bottom), the visualized range in 3D surface is indicated in the Table 3

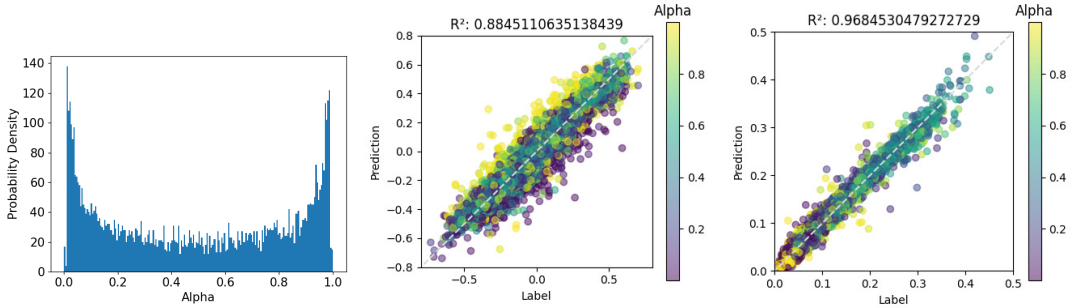


Figure 11: Histogram of the volume fraction α on the test surface and R^2 score on the curvature (middle) and area (right) prediction colored by α

[2] William L Hamilton, Rex Ying, and Jure Leskovec. "Inductive representation learning on large graphs". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 1025–1035.

[3] D.J.E. Harvie, M.R. Davidson, and M. Rudman. "An analysis of parasitic current generation in Volume of Fluid simulations". In: *Applied Mathematical Modelling* 30.10 (2006). Special issue of the 12th Biennial Computational Techniques and Applications Conference and Workshops (CTAC-2004) held at The University of Melbourne, Australia, from 27th September to 1st October 2004, pp. 1056–1066. ISSN: 0307-904X. DOI: https://doi.org/10.1016/j.apm.2005.08.015. URL: https://www.sciencedirect.com/science/article/pii/S0307904X05001666.

[4] Douglas Kothe et al. "Volume tracking of interfaces having surface tension in two and three dimensions". In: *34th aerospace sciences meeting and exhibit*. 1996, p. 859.

[5] Daniel Lörstad et al. "Assessment of volume of fluid and immersed boundary methods for droplet computations". In: *International journal for numerical methods in fluids* 46.2 (2004), pp. 109–125.

[6] HV Patel et al. "Computing interface curvature from volume fractions: A machine learning approach". In: *Computers & Fluids* 193 (2019), p. 104263.

[7] Yinghe Qi et al. "Computing curvature for volume of fluid methods using machine learning". In: *Journal of Computational Physics* 377 (2019), pp. 155–161.

[8] Dmytro Svyetlichnyy. "Neural networks for determining the vector normal to the surface in CFD, LBM and CA applications". In: *International Journal of Numerical Methods for Heat & Fluid Flow* (2018).

[9] Matthew W Williams et al. "Numerical methods for tracking interfaces with surface tension in 3-D mold filling processes". In: *Fluids Engineering Division Summer Meeting*. Vol. 36150. 2002, pp. 751–759.

[10] MW Williams, DB Kothe, and EG Puckett. "Accuracy and convergence of continuum surface tension models". In: *Fluid dynamics at interfaces* (1998), pp. 294–305.

[11] Zonghan Wu et al. "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[12] Jie Zhou et al. "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (2020), pp. 57–81.