EUROGRAPHICS 2025 / A. Bousseau and A. Dai
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 2

VortexTransformer: End-to-End Objective Vortex Detection in 2D Unsteady Flow Using Transformers

X. Zhang[1] and P. Rautek[1] and M. Hadwiger[1]

[1]King Abdullah University of Science and Technology, Saudi Arabia

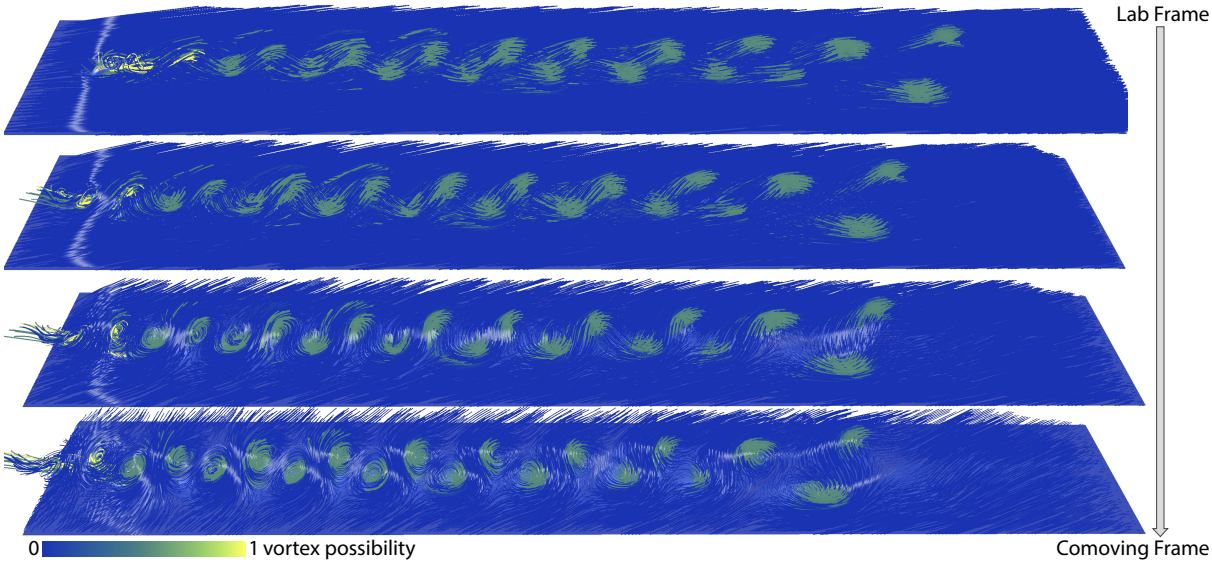

Figure 1: *Results visualized in different reference frames: On the top the pathlines are shown relative to the input lab frame. Vortex structures are not visible from the pathline geometry, however the prediction was computed correctly in this frame of reference. By transforming the pathlines into a moving reference frame, the vortex structures gradually become visible. On the bottom the pathlines are visualized in a co-moving reference frame. The reference frame was not explicitly provided to the VortexTransformer network. In the co-moving reference frame the vortex structures become clearly visible indicating that the network's predictions are correct.*

Abstract

Vortex structures play a pivotal role in understanding complex fluid dynamics, yet defining them rigorously remains challenging. One hard criterion is that a vortex detector must be objective, i.e., it needs to be indifferent to reference frame transformations. We propose VortexTransformer, a novel deep learning approach using point transformer architectures to directly extract vortex structures from pathlines. Unlike traditional methods that rely on grid-based velocity fields in the Eulerian frame, our approach operates entirely on a Lagrangian representation of the flow field (i.e., pathlines), enabling objective identification of both strong and weak vortex structures. To train VortexTransformer, we generate a large synthetic dataset using parametric flow models to simulate diverse vortex configurations, ensuring a robust ground truth. We compare our method against CNN and U-Net architectures, applying the trained models to real-world flow datasets. VortexTransformer is an end-to-end detector, which means that reference frame transformations as well as vortex detection are handled implicitly by the network, demonstrating the ability to extract vortex boundaries without the need for parameters such as arbitrary thresholds, or an explicit definition of a vortex. Our method offers a new approach to determining objective vortex labels by using the objective pairwise distances of material points for vortex detection and is adaptable to various flow conditions.

CCS Concepts

• ***Human-centered computing*** → *Scientific visualization;* • ***Computing methodologies*** → *Neural networks; Vector / tensor field visualization;*

1. Introduction

Vortex extraction from time-varying fluid data is a critical problem with applications across multiple domains, including geophysical meteorology, aerospace applications, and automotive design. However, the inherent complexity of time-varying fluid fields and the elusive nature of vortices present significant challenges for researchers attempting to extract and analyze these structures.

While there is no universally agreed-upon definition of a vortex, it is generally accepted that vortices represent regions of co-rotating fluid particles. A few commonly accepted properties of vortex cores and topological features [BYH*20] are:

- Lagrangian. In 2D, vortex cores are trajectories of material points (i.e., pathlines). In 3D, vortex cores are curves advected by the flow (i.e., path surfaces).
- Objectivity. The vortex structure and vortex core line should remain invariant to rigid observer motion (any time-dependent rotation and translation). We introduce the concept of the observer and provide a definition of objectivity below.

Observer. The term observer refers to a time-varying moving reference frame, abstracting a moving camera that is recording relative velocities. A suitable choice of observer is essential for analyzing complex unstable flows, such as those in CFD wind tunnel simulations. Figure 1 illustrates the significance of reference frames by comparing a cylinder flow in the original reference frame (Fig. 1 top) with a co-moving reference frame, where vortex structures become more readily visible. (Fig. 1 bottom).

For many flow fields, we cannot find one reference frame that follows all the vortices. Prior objective vortex extraction methods split feature detection into two steps: first, they compute an observer that optimally co-moves (often called the *optimal observer*) with fluid flow patterns, using an expensive numerical optimization method [GGT17, RZW*23, HMTR], and second, they detect and visualize features in the observed flow field (i.e., the flow field relative to the *optimal observer*).

Objectivity. Given a time-dependent reference frame transformation (rotation Q and translation c):

$$x^* = Q(t)x + c(t) \tag{1}$$

Objectivity can be formally defined as follows: A scalar field s is objective if the scalar values remain unchanged under any rigid-body reference frame transformation, i.e. $s^*(x^*) = s(x)$. A vector field $\mathbf{v}$ is objective if the vectors transform as: $\mathbf{v}^*(x^*) = Q(t)\mathbf{v}(x)$. A second-order tensor field T is objective if the tensors transform as $T^*(x^*) = Q(t)T(x)Q(t)^T$.

Physical features must be invariant to the observer to ensure they are meaningful [Hal05]. Figure 2 illustrates the importance of objectivity in vortex detection. In Figure 2 (a), a zero field (i.e., all velocities are zero) is visualized from a rotating observer, where the relative velocities appear similar to a vortex due to the camera's rotation. An objective vortex detection method should not misinterpret such camera-induced motion as a vortex. In contrast, Figure 2 (b) shows a genuine vortex that remains invariant under any reference frame transformation.

However, existing deep learning-based vortex extraction methods [DWL*19, DCW*22, BCG20, dTG*24, ZDM*14], pay minimal attention to the characteristics of vortices, with little consideration of fundamental principles such as objectivity and Lagrangian invariance.

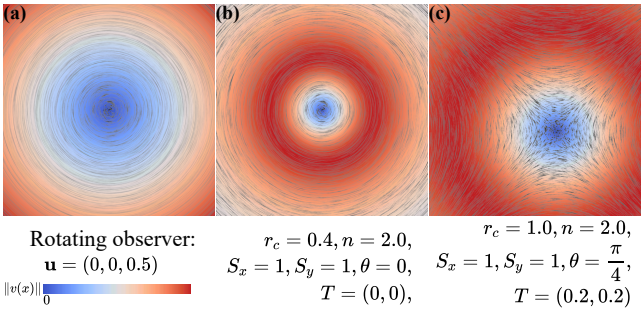

Figure 2: *From left to right: zero field observed by a rotating observer* $\mathbf{u} = (0, 0, 0.5)$*; objective vortex generated by the Vatistas profile; and saddle field generated by the Vatistas profile. The vector field is visualized using LIC [CL93], with color coding indicating velocity magnitude.*

1.1. Contributions

Our main contributions are:

- We propose the first end-to-end, objective, deep-learning vortex extractor using a Lagrangian representation. Our VortexTransformer (VT) learns the flow map from the material trajectories, especially from their relative positions. Our approach is based on the Point Transformer architecture [ZJJ*21]. We added a new *Pathline Sampling Layer* (PSL) to the network to explore the pathline cluster's invariance properties. In the ablation study, we show that PSL improves the performance of network learning and generalization.
- Extending the work of Berenjkoub et al. [BCG20] we provide a methodology for generating a training dataset that contains vortex labels that can be used to train for objectivity. We generate the labeled data by applying multiple observer transformations to steady velocity fields and maintain the known segmentation of the Vatistas vortex profile by forcing the observer transformation at initial time is identity. This solves the problem of a lack of ground truth data with reliable labels in the field of machine learning for *objective* vortex detection.
- We experimentally demonstrate that previous deep-learning vortex extractors were not objective.

2. Related Work

2.1. Traditional Vortex Detection Methods

Traditional vortex detection techniques can generally be classified into two main categories: threshold-based and line-based methods, respectively. Threshold-based methods, including Q-criterion [Hun87], λ_2 [JH95], and Δ [CPC90] criterion, rely on the velocity gradient tensor, emphasizing the local rotational behavior of the flow. None of these local detectors mentioned above are objective, for more details and other criteria, please refer to the overview [GT18].

Objective criteria like Instantaneous Vorticity Deviation (IVD) and Lagrangian-Averaged Vorticity Deviation (LAVD) [HHFH16] offer a more global perspective. However, computing IVD requires two user-defined hyper-parameters: the neighborhood size for averaging the vorticity and the threshold for segmentation. The definition of IVD is provided in Appendix A. Determining appropriate values for these parameters is particularly challenging given the multiscale nature of vortex structures, which exhibit significant variations in both size and angular momentum. This complexity often necessitates a human-in-the-loop approach to ensure that IVD or LAVD methods produce clear and unambiguous iso-contours, creating obstacles for practical applications.

Line-based vortex extraction methods, such as the reduced velocity criterion by Sujudi and Haimes [SH95], identify vortex corelines by analyzing the eigenvectors of the Jacobian matrix J, with vectors v satisfying $v \parallel Jv$. Despite its effectiveness, the Sujudi-Haimes method often generates noisy results, necessitating comprehensive pre-processing and post-processing steps to ensure accurate vortex detection. Fuchs et al. [FPS*08] recommend starting with the traditional Sujudi-Haimes approach and only switching to higher-order methods if necessary, as these can increase computational costs and complexity. Building on this, Weinkauf et al. [WSTH07] and Fuchs et al. [FPS*08] extended the method to address the discrepancy between pathline and streamline corelines in unsteady flows, improving robustness in more dynamic scenarios.

2.2. Machine Learning for Vortex Extraction

In recent years, machine learning and deep learning methods have been increasingly applied to fluid flow visualization. Zhang et al. proposed using the adaptive boosting [ZDM*14] algorithm to merge different local vortex criteria. Various convolutional neural network (CNN) variants have been applied to vortex identification. [DWL*19] propose the first Convolution network (VortexNet), they utilized a user-defined threshold for the IVD as labels to train a convolutional network, extracting vortex from the velocity grid. Subsequent work modified the last several layers of [DWL*19] from linear layers to transposed convolution [WDY*21]. Later studies frequently employed the U-Net architecture [DCW*22, DBW*22]. Although these methods claim objectivity by using the IVD criterion for segmentation labels [DWL*19], IVD itself is known to be of limited use. These methods inherit these limitations, making errors when IVD is unreliable. Furthermore, treating the vector field like an image aligns the network's behavior closer to that of an edge detector.

Berenjkoub et al. [BCG20] utilized Vatistas velocity to generate steady vector fields and more reliable vortex segmentation labels. Berenjkoub et al. [BCG20] achieve objectivity when an optimal reference frame is pre-computed by optimization, the input to the network is transformed input field and the quality of the results largely depends on pre-computed transformation. The choice of the observer, however, is non-trivial, particularly for flows without a global reference frame. Therefore, this approach is not an end-to-end approach like ours.

All methods mentioned apply convolutional neural networks to Eulerian slices of the vector field, neglecting the fluid's time-varying nature and often resulting in suboptimal segmentation. De Silva et al. [dTG*24] present the first Lagrangian vortex detector that avoids training directly on vector fields. Instead, they integrate pathlines and then convert these pathlines into binary images. The binary images are then fed to the network to predict whether a given pathline is inside a vortex. Since a single pathline does not provide enough information about the overall flow field, relying on individual pathlines to infer vortex structures has severe shortcomings. Their approach depends on the accumulated curl along pathlines, which is not an objective property. This leads to false positive detections in certain reference frames.

Lguensat et al. [LSF*18] extracted ocean eddies from sea surface height data, while Franz et al. [FRM*18] trained a neural network using the Okubo-Weiss criterion as input to detect ocean eddies, subsequently employing a recurrent neural network (RNN) for tracking. Bai et al. [BWL19] utilized CNNs to analyze images of streamlines for ocean eddy detection, and Kim and Gunther [KG19] developed a CNN to extract a reference frame that stabilizes unsteady flows for vortex coreline extraction. Liu et al. [LLW*19] also applied CNNs to extract shock waves. Beyond vortex detection, deep learning techniques have been employed to extract flowline features by converting flowlines into binary volumes [HTW18, dSZS*23].

Overall, deep learning approaches so far have ignored objectivity as a prerequisite for vortex extraction. A brief overview of the properties of representative methods is presented in Table 1.

2.3. Objective Vortex Detection Methods

Since the work of Haller [Hal05], objectivity, also called frame indifference, has been recognized as a crucial property that all vortex detectors should have. Often, a specific vortex detection method is defined first, with objectivity then proved specifically only for the given method, such as for LAVD [HHFH16] or for the objective deformation component of the input flow [KPH22].

The first generic method to enforce objectivity was presented by Günther and Theisel [GGT17]. Subsequent work built on the concept of Killing vector fields and Lie derivatives as a general mathematical framework [HMTR]. This enables exploiting the Lie group and Lie algebra structures of observer motions, which is also possible for flow fields defined in curved spaces [RMB*21], and also greatly facilitates interactive exploration of different observers, either from a pre-computed set of observers [ZHTR22], or for observers interactively computed in a region of interest determined by using a lens metaphor [RZW*23]. Recent work has also focused on enforcing that vortex core lines should be guaranteed to be pathlines of the input flow [GT24].

2.4. Point Transformer

Deep learning for point cloud processing often relies on permutation-invariant operators like pointwise MLPs and pooling layers to aggregate features across point sets [QSMG17, QYSG17]. Some approaches enhance this by connecting points into a graph and applying graph operations [LS18, SFYT18], such

Method	VortexNet [DWL*19]	VourtexBoundary [BCG20]	MVUnet [DCW*22]	VortexViz [dTG*24]	Ours
Training data	RealData+IVD	Vatistas Profile	RealData+IVD	RealData+IVD	Vatistas Profile + reference frame transformations
Input	Patch Velocity	Patch Velocity	Patch Velocity+Vorticity+IVD	Flowline Image+Accumulate Vorticity	Relative Positions+IVD
Model	CNN2D	Unet	Unet	CNN2D+MLP	Transformer
Output	Cls	Seg	Seg	Cls	Seg
Objective	✗	~	✗	✗	✓
Data representation	Eulerian	Eulerian	Eulerian	Eulerian/Lagrangian	Lagrangian

Table 1: ***Comparison of deep learning vortex extraction models:*** *Our model is the first to use a transformer architecture which allows us to directly train on Lagrangian data representation (i.e., particle trajectories). We augment the training data with multiple reference frame transformations for better ground truth data. Further, our method is the first one to purely train on objective properties (i.e., relative positions and IVD). In the above table, Cls denotes "vortex classification", Seg denotes "Vortex segmentation".*

as DGCNN [WSL*19], which performs graph convolutions on kNN, or PointWeb [ZJFJ19], which densely connects local neighborhoods. DeepGCNs [LMTG19] explore the benefits of deeper graph convolutional networks for 3D scene understanding.

Following the success of Transformer and self-attention models in revolutionizing natural language processing [Vas17, Dev18, WFB*19, Dai19], there has been a notable shift towards transformer-based architectures in point cloud learning [ZJJ*21,GCL*21,RRL23,WLJ*22,YGX*23,WJW*24]. These transformer-based methods have demonstrated that the scaled-dot attention mechanism is highly effective in capturing the geometric features of point clouds.

3. Method

We highlighted the need for an objective and Lagrangian vortex extraction method. To achieve this, Section 3.1 introduces our training data generation, while Section 3.2 introduces our Vortex Transformer model designed to meet these requirements.

3.1. Data Generation

3.1.1. Steady Field Generation

In order to generate massive segmentation labels of unsteady flow fields, we generate synthetic steady flow fields using the Vatistas [VKM91] experimentally-obtained vortex velocity profile. The steady Vatistas velocity at any point $\mathbf{x} = (x, y)$ of 2D plane is given by:

$$\mathbf{v}(\mathbf{x}) = \mathbf{S}_i \cdot \mathbf{x} \cdot \frac{v_0(\|\mathbf{x}\|)}{\|\mathbf{x}\|}, \tag{2}$$

$$\text{with} \quad v_0(r) = \frac{r}{2\pi r_c^2 \left(\left(\frac{r}{r_c}\right)^{2n} + 1\right)^{\frac{1}{n}}} \tag{3}$$

where $v_0(r)$ is the Vatistas velocity profile [VKM91], r_c denotes the radius with maximum velocity and n controls the shape of the velocity profile function. The effect of n is further illustrated in [KG19]. Matrix $\mathbf{S}_i$ with $i \in \{1,2,3\}$ defines three base shapes (saddle, clockwise vortex, counterclockwise vortex):

$$\mathbf{S}_1 = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}}_{\text{saddle}} \quad \mathbf{S}_2 = \underbrace{\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}}_{\text{center (cw)}} \quad \mathbf{S}_3 = \underbrace{\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}}_{\text{center (ccw)}} \tag{4}$$

For $\mathbf{S}_2$ and $\mathbf{S}_3$, the vortex boundaries are locations with maximal tangential velocity, i.e. vortex area in these two cases are area with positive signed distance:

$$d(\mathbf{x}) = r_c - \|\mathbf{x}\| \tag{5}$$

The steady velocity field $\mathbf{v}(\mathbf{x})$ is then deformed by introducing a deformation matrix D, which is the composition of rotation θ and a non-uniform scaling (s_x, s_y), and a translation vector $\mathbf{T}$:

$$\mathbf{D}(\theta, s_x, s_y) = \begin{pmatrix} s_x \cos(\theta) & -s_y \sin(\theta) \\ s_x \sin(\theta) & s_y \cos(\theta) \end{pmatrix} \tag{6}$$

$$\mathbf{x}' = \mathbf{D} \cdot \mathbf{x} + \mathbf{T} \tag{7}$$

$$d'(\mathbf{x}') = d(\mathbf{D}^{-1} \cdot (\mathbf{x}' - \mathbf{T})) \tag{8}$$

$$\mathbf{v}'(\mathbf{x}') = \mathbf{D} \cdot \mathbf{v}(\mathbf{x}) = \mathbf{D} \cdot \mathbf{v}(\mathbf{D}^{-1} \cdot (\mathbf{x}' - \mathbf{T})) \tag{9}$$

The above recipe is discussed in detail in [BCG20]. Figure. 2 (b-c) presents examples of a steady Vatistas velocity patch. It's important to highlight that the Vatistas function is nonlinear to radius, meaning a rotating observer cannot cancel out the resulting vortex.

We extend [BCG20] by allowing mixtures of up to two vortices ($m \in [1-2]$), given in eq. 10, ensuring that the vortices are sufficiently spaced apart, such that the distance between any two vortex cores exceeds the sum of their radius. This extension enables the generation of more complex and realistic flow scenarios while ensuring that the vortices do not significantly influence each other's

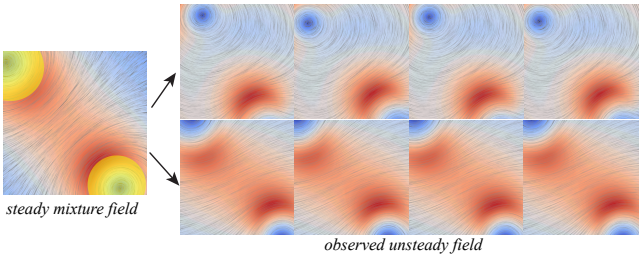

Figure 3: *A steady mixture field with two Vatistas profiles (left) observed by different observers results in various observed unsteady fields (right). In the steady field, yellow is used to indicate the vortex segmentation label, and the corresponding observed unsteady fields shares this segmentation for the first time slice.*

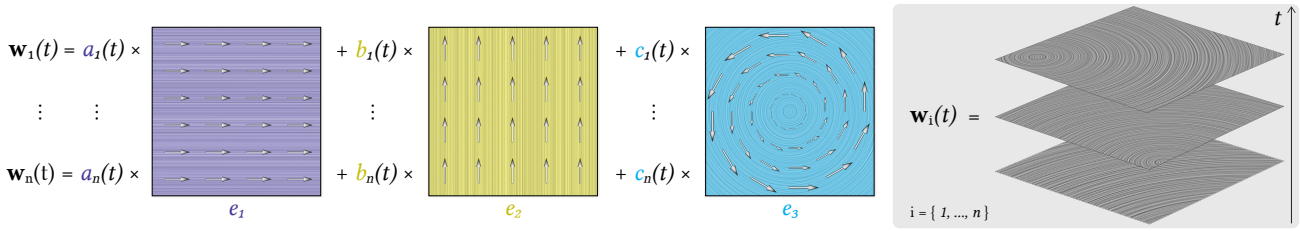

Figure 4: ***Observer representation using Killing fields.*** *Any physical observer is determined by a time-dependent Killing field* $\mathbf{w}(x,t) = a(t)\,\mathbf{e}_1(x) + b(t)\,\mathbf{e}_2(x) + c(t)\,\mathbf{e}_3(x)$. *The basis vector fields* $\mathbf{e}_1$ *(purple),* $\mathbf{e}_2$ *(yellow), and* $\mathbf{e}_3$ *(cyan) are constant basis vector fields. Any possible (time-dependent) observer* $\mathbf{w}_i$ *is solely determined by a time-dependent function* $t \mapsto (a_i(t), b_i(t), c_i(t))$ *of three scalar coefficients* (a_i, b_i, c_i) *per time* t.

segmentation labels.

$$\mathbf{v}(x,y) = \sum_{p=1}^{m} \mathbf{v}_p(x,y), \tag{10}$$

3.1.2. Parameter Space Fitting

In order to restrict the parameter space to physically plausible flows, similar to [KG19], we split the real flow data (**2D Unsteady Cylinder Flow with von Karman Vortex Street** [Pop04], **Boussinesq Flow** and RFC64 [GST16]) into spatial patches, and use gradient descent to find Vatistas parameters to fit every patch. The detail and result Vatistas parameter distribution histogram is given in the Appendix B.

3.1.3. Unsteady Field and Pathline Generation

To achieve objectivity, we introduce arbitrary rigid observers defined by six time-dependent parameters $(a(t), b(t), c(t))$, and their time derivatives $(\dot{a}(t), \dot{b}(t), \dot{c}(t))$ (we ignore higher order derivatives). By defining e_1 and e_2 as two fixed, chosen basis vector fields for translation and e_3 as the basis vector field for rotation, we map the time-dependent parameters $(a(t), b(t), c(t))$ to a time-dependent Killing field [ZHTR22] and its time derivatives. This mapping enables us to describe the observer's motion, as illustrated in Figure 4.

The Killing field represents infinitesimal isometry mapping on any manifold, by integrating observer in the Killing field $\mathbf{w}(x,t) = a(t)\,\mathbf{e}_1(x) + b(t)\,\mathbf{e}_2(x) + c(t)\,\mathbf{e}_3(x)$, we get the observer's motion ϕ_t explicitly as:

$$\phi_{\mathbf{t}}(x) = P(t) + \mathbf{R}(t)\big(x - P(t_0)\big) \tag{11}$$

Here, $P(t)$ represents the observer's position at time t, t_0 is the integration starting time, and $\mathbf{R}(t)$ denotes the observer's frame rotation.

In 2D Cartesian coordinates, $\mathbf{R}(t)$ is given by

$$\mathbf{R}(t) = \begin{bmatrix} \cos\alpha(t) & -\sin\alpha(t) \\ \sin\alpha(t) & \cos\alpha(t) \end{bmatrix}. \tag{12}$$

The observer's pathline $t \mapsto P(t)$ and the integrated frame rotation $\alpha(t)$ are the solutions of

$$\frac{d}{dt}P(t) = \mathbf{w}\big(P(t),t\big), \quad \frac{d}{dt}\alpha(t) = c(t), \tag{13}$$

We solve these two ODEs through numerical integration, enforcing the conditions $\alpha(t_0) = 0$ and $P(t_0) = \mathbf{0}$. This ensures that the vortex segmentation for all observers is consistent at the initial time slice.

$$P(t) = P(t_0) + \int_{t_0}^{t} \mathbf{w}\big(P(\tau),\tau\big)\,d\tau, \quad \alpha(t) = \int_{t_0}^{t} c(\tau)\,d\tau. \tag{14}$$

The reference frame transformation brought by this observer is

$$\phi_{\mathbf{t}}{}^{-1}(x) = \mathbf{Q}(t)x + T(t) \tag{15}$$

where $\mathbf{Q}(t) = \mathbf{R}^T(t), T(t) = P(t_0) - \mathbf{Q}(t)P(t)$. We obtain the observed unsteady field:

$$\begin{aligned} \mathbf{v}^*(\mathbf{x}^*,t) &= \mathbf{Q}(t)\cdot\mathbf{v}(x,t) + \dot{Q}(t)x + \dot{T}(t) \\ x &= \mathbf{Q}^T(t)(x^* - \mathbf{T}(t)) \end{aligned} \tag{16}$$

After fitting the Vatistas parameter distribution, we generated a synthetic dataset of steady fields defined on a 2D unit domain, i.e., $\tilde{\mathcal{X}} \times \tilde{\mathcal{Y}} = [-1,1]^2$, comprising three parts: 1500 patches derived from fitting real data, 1400 randomly generated steady fields using the fitted distribution, and 100 manually added Killing fields to the steady field collection. For each steady field, we generated 20 rigid body motion observers (i.e., Killing fields) by randomly sampling $(a, b, c, \dot{a}, \dot{b}, \dot{c})$ from a uniform distribution within the time domain $\tilde{\mathcal{T}} = [0,1]$, thereby transforming each steady field into 20 unsteady variants. The ground-truth vortex segmentation for these unsteady fields is derived directly from the Vatistas profile parameters and remains consistent across all 20 variants. In total, we generated 3000 steady velocity fields, which were transformed into 60000 unsteady vector fields. The dataset was split into training and validation sets in a 9:1 ratio. Figure 3 shows an example of the steady mixture field and its different observer transformations.

3.2. Vortex Transformer (VT)

Our vortex transformer comprises four primary components: the Pathline Sampling Layer (PSL), the Pathline Transformer (PT) block, the global pooling layer (GP) and the feature propagation layer (FP). An illustration of our vortex transformer is given in Figure 5. Building upon the Point Transformer [ZJJ*21], we have tailored our vortex transformer to effectively process pathline cluster structures, leveraging the unique invariances of pathline clusters.

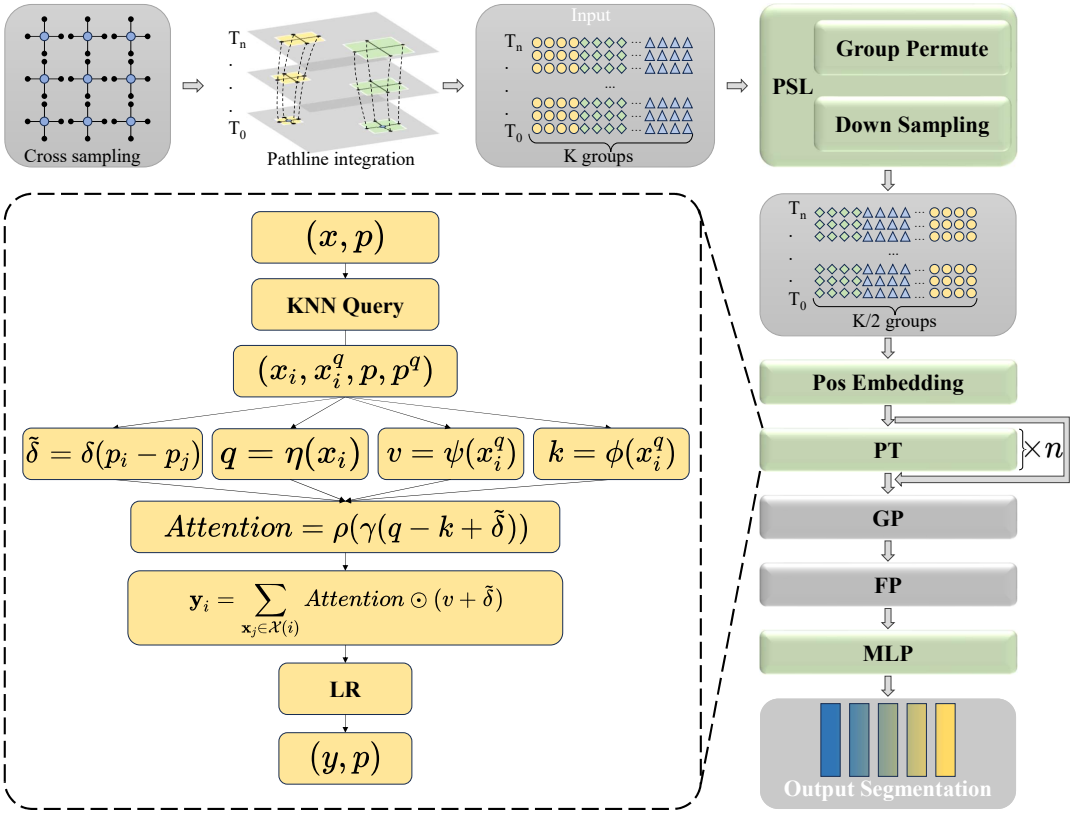

Figure 5: ***VortexTransformer architecture.*** *We first sample K groups of pathlines using a "cross" sampling pattern. The pathlines are then processed through a Pathline Sampling Layer (PSL), which performs group permutation as well as spatial and temporal downsampling operations. A linear layer is applied for positional embedding, and the input is subsequently treated as a point cloud, which is passed through a series of Pathline Transformer (PT) blocks. Each PT block consists of KNN queries, relative position embeddings, and attention mechanisms. Following this, a Global Pooling (GP) operation is performed along the time-sequence dimension, after which Feature Propagation is applied to transfer features from the downsampled pathlines back to the original pathlines. The final segmentation is produced by passing the features through a Multi-Layer Perceptron (MLP) with a sigmoid activation function.*

3.2.1. Pathline Cluster Representation

Cross sampling. Recognizing that individual pathlines are insufficient to capture the local fluid behavior, we always gather seeding points in groups. We sample candidate central positions (blue points in Fig. 5) in a regular grid format. For a given candidate central position (x,y), we seed four points at $(x\pm\Delta, y\pm\Delta)$ around it (black points in Fig. 5). Pathlines are then integrated from these four points, forming a local group and resulting in four pathlines that encode a discrete local flow map.

We seed 64 groups within each velocity patch, resulting in a total of 256 pathlines, which together form a pathline cluster. This representation can be interpreted as a specialized point cloud, where each point in 2D is now represented by (x,y,t).

Given an input pathline cluster of k groups, represented as

$$M(t) = \{\{c_0^1(t), c_1^1(t), c_2^1(t), c_3^1(t)\}, \ldots, \{c_0^k(t), c_1^k(t), c_2^k(t), c_3^k(t)\}\}, \quad (17)$$

where $c_i^j(t)$ is the i-th pathline in group j, consisting of L steps:$c_i^j(t) = (c_i^j(0), c_i^j(1), ..., c_i^j(L-1))$. Pathline cluster is a kind of Lagrangian representation of an unsteady flow map and exhibits several key invariances:

- **Permutation Invariance:** The flow map encoded in the pathline cluster must remain invariant under any reordering of the pathlines. Therefore, our network must act as a symmetry function, ensuring that $S(M(t))$ remains unchanged by permutations of the pathlines, here $S(c_1, c_2, \ldots, c_n)$ represents a symmetry function.
- **Temporal Down-sampling Invariance:** The flow map encoding should remain consistent under temporal down-sampling or re-sampling, assuming negligible discretization effects.
- **Observer Invariance:** Rigid body transformations of the pathline cluster will not affect the intrinsic spiral motion, adhering to the principle of vortex objectivity. A key observation is that, while different observers affect fluid velocity, making feature extraction difficult, a rigid observer does not change the **relative positions** between Lagrangian points.

3.2.2. Pathline Sampling Layer (PSL)

The Pathline Sampling Layer (PSL) performs temporal and spatial down sampling, followed by group permutation. This design preserves local structure while enhancing generalization to varied sampling patterns. A PSL layer down-sample input $\mathbf{X}$ as:

$$\mathbf{X}' = \mathcal{F}(\mathcal{D}t(\mathcal{D}_s(\mathbf{X}))) \tag{18}$$

$\mathcal{D}s$ is the spatial down-sampling operation that randomly picks 50% spatial groups from input pathline cluster. $\mathcal{D}t$ is the temporal down-sampling operation where we randomly pick 50% discrete time steps from the input pathline cluster, note we always keep the first and the last time step. After spatial and temporal down-sampling, a tensor X with shape (K,L,C) reprents K pathlines, L steps,C features, become a new tensor X' with $(K\times 0.5, L\times 0.5, C)$. $\mathcal{F}$ is the group permutation function:

$$\begin{aligned} M'(t) &= \mathcal{F}(M(t)) \\ &= \{\{c_0^{p_1}(t), c_1^{p_1}(t), c_2^{p_1}(t), c_3^{p_1}(t)\}, \ldots, \\ &\quad \{c_0^{p_k}(t), c_1^{p_k}(t), c_2^{p_k}(t), c_3^{p_k}(t)\}\} \end{aligned} \tag{19}$$

where $\{p_1, p_2, \ldots p_k\}$ represent a random permutation of group indices.

Before feeding the sampled pathline into the next stage, there are two linear layers. One is a raw position embedding, which maps the position and time t into the transformer's hidden dimension $h/2$. The other is a feature embedding, which maps the point's feature (IVD) into the transformer's hidden dimension $h/2$.

3.2.3. Pathline Transformer (PT) Block

The Pathline Transformer (PT) block operates in the following steps:

KNN Query. The input pathline cluster is first reshaped from (K,L,C) to $(K\times L, C)$, treating it as a point cloud of $K\times L$ points. For each point, we compute its k-nearest neighbors and collect their features and positions x^q, p^q. Notably, when computing distances, we utilize space-time coordinates (x,y,t) for each material point, implicitly imposing a space-time metric defined by time and space grid resolution and our domain $\tilde{\mathcal{X}}\times\tilde{\mathcal{Y}}\times\tilde{\mathcal{T}} = [-1,1]^2\times[0,1]$.

Relative Position Embedding. For each point and its k-nearest neighbors, we compute relative positions and generate position embedding using a multi-layer perceptron δ to get learnable relative position embedding:

$$\tilde{\delta} = \delta(p_i - p_j) \tag{20}$$

Attention Mechanism. The self-attention mechanism transforms the input pair (x,p) into an output pair (y,p), where x and y represent features, and p denotes position. This transformation leverages three distinct linear layers (η, ϕ, and ψ) to map the input feature x and its KNN features x^q into query (q), key (k), and value (v) tensors, respectively, see Figure 5. We incorporate a subtraction relation and augment both the attention vector and the transformed features with a relative position encoding $\tilde{\delta}$. The attention is $\rho(\gamma(q-k+\tilde{\delta}))$, where γ represents another MLP, ρ denotes softmax function. Note that we use green to highlight learnable layer. Given the input of the PT layer as a set of points $\mathcal{X}(i)$, this process can be formally expressed as:

$$\begin{aligned} q,k,v &= \eta(x_i), \phi(x_i^q), \psi(x_i^q) \\ \mathbf{y}_i &= \sum_{\mathbf{x}_j\in\mathcal{X}(i)} \rho(\gamma(q-k+\tilde{\delta}))\odot(v+\tilde{\delta}) \quad (3) \end{aligned} \tag{21}$$

Similar to the Point Transformer, after the attention layer, a linear layer followed by a ReLU nonlinearity (LR) is applied at the end of this PT block. And residual connection is applied for every PT block as always.

3.2.4. Feature Pooling and Propagation

Our model stacks multiple PT blocks, followed by a global pooling layer. The global pooling layer (**GP**) reshapes the tensor from $(K\times L, F)$ back to (K,L,F) and applies a symmetry function combining average and max operations along the time-sequence dimension, i.e.

$$y_i = mean(x_i) + max(x_i) \tag{22}$$

To conclude the processing pipeline, a non-learnable feature propagation layer (**FP**) propagates features from spatially down-sampled pathlines to the original input pathlines by spatial interpolation. A final MLP produces the predicted segmentation for each pathline. All activation functions used in our network are ReLU, except for the last one of our last MLP, which is a sigmoid function. Implementation details are given in Appendix. C.

3.2.5. Pathline Jittering

As a data augmentation technique, we apply a jittering process to each discretized pathline $c(t)$ in the training set before feeding it into the network. This process enhances the robustness and generalization capabilities of our model by introducing controlled variability to the input data. The jittering procedure is as follows: We iterate through all points P_i, excluding the first and last points (i.e., $i\in[2,n-1]$), for each point P_i, we compute its tangent vector T using the preceding points:$T = P_i - P_{i-1}$. Then we normalize the tangent vector T to obtain a unit direction vector $T_{norm} = \frac{T}{\|T\|}$, disturb the position of P_i by generating a random coefficient ϵ and applying it along the normalized tangent direction:$P_i' = P_i + \epsilon\cdot T_{norm}$. Here, ϵ is a small random value drawn from a Gaussian distribution with a predefined range (1%) to control the magnitude of the jitter.

4. Evaluation

We evaluate the correctness of our proposed VT using **Precision**, **Recall**, and **F1**. Precision is defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \tag{23}$$

where TP denotes True Positives and FP denotes False Positives. **Recall** is defined as:

$$\text{Recall} = \frac{\text{TP}}{\text{TP}+\text{FN}} \tag{24}$$

where FN denotes False Negatives. The **F1** metic is the harmonic mean of Precision and Recall, defined as:

$$\text{F1} = 2\cdot\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}} \tag{25}$$

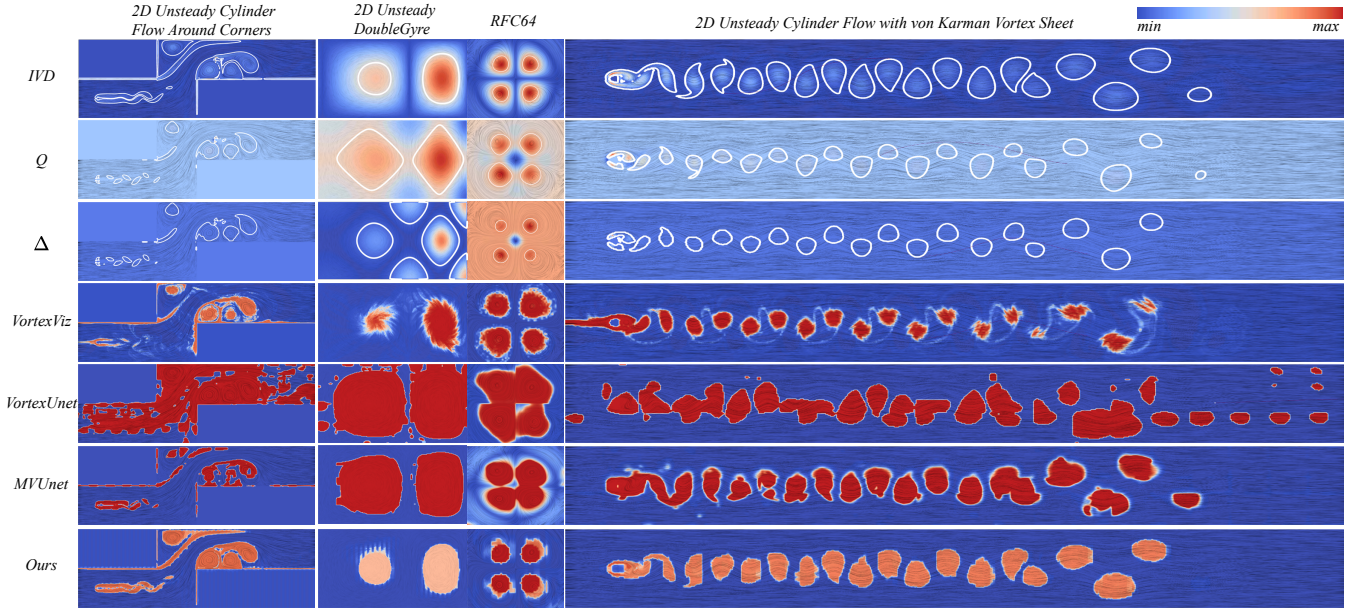

Figure 6: *Visual comparison of different flow scenarios. The first column shows results for the 2D unsteady cylinder flow around corners, followed by the 2D unsteady double gyre in the second column. Subsequent columns display results from RFC64 and the 2D unsteady cylinder flow with von Karman vortex street. Each row represents different techniques: IVD, Q-criterion, Δ-criterion, VortexViz [dTG*24], VortexUnet [BCG20], MVUnet [DCW*22], and our proposed method, highlighting the effectiveness of our approach in capturing complex flow patterns.*

The F1 provides a balance between precision and recall, making it a suitable metric for our evaluation.

4.1. Testing on Synthetic Data

4.1.1. Benckmark

Given the inherent ambiguity in vortex definitions and the absence of ground truth in real-world scenarios, evaluating vortex detection methods poses a significant challenge. While the IVD metric is often employed as a proxy for ground truth, it has notable limitations. IVD can fail in certain scenarios and is highly sensitive to threshold selection. Consequently, segmentation metrics computed on real data using IVD as ground truth become a transform function of human-selected thresholds, rather than a true measure of a vortex detector efficiency.

To provide a place for evaluation, we conduct our quantitative benchmark tests on our synthetic dataset. Specifically, we used the validation split of this dataset, where the ground truth is well-defined and unambiguous. For evaluation, we didn't make a distinction between the validation and test sets. And to compare with other methods, we also train them on our synthetic dataset. We use the best architecture (Unet) from [BCG20] (VortexUnet), MVUnet [DCW*22], and VortexViz [dTG*24] as our baselines for comparison. The code and network details of VortexViz [dTG*24] have not yet been published. We implemented following their description, using three layers of 2D convolution for the flowline images, combined with a two-layer MLP for the associated information vector. After concatenating these two features, another two-layer MLP is applied to generate the final prediction.

*We do not provide results for VortexNet [DWL*19] as it employs a simple 2D convolutional networks to learn from velocity grids, essentially a sub-model of VortexUnet [BCG20] and MVUnet [DCW*22].*

Since [DCW*22], [BCG20] cannot handle unsteady fields directly, we provided them with slices of the unsteady field. For each slice, before feeding the data into their networks, we subtracted the average flow to help mitigate the influence of the observer's motion. The results are presented in Table 2.

Table 2: *Vortex Detection Performance on Synthetic Data*

Method	Precision	Recall	F1
VortexUNet [BCG20]	0.401	0.507	0.448
MVUnet [DCW*22]	0.652	0.88	0.749
VortexViz [dTG*24]	0.788	0.753	0.770
Ours	**0.927**	**0.885**	**0.904**

4.1.2. Beads flow

Beads flow. We use the analytic beads flow defined by Weinkauf and Theisel [WT10]. The beads flow contains rotational as well as contracting motion (negative divergence), see sub-figure (a) of Fig. 7. The rotational component of the Beads flow can be removed by a co-rotating observer, leaving only the remaining contracting

part, see Fig. 7(b). Therefore, the Beads flow should not be classified as a vortex.

Divergence-free beads flow. This vector field is a divergence-free adaptation of the original Beads flow data [GST16], commonly used as a benchmark for rotation-invariant vortex detection. It represents a zero field observed by a rotating camera, thus it should not be classified as an objective vortex.

Figure. 7 presents a comparison between our approach, existing deep learning methods, and traditional threshold-based techniques. Velocity grid-based methods like VortexUnet [BCG20] predominantly focus on edge patterns that ensemble rotational features. Although MVUnet [DCW*22] incorporates IVD into its inputs, it still struggles to determine the presence of vortices within the beads dataset. VortexViz [dTG*24] converts the flow into binary pathline images but relies on single pathlines for prediction, lacking information to remove the observer's motion and understand the flow's true structure. Figure 7 demonstrates that previous deep learning vortex extractors are not objective.

In contrast, our proposed VortexTransformer successfully distinguishes between true vortices and rotational artifacts. It determines that all rotational components in this field can be fully described by a rotating reference frame, whereas conventional deep learning methods based on velocity grids are misled by rotational edge patterns. This highlights the objectivity of our approach in avoiding such false positives.

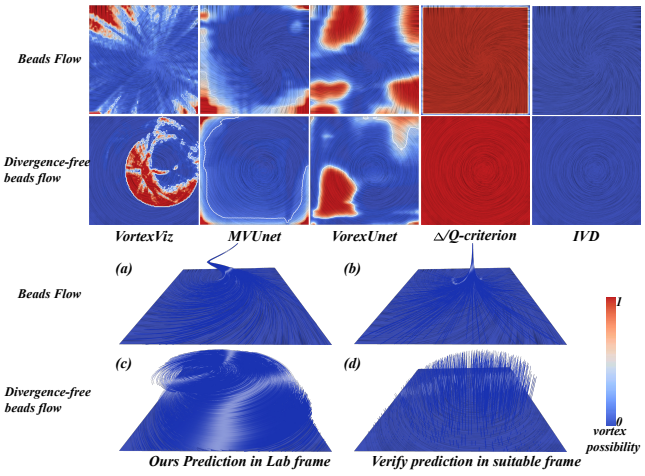

Figure 7: *Comparison of results on the **Beads flow** and **Divergence-free beads flow** data. The first two rows display results from existing techniques: VortexViz, MVUnet, VortexUnet, Δ/Q-criterion, and IVD. (a) and (c) present the outcomes of our approach, showcasing the network's predictions directly in the input flow's lab frame. In contrast, (b) and (d) verify these predictions by transforming them into a suitable reference frame, where the rotational velocity component is removed. In this frame, the Beads flow is a pure contraction, and the Divergence-free beads flow is a zero field. Our model accurately predicts a zero probability for each pathline belonging to a vortex.*

4.1.3. RFC flow

The RFC flow [GST16] is an analytical 2D vector field containing four vortices swirling around four vortex centers. It is constructed from a steady flow field given by,

$$\mathbf{v}(x,y) = \begin{pmatrix} -x(2y^2-1)e^{(-(x^2+y^2))} \\ y(2x^2-1)e^{(-(x^2+y^2))} \end{pmatrix}. \tag{26}$$

The steady vector field is defined in domain $\mathbb{D} = [-2,2] \times [-2,2]$. The final unsteady flow data is generated from an observer rotating with unit speed around an axis that passes through the origin and is orthogonal to the 2D plane. We discretized the flow field into a $64 \times 64 \times 64$ velocity field, with the results presented in the third column of Figure 6. For this flow field, IVD can serve as a reference for proximal ground truth.

4.2. Testing on Numerical Data

We used four unsteady 2D numerical datasets, namely **2D Unsteady Cylinder Flow Around Corners** [RG19], **2D Unsteady DoubleGyre** [SLM05], **2D Unsteady Cylinder Flow with von Karman Vortex Street** [Pop04] and a **Boussinesq Flow** for visual comparison. Figure 6 presents a comparison of our method with baseline methods and threshold-based approaches. We carefully set an appropriate threshold for the IVD metric (rendering as white iso-contour), allowing it to serve as a proximate ground truth in the displayed time slices across the various fluid datasets.

It is important to highlight that both VortexUnet and our method were trained on our synthetic data and have not been exposed to the real fluid test data. In contrast, VortexViz and MVUnet were trained using real fluid data, as described in their respective publications. From the visual comparisons, despite our network not being trained on real data, it successfully segments boundaries that closely align with the IVD, demonstrating superior performance compared to networks trained on real datasets.

4.3. Robustness Against Noise

In many fluid scenarios, the objective IVD metric serves as a solid foundation for vortex detection. However, the need for neural networks arises for several reasons. One compelling reason for employing neural networks is their superior robustness. Figure. 8, the first and second rows present the results of adding noise to our RFC64 dataset. The second column illustrates that 10% noise (to magnitude) has a minimal impact on our neural network model. In the third column, the results for the Q-criterion (first row) and IVD (second row) show that the introduction of 10% noise significantly deteriorates the performance of both metrics. The fourth and fifth columns demonstrate aggressive testing with noise magnitudes of 20% and 50%, respectively, yet our neural network still manages to extract some accurate pathline segmentation. In each column (except the third column), the first row displays the predictions made by our network in the lab frame, while the second row verifies the network's pathline segmentation in a reference frame that removes the observer's rotation. We employ color coding for the pathlines based on the probability of their belonging to a vortex or not.

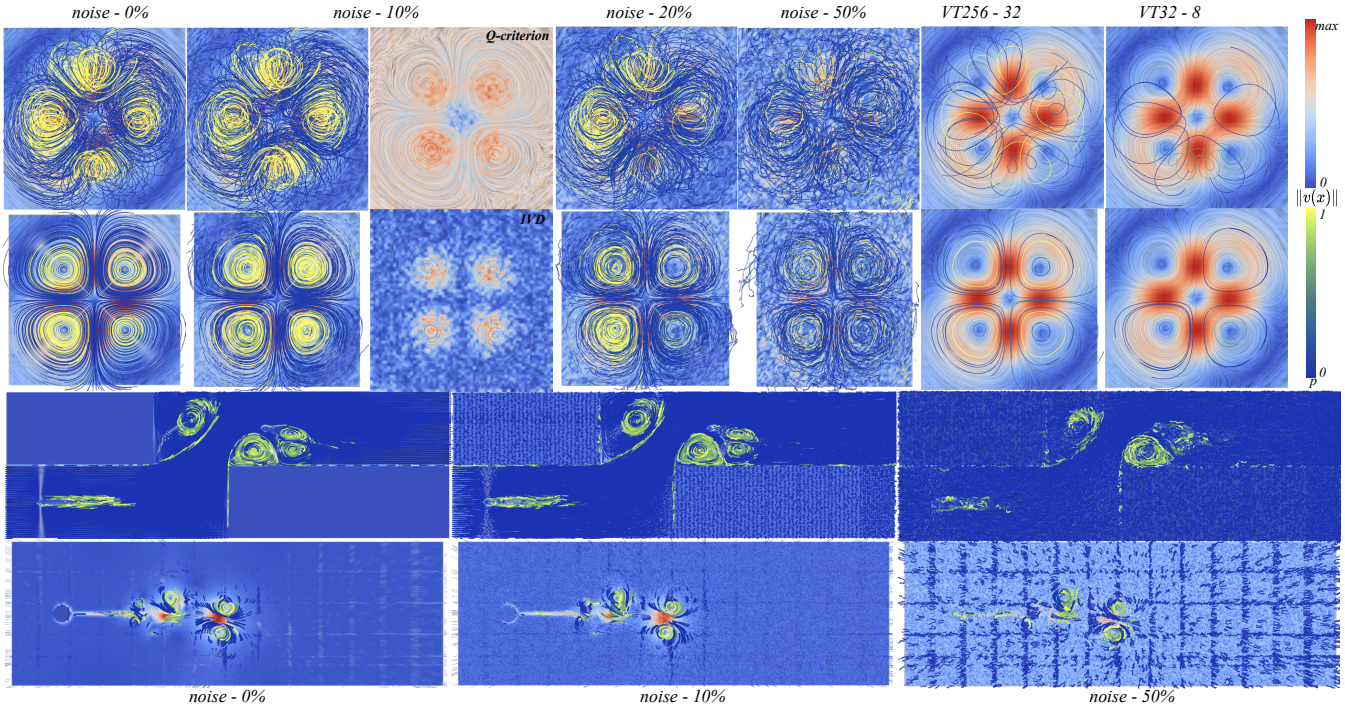

Figure 8: ***Impact of noise and down-sampling.*** *Rows 1 and 2 display our VortexTransformer (VT) model's predictions on the RFC64 dataset with increasing noise levels (0%, 10%, 20%, and 50%). Row 1 presents the predictions in the lab frame, while Row 2 showcases predictions in a suitable rotating reference frame. Row 3 illustrates the 2D unsteady cylinder flow around corners with added noise, and Row 4 depicts the Boussinesq flow under similar conditions. Pathline color coding indicates vortex probability, while the color coding of the plane represents velocity magnitude.*

The third row presents results from adding noise to the **2D Unsteady Cylinder Flow Around Corners**, while the fourth row displays results from adding noise to a **Boussinesq** flow. Overall, our network demonstrates strong robustness to noise, underscoring its efficacy as a vortex detection method.

4.4. Pathline Down-Sampling

We represent the flow field using a pathline cluster $M(t) = \{C_k(t)\}$, which describes the discrete flow map. During training, we sample 256 pathlines (64 cross groups). However, this raises the question: Is this sampling rate optimal? To investigate this, we conducted experiments by progressively reducing the sampling rate and observed its effects on segmentation performance in unsteady fields. Table 5 presents the results, showing the segmentation metrics for pathlines (without propagation to the input patch grid). Surprisingly, our VT model demonstrates resilience to pathline down-sampling. However, the mismatch between training and testing conditions has a more significant impact when the model is trained with fewer than 64 pathlines. A visual example of a model trained with 256 pathlines but tested with 32 pathlines (VT256-32), as well as a model trained with 32 pathlines and tested with 8 pathlines (VT32-8), is presented in the fifth and sixth columns of Figure 8.

To achieve a clear vortex boundary segmentation on the input patch grid, we must propagate the pathline segmentation back to the grid. This process can be computationally intensive for a large number of pathlines. If an insufficient number of pathlines is used, the resulting grid segmentation may suffer from inadequate sampling, potentially leading to ambiguities or inaccuracies in the vortex boundary definition.

4.5. Point Feature

Table 3: ***VortexTransformer with different point features.*** *v represents velocity, dis is the distance to the pathline origin, and pos,t indicates the spatial 2D coordinates and time of a point. Number highlighted in green show performance improvements compared to using only pos,t. Rows shaded in gray is our final model.*

Features	Precision	F1
pos,t	0.718	0.683
pos,t + v	**0.772** (+0)	**0.696** (+0.013)
pos,t + dis	**0.772** (+0.054)	**0.699** (+0.016)
pos,t + IVD	**0.927** (+0.209)	**0.904** (+0.221)
pos,t + IVD+dis	**0.921** (+0.203)	**0.904** (+0.221)
pos,t + IVD+dis+v	**0.921** (+0.203)	**0.904** (+0.221)

Our VortexTransformer accepts point coordinates from pathlines

and can also incorporate other features. In Table 3, we compared the validation F1 and Precision metrics by adding different features (velocity, distance to pathline origin, IVD). Through this experiment, we found that when the network has access to IVD information, the velocity and distance to origin features do not contribute significantly. Conversely, when IVD is not included, the distance to origin provides a modest performance improvement, while velocity along the pathline proves to be unimportant. Ultimately, our VT Transformer model takes the coordinates of points along the pathline (including time) as input, with IVD used as an additional feature.

4.6. Impact of KNN Neighborhood Size

Table 4: ***Performance comparison of different KNN neighbor size on various datasets.***

Neighborsize(k)	Precision	Recall	F1
2	0.899	0.759	0.823
4	0.896	0.787	0.838
8	0.901	0.810	0.853
16	0.927	0.885	**0.904**
32	0.873	0.898	0.885
64	0.527	0.829	0.644

The Pathline Transformer (PT) layer incorporates a KNN mechanism. We investigated the effect of the neighborhood size parameter k, which defines the number of nearest neighbors considered for each point, thereby influencing the receptive field size. Table 4 summarizes the results across different k values. We observed a increase in the false positive rate as K grows, leading to a decline in both precision and F1 score.

5. Ablation Study

5.1. Pathline Sampling Layer (PSL)

We conducted a comparison between our Pathline Sampling Layer (PSL) and the commonly used Farthest Point Sampling (FPS) technique. In Figure 9, the cyan curve labeled **"VT"** represents our proposed model, which integrates PSL, Relative Position Embedding (RPE), and jittering of the pathlines. The red curve ("xPSL") illustrates the training and validation loss when PSL is replaced by FPS (essentially making the network operate similarly to Point Transformer [ZJJ*21], while we keep the final pooling layer executed along the pathline sequence dimension). Our results, averaged over three runs, indicate that PSL consistently outperforms FPS, which behaves similarly to random sampling. We attribute this to the fact that individual pathlines alone fail to capture sufficient information about the overall flow map.

5.2. Relative Position Embedding

The black curve ("xRPE") in Figure 9 shows the effect of replacing the relative position embedding described in Eq. 20 with:

$$\tilde{\delta} = \delta(p_i)$$

Table 5: ***Impact of downsampling pathlines on F1 score for vortex detection.*** *We investigate the effect of pathline quantity in our synthetic test dataset. The table presents the performance results of testing with varying numbers of pathlines after training with different quantities.*

model trained pathlines	model test pathlines					
	256	128	64	32	16	8
256	0.904	0.915	0.906	0.919	0.920	0.915
128	0.921	0.921	0.928	0.903	0.877	0.895
64	0.831	0.825	0.871	0.848	0.833	0.782
32	0.760	0.814	0.905	0.756	0.810	0.766
16	0.832	0.794	0.824	0.840	0.860	0.762
8	0.609	0.631	0.631	0.725	0.755	0.855

This modification removes relative position information from the network, resulting in a significant increase in both training and validation loss. The curve represents the average of three runs, and the outcome reinforces our intuition that learning directly from absolute positions is inadequate, as the relative positional relationships between points are crucial for objective feature extraction.

In the Eulerian perspective, fluid dynamics are represented as a velocity field, making the extraction of objective features challenging when subjected to arbitrary observer transformations. Traditional approaches generally address this challenge by either: (1) employing optimization techniques to identify an appropriate observer that stabilizes the flow as much as possible, followed by feature extraction in the stabilized field, or (2) directly computing objective measures, such as Instantaneous Vorticity Deviation (IVD). However, vortex extraction based on IVD often requires human intervention, since vector fields can exhibit varying dynamic scales and ranges.

From the Lagrangian perspective, arbitrary rigid observer transformations alter the observed pathlines but preserve the relative positions between pathlines. Rigid observer motion is, in fact, the integral of a Killing vector field, where the Killing field represents an infinitesimal isometry. Thus, observer transformations can be understood as the integration (or composition) of time-dependent isometry mappings. Crucially, throughout these isometry mappings, the relative positions between pathlines remain invariant.

5.3. Jittering

The blue curve ("xjittering") in Figure 9, also averaged over three runs, demonstrates the impact of removing jittering from the training process. While the training loss decreases more rapidly without jittering, the validation loss converges to a higher value. This suggests that jittering aids in reducing validation loss, enhancing the model's generalization capability and robustness to variations in the data.

6. Conclusion

In this work, we introduced VortexTransformer (VT), a novel deep learning approach that leverages a Lagrangian perspective to perform end-to-end vortex detection using pathlines. Unlike tradi-

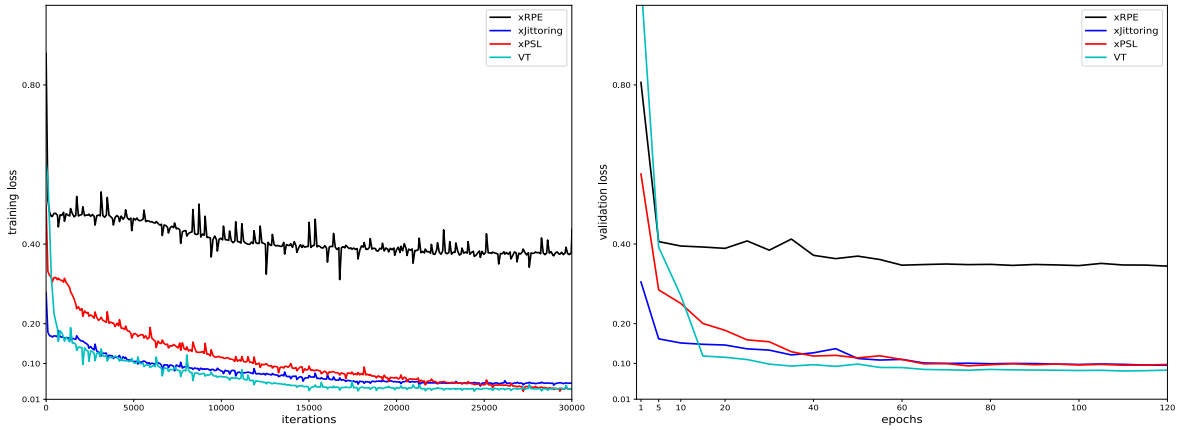

Figure 9: *Training (left) and validation (right) loss curves from the ablation study. The cyan curve represents our proposed VT model. The red curve corresponds to the model with the Pathline Sampling Layer (PSL) removed. The blue curve shows the model without the jittering process. The black curve illustrates the results when the relative position encoding (RPE) is replaced by absolute position embedding in the MLP.*

tional methods that rely on grid-based velocity fields and subjective thresholds, VT operates directly on material trajectories, learning from the relative positions of pathlines. This allows the model to objectively extract both strong and weak vortex structures, independent of arbitrary reference frame transformations.

Our Pathline Sampling Layer (PSL) was shown to improve performance over standard farthest point sampling (FPS) techniques, as demonstrated in the ablation study. Moreover, by incorporating relative position embedding (RPE) and jittering techniques, we further enhanced the model's generalization and robustness, particularly in dynamic flow conditions.

To address the lack of reliable ground truth data for objective vortex detection, we extended the dataset generation method of [BCG20, KG19] by applying multiple observer transformations to synthetic vortex configurations. This approach provides the necessary labeled data of an unsteady vector field to train for objectivity, filling a critical gap in the field.

Through extensive experimentation, we demonstrated that existing deep-learning vortex extractors lack objectivity, while VortexTransformer successfully overcomes these limitations, making it a robust tool for vortex detection in various flow scenarios. We believe that this method offers a significant step forward in the accurate, objective extraction of vortex structures and can be applied to a wide range of real-world fluid dynamics problems. The code and trained models will be provided in the supplementary materials upon acceptance of the paper for publication.

Acknowledgments

This work was supported by King Abdullah University of Science and Technology (KAUST) baseline funding.

Appendix A: Instantaneous Vorticity Deviation (IVD)

The vorticity ω of the fluid is defined as $\omega = \nabla \times \mathbf{v}$, and the instantaneous spatial mean vorticity $\bar{\omega}(t)$ over a domain $U(t)$ is given by:

$$\bar{\omega}(t) = \frac{\int_{U(t)} \omega(\mathbf{x},t)\, dV}{\mathrm{vol}(U(t))},$$

where $\mathrm{vol}(\cdot)$ represents the volume for three-dimensional flows and the area for two-dimensional flows. The element dV corresponds to the volume or area element within $U(t)$. The *instantaneous vorticity deviation (IVD)* is defined as the absolute difference between the vorticity at a point and the average vorticity of its local neighborhood:

$$\mathrm{IVD}(\mathbf{x},t) = |\omega(\mathbf{x},t) - \omega_{\mathrm{avg}}(t)|\,.$$

Appendix B: Vatistas Parameters for Data Synthesis

Parameters Fitting

We split the real flow data into spatial 32x32 patches, the patch is sliding in the real flow dataset like a convolution kernel. And we take the result patches from time step 600-1000 of **2D Unsteady Cylinder Flow with von Karman Vortex Street** [Pop04], time step 500-800 of **Boussinesq Flow** and all time slice of RFC64 [GST16]). Following [KG19], we employ 200 iterations of simulated annealing, followed by an additional 200 iterations of gradient descent, using the same distance metric to fit the real flow. We statistically recording fitting result parameters as gaussian distribution, and list in Table 6.

Data Generation

For each steady field, we generated 20 rigid Killing observers by randomly sampling $(a,b,c,\dot{a},\dot{b},\dot{c})$ from a uniform distribution within the time domain $\tilde{\mathcal{T}} = [0,1]$, thereby deforming each steady field into 20 unsteady variants. The unsteady field is first generated

in analytical way, defined in physical domain $\tilde{\mathcal{X}} \times \tilde{\mathcal{Y}} = [-1,1]^2 \times \tilde{\mathcal{T}} = [0,1]$, and then we resample it to resolution $5 \times 32 \times 32$. Table 7 presents the parameters used in our unsteady data generation. Finally, we introduce ±1% noise to the magnitude of the fields. This choice balances computational efficiency with the local nature of objective reference frame transformations. Notably, the temporal derivative to be minimized depends only on up to second-order derivatives [GGT17]. Given the unit domain and its resolution, we have calibrated the transformation parameters to appropriate levels.

Table 6: ***Distribution of Individual Vatistas Parameters.***

Parameter	Mean (μ)	Standard Deviation (σ)
θ	0.00	0.80
s_x	0.00	0.26
s_y	0.00	0.25
r_c	1.25	0.93
n	2.06	0.75
t_x	0.00	0.61
t_y	0.00	0.62

Table 7: ***Parameters for data synthesis and analysis.***

Param.	Description	Value
Ω	Domain	$[-1,1]^2 \times [0-1]$
L	Pathline steps	16×16
c	angular velocity	$[-0.11, 0.11]$
a, b	translation velocity	$[-0.15, 0.15]$
$\dot{a}, \dot{b}, \dot{c}$	deriv. of a, b	$[-0.01, 0.01]$
ϵ	Pathline jittering range	$[-0.01, 0.01]$
Δ	corss sampling distance	$[-0.005, 0.005]$

Data Normalization

Our training dataset is defined in the 2D unit domain $\tilde{\mathcal{X}} \times \tilde{\mathcal{Y}} = [-1,1]^2$ following a time domain $\tilde{\mathcal{T}} = [0,1]$. When testing on real flow data, we need to transform the input pathline cluster (point cloud) from the patch's physical domain to this domain. For a patch slices $\mathbf{v}_p : \mathcal{X} \times \mathcal{Y} \times \mathcal{T} \to \mathcal{X} \times \mathcal{Y}$, defined in the domain $\mathcal{X} \times \mathcal{Y} \times \mathcal{T} = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}] \times [t_{\min}, t_{\max}]$, we transform the original point (x, y, t) to $(\bar{\mathbf{x}}_p, \bar{\mathbf{x}}_p, \bar{\mathbf{x}}_p)$ in the unit domain via:

$$\begin{aligned} \bar{\mathbf{x}}_p &= 2 \cdot \frac{\mathbf{x} - x_{\min}}{x_{\max} - x_{\min}} - 1 \\ \bar{\mathbf{y}}_p &= 2 \cdot \frac{\mathbf{y} - y_{\min}}{y_{\max} - y_{\min}} - 1 \\ \bar{\mathbf{t}}_p &= \frac{\mathbf{t} - t_{\min}}{t_{\max} - t_{\min}} \end{aligned} \tag{27}$$

This normalization ensures consistency across all input data, facilitating improved network training and generalization.

Appendix C: Training Details

Our baseline vortex transformer consists of $n = 3$ PT blocks, the hidden dimension h of the transformer is 144, with one pathline sampling layer (PSL) positioned before the first PT block. We use AdamW [Kin14] optimizer with a weight decay of 1×10^{-6} and implemented a cosine learning rate scheduler. The training batch size was set to 100, featuring a warm-up period of 5 epochs, followed by 195 epochs, in total 200 epochs. The generated pathline within a patch has 64 groups, 4 pathlines per group, and pathline has $L = 16$ steps. We employ the ReLU activation function throughout the network, except in the final layer where we utilize a sigmoid activation to output the vortex probability. For training, we apply a binary classification loss, defined as:

$$\mathcal{L}_{\text{binary}} = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)], \tag{28}$$

where y_i represents the ground truth label, $\hat{y}_i$ is the predicted probability of the pathline belonging to a vortex, and N is the total number of samples.

Performance

The training was conducted on an Intel(R) Xeon(R) Gold 6230R CPU @ 2.10GHz (2 processors) and an NVIDIA A-100 GPU. The training process for 200 epochs took approximately 14 hours.

Table 8: ***Batch inference Time of different methods.***

Method	Range([s])	Average([s])
VortexUNet [BCG20]	$2.7 \sim 4.2$	2.8
MVUnet [DCW*22]	$2.5 \sim 4.8$	2.5
VortexViz [dTG*24]	$0.5 \sim 1.8$	**0.6**
Ours	$0.8 \sim 2.4$	1.2

During testing, we split real flow data into 32×32*patch* and seeding 256 pathlins within it, and the position of pathlines has been normalized, see Appendix. B.

For visualization, we load the model using libtorch. Table 8 presents the inference time of our method compared to various baselines. The reported metric represents the time (in seconds) required for a single forward pass of a 32×32 patch with a batch size of 8.

References

[BCG20] BERENJKOUB M., CHEN G., GÜNTHER T.: Vortex boundary identification using convolutional neural network. In **2020 IEEE Visualization Conference (VIS)** (Oct. 2020), pp. 261–265. doi:10.1109/VIS47514.2020.00059. 2, 3, 4, 8, 9, 12, 13

[BWL19] BAI X., WANG C., LI C.: A streampath-based rcnn approach to ocean eddy detection. **IEEE Access 7** (2019), 106336–106345. 3

[BYH*20] BUJACK R., YAN L., HOTZ I., GARTH C., WANG B.: State of the art in time-dependent flow topology: Interpreting physical meaningfulness through mathematical properties. **Computer Graphics Forum 39**, 3 (June 2020), 811–835. doi:10.1111/cgf.14037. 2

[CL93] CABRAL B., LEEDOM L. C.: Imaging vector fields using line integral convolution. In **Proceedings of the 20th annual conference on Computer graphics and interactive techniques** (1993), pp. 263–270. 2

[CPC90] CHONG M. S., PERRY A. E., CANTWELL B. J.: A general classification of three-dimensional flow fields. **Physics of Fluids A: Fluid Dynamics 2**, 5 (1990), 765–777. 2

[Dai19] DAI Z.: Transformer-xl: Attentive language models beyond a fixed-length context. **arXiv preprint arXiv:1901.02860** (2019). 4

[DBW*22] DENG L., BAO W., WANG Y., YANG Z., ZHAO D., WANG F., BI C., GUO Y.: Vortex-u-net: An efficient and effective vortex detection approach based on u-net structure. **Applied Soft Computing 115** (Jan. 2022), 108229. doi:10.1016/j.asoc.2021.108229. 3

[DCW*22] DENG L., CHEN J., WANG Y., CHEN X., WANG F., LIU J.: Mvu-net: A multi-view u-net architecture for weakly supervised vortex detection. **Engineering Applications of Computational Fluid Mechanics 16**, 1 (Dec. 2022), 1567–1586. doi:10.1080/19942060.2022.2104930. 2, 3, 4, 8, 9, 13

[Dev18] DEVLIN J.: Bert: Pre-training of deep bidirectional transformers for language understanding. **arXiv preprint arXiv:1810.04805** (2018). 4

[dSZS*23] DE SILVA A., ZHAO M., STEWART D., HASAN F., DUSEK G., DAVIS J., PANG A.: Ripviz: Finding rip currents by learning pathline behavior. **IEEE Transactions on Visualization and Computer Graphics** (2023). 3

[dTG*24] DE SILVA A., TEE N., GHANEKAR O., KHAN F. H., DUSEK G., DAVIS J., PANG A.: Vortexviz: Finding vortex boundaries by learning from particle trajectories, Apr. 2024. arXiv:2404.01352, doi:10.48550/arXiv.2404.01352. 2, 3, 4, 8, 9, 13

[DWL*19] DENG L., WANG Y., LIU Y., WANG F., LI S., LIU J.: A cnn-based vortex identification method. **Journal of Visualization 22**, 1 (Feb. 2019), 65–78. doi:10.1007/s12650-018-0523-1. 2, 3, 4, 8

[FPS*08] FUCHS R., PEIKERT R., SADLO F., ALSALLAKH B., GRÖLLER M. E.: Delocalized unsteady vortex region detectors. In **VMV** (2008), vol. 8, pp. 81–90. 3

[FRM*18] FRANZ K., ROSCHER R., MILIOTO A., WENZEL S., KUSCHE J.: Ocean eddy identification and tracking using neural networks. In **Igarss 2018-2018 IEEE international geoscience and remote sensing symposium** (2018), IEEE, pp. 6887–6890. 3

[GCL*21] GUO M.-H., CAI J.-X., LIU Z.-N., MU T.-J., MARTIN R. R., HU S.-M.: Pct: Point cloud transformer. **Computational Visual Media 7** (2021), 187–199. 4

[GGT17] GÜNTHER T., GROSS M., THEISEL H.: Generic objective vortices for flow visualization. **ACM Transactions on Graphics** (2017). doi:10.1145/3072959.3073684. 2, 3, 13

[GST16] GÜNTHER T., SCHULZE M., THEISEL H.: Rotation invariant vortices for flow visualization. **IEEE TVCG 22**, 1 (2016), 817–826. 5, 9, 12

[GT18] GÜNTHER T., THEISEL H.: The State of the Art in Vortex Extraction. **Computer Graphics Forum 37**, 6 (2018), 149–173. 2

[GT24] GÜNTHER T., THEISEL H.: Objective lagrangian vortex cores and their visual representation. **IEEE Transactions on Visualization and Computer Graphics** (2024), to appear. 3

[Hal05] HALLER G.: An objective definition of a vortex. **Journal of Fluid Mechanics 525** (2005), 1–26. doi:10.1017/S0022112004002526. 2, 3

[HHFH16] HALLER G., HADJIGHASEM A., FARAZMAND M., HUHN F.: Defining coherent vortices objectively from the vorticity. **Journal of Fluid Mechanics 795** (2016), 136–173. 3

[HMTR] HADWIGER M., MLEJNEK M., THEUSSL T., RAUTEK P.: Time-dependent flow seen through approximate observer killing fields. 1257–1266. 2, 3

[HTW18] HAN J., TAO J., WANG C.: Flownet: A deep learning framework for clustering and selection of streamlines and stream surfaces. **IEEE transactions on visualization and computer graphics 26**, 4 (2018), 1732–1744. 3

[Hun87] HUNT J.: Vorticity and vortex dynamics in complex turbulent flows. **Transactions of the Canadian Society for Mechanical Engineering 11**, 1 (1987), 21–35. 2

[JH95] JEONG J., HUSSAIN F.: On the identification of a vortex. **Journal of Fluid Mechanics 285** (1995), 69–94. 2

[KG19] KIM B., GÜNTHER T.: Robust reference frame extraction from unsteady 2d vector fields with convolutional neural networks. **Computer Graphics Forum 38**, 3 (2019), 285–295. doi:10.1111/cgf.13689. 3, 4, 5, 12

[Kin14] KINGMA D. P.: Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980** (2014). 13

[KPH22] KASZÁS B., PEDERGNANA T., HALLER G.: The objective deformation component of a velocity field. **European Journal of Mechanics - B/Fluids** (2022). URL: https://www.sciencedirect.com/science/article/pii/S0997754622001571, doi:10.1016/j.euromechflu.2022.12.007. 3

[LLW*19] LIU Y., LU Y., WANG Y., SUN D., DENG L., WANG F., LEI Y.: A cnn-based shock detection method in flow visualization. **Computers & Fluids 184** (2019), 1–9. 3

[LMTG19] LI G., MULLER M., THABET A., GHANEM B.: Deepgcns: Can gcns go as deep as cnns? In **Proceedings of the IEEE/CVF international conference on computer vision** (2019), pp. 9267–9276. 4

[LS18] LANDRIEU L., SIMONOVSKY M.: Large-scale point cloud semantic segmentation with superpoint graphs. In **Proceedings of the IEEE conference on computer vision and pattern recognition** (2018), pp. 4558–4567. 3

[LSF*18] LGUENSAT R., SUN M., FABLET R., TANDEO P., MASON E., CHEN G.: Eddynet: A deep neural network for pixel-wise classification of oceanic eddies. In **IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium** (2018), IEEE, pp. 1764–1767. 3

[Pop04] POPINET S.: Free computational fluid dynamics. **ClusterWorld 2**, 6 (2004), 7. 5, 9, 12

[QSMG17] QI C. R., SU H., MO K., GUIBAS L. J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In **Proceedings of the IEEE conference on computer vision and pattern recognition** (2017), pp. 652–660. 3

[QYSG17] QI C. R., YI L., SU H., GUIBAS L. J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. **Advances in neural information processing systems 30** (2017). 3

[RG19] ROJO I. B., GÜNTHER T.: Vector field topology of time-dependent flows in a steady reference frame. **IEEE Transactions on Visualization and Computer Graphics 26**, 1 (2019), 280–290. 9

[RMB*21] RAUTEK P., MLEJNEK M., BEYER J., TROIDL J., PFISTER H., THEUSSL T., HADWIGER M.: Objective observer-relative flow visualization in curved spaces for unsteady 2D geophysical flows. **IEEE Transactions on Visualization and Computer Graphics (Proceedings IEEE Scientific Visualization 2020) 27**, 2 (2021), 283–293. doi:10.1109/TVCG.2020.3030454. 3

[RRL23] ROBERT D., RAGUET H., LANDRIEU L.: Efficient 3d semantic segmentation with superpoint transformer. In **Proceedings of the IEEE/CVF International Conference on Computer Vision** (2023), pp. 17195–17204. 4

[RZW*23] RAUTEK P., ZHANG X., WOSCHIZKA B., THEUSSL T., HADWIGER M.: Vortex lens: Interactive vortex core line extraction using observed line integral convolution. **IEEE Transactions on Visualization and Computer Graphics** (2023). 2, 3

[SFYT18] SHEN Y., FENG C., YANG Y., TIAN D.: Mining point cloud local structures by kernel correlation and graph pooling. In **Proceedings of the IEEE conference on computer vision and pattern recognition** (2018), pp. 4548–4557. 3

[SH95] SUJUDI D., HAIMES R.: Identification of swirling flow in 3-d vector fields. In **Proceedings of the 12th Computational Fluid Dynamics Conference** (1995), pp. 792–799. doi:https://doi.org/10.2514/6.1995-1715. 3

[SLM05] SHADDEN S. C., LEKIEN F., MARSDEN J. E.: Definition and properties of lagrangian coherent structures from finite-time lyapunov exponents in two-dimensional aperiodic flows. **Physica D: Nonlinear Phenomena 212**, 3-4 (2005), 271–304. 9

[Vas17] VASWANI A.: Attention is all you need. **Advances in Neural Information Processing Systems** (2017). 4

[VKM91] VATISTAS G. H., KOZEL V., MIH W.: A simpler model for concentrated vortices. **Experiments in Fluids 11** (1991), 73–76. 4

[WDY*21] WANG Y., DENG L., YANG Z., ZHAO D., WANG F.: A rapid vortex identification method using fully convolutional segmentation network. **The Visual Computer 37**, 2 (2021), 261–273. 3

[WFB*19] WU F., FAN A., BAEVSKI A., DAUPHIN Y. N., AULI M.: Pay less attention with lightweight and dynamic convolutions. **arXiv preprint arXiv:1901.10430** (2019). 4

[WJW*24] WU X., JIANG L., WANG P.-S., LIU Z., LIU X., QIAO Y., OUYANG W., HE T., ZHAO H.: Point transformer v3: Simpler faster stronger. In **Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition** (2024), pp. 4840–4851. 4

[WLJ*22] WU X., LAO Y., JIANG L., LIU X., ZHAO H.: Point transformer v2: Grouped vector attention and partition-based pooling. **Advances in Neural Information Processing Systems 35** (2022), 33330–33342. 4

[WSL*19] WANG Y., SUN Y., LIU Z., SARMA S. E., BRONSTEIN M. M., SOLOMON J. M.: Dynamic graph cnn for learning on point clouds. **ACM Transactions on Graphics (tog) 38**, 5 (2019), 1–12. 4

[WSTH07] WEINKAUF T., SAHNER J., THEISEL H., HEGE H.-C.: Cores of swirling particle motion in unsteady flows. **IEEE Transactions on Visualization and Computer Graphics 13**, 6 (2007), 1759–1766. 3

[WT10] WEINKAUF T., THEISEL H.: Streak lines as tangent curves of a derived vector field. **IEEE TVCG 16**, 6 (2010), 1225–1234. 8

[YGX*23] YANG Y.-Q., GUO Y.-X., XIONG J.-Y., LIU Y., PAN H., WANG P.-S., TONG X., GUO B.: Swin3d: A pretrained transformer backbone for 3d indoor scene understanding. **arXiv preprint arXiv:2304.06906** (2023). 4

[ZDM*14] ZHANG L., DENG Q., MACHIRAJU R., RANGARAJAN A., THOMPSON D., WALTERS D. K., SHEN H.-W.: Boosting techniques for physics-based vortex detection. **Computer Graphics Forum 33**, 1 (2014), 282–293. doi:10.1111/cgf.12275. 2, 3

[ZHTR22] ZHANG X., HADWIGER M., THEUSSL T., RAUTEK P.: Interactive exploration of physically-observable objective vortices in unsteady 2d flow. **IEEE Transactions on Visualization and Computer Graphics 28**, 1 (2022), 281–290. doi:10.1109/TVCG.2021.3115565. 3, 5

[ZJFJ19] ZHAO H., JIANG L., FU C.-W., JIA J.: Pointweb: Enhancing local neighborhood features for point cloud processing. In **Proceedings of the IEEE/CVF conference on computer vision and pattern recognition** (2019), pp. 5565–5573. 4

[ZJJ*21] ZHAO H., JIANG L., JIA J., TORR P. H., KOLTUN V.: Point transformer. In **Proceedings of the IEEE/CVF International Conference on Computer Vision** (2021), pp. 16259–16268. 2, 4, 5, 11