Extracting, Tracking, and Visualizing Magnetic Flux Vortices in 3D Complex-Valued Superconductor Simulation Data

Hanqi Guo, Carolyn L. Phillips, Tom Peterka, Dmitry Karpeyev, and Andreas Glatz

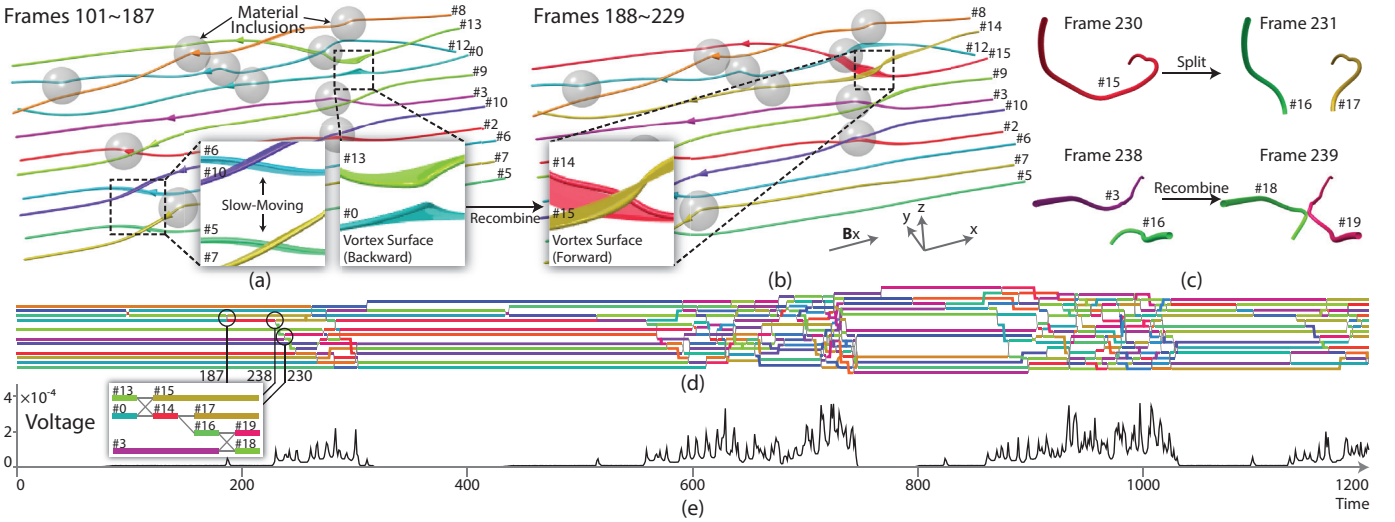

Fig. 1. Visualization of a superconductor in a periodic dissipative state (`Unstable_BX` dataset) based on vortex extraction and tracking results: (a) vortex lines at frame 187; (backward) vortex surfaces showing the trajectories of vortex lines from frame 101-187; (b) vortices #0 and #13 recombine into #14 and #15 at frame 188; (forward) vortex surfaces showing the trajectories from frame 188-229; (c) splitting and recombination events at frames 231 and 239; (d) event diagram; (e) voltage chart. The arrows on the vortex lines indicate their chiralities.

Abstract—We propose a method for the vortex extraction and tracking of superconducting magnetic flux vortices for both structured and unstructured mesh data. In the Ginzburg-Landau theory, magnetic flux vortices are well-defined features in a complex-valued order parameter field, and their dynamics determine electromagnetic properties in type-II superconductors. Our method represents each vortex line (a 1D curve embedded in 3D space) as a connected graph extracted from the discretized field in both space and time. For a time-varying discrete dataset, our vortex extraction and tracking method is as accurate as the data discretization. We then apply 3D visualization and 2D event diagrams to the extraction and tracking results to help scientists understand vortex dynamics and macroscale superconductor behavior in greater detail than previously possible.

Index Terms—Superconductor, Vortex extraction, Feature tracking, Unstructured grid

1 Introduction

Superconductors, materials that can conduct current without any loss, are used in applications ranging from MRI machines to particle accelerators. Materials scientists are interested in understanding and controlling the complex dynamic properties of superconductors. Designing superconductors that can sustain higher lossless, or critical, currents at higher temperatures could lead to technological advances affecting low-cost power transmission in the electrical grid, computing technology, and improved electromagnets.

In the Ginzburg-Landau theory for superconductivity, the superconducting properties of the material can be expressed in terms of the *order parameter* ψ ($\psi \in \mathbb{C}$), which is a complex-valued scalar field. The value of ψ is related to the density of superconducting components. Singularities of ψ correspond to an extremely important feature of the superconductor, magnetic flux vortices. The dynamics of the vortices fundamentally determine the electromagnetic response of the material. Above a critical level, an externally applied magnetic field penetrates a type-II superconductor in the form of flexible flux tubes. The magnetic flux in the vortex core is surrounded by a circular supercurrent. When the vortices move, the system becomes dissipative; and a finite voltage drop across the system, corresponding to resistance, is observed. Thus, the behavior of the vortices is an important determinant of the performance of the material. Material defects, or so-called inclusions, distributed through the type-II superconductor can trap the vortices, pinning them in place and allowing the material to sustain a higher current.

Extracting, tracking, and visualizing the vortex dynamics in large-scale time-dependent Ginzburg-Landau (TDGL) superconductor simulation data are needed in order to understand the dissipative material behavior and the impact of adding material inclusions. Until recently,

- *Hanqi Guo is with the Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA. E-mail: hguo@anl.gov.*
- *Carolyn L. Phillips is with the Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA. E-mail: cphillips@anl.gov.*
- *Tom Peterka is with the Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA. E-mail: tpeterka@mcs.anl.gov.*
- *Dmitry Karpeyev is with the Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, USA. E-mail:*

Manuscript received 31 Mar. 2015; accepted 1 Aug. 2015; date of publication 20 Aug. 2015; date of current version 25 Oct. 2015. For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org. Digital Object Identifier no. 10.1109/TVCG.2015.2466838

numerical simulations have been limited to 2D [7, 18] or small-scale 3D [9] domains. Now, large-scale 3D simulations have been implemented [28, 12] in which macroscale phenomena can be observed, including collective dynamics of many vortices. Materials and computational scientists have developed two TDGL models for structured and unstructured meshes, respectively.In order to determine how vortex dynamics relate to the macroscopically observable system behaviors in large-scale TDGL simulations, however, new methods are required for extracting and visualizing complex vortex motions. In our work, we demonstrate how the vortex extraction and tracking algorithms, together with visualization applications, can elucidate the details of a periodic dissipative vortex state that emerges under certain field and material sample conditions.

The core part of this work is vortex extraction and tracking in 3D time-varying, complex-valued scalar fields. We define extraction to mean detecting features in a single time frame and tracking to mean correlating and connecting the features over time. In a recent study [25], a vortex extraction algorithm was proposed for a single time frame, yet challenges remain for tracking vortices over time. This topic is related to but fundamentally different from vortex extraction in fluid flow. Although much studied in flow visualization [26, 16], vortices in 3D vector fields are not well defined and have multiple criteria, for example, local extrema of vorticity magnitude [39], and λ_2 [14]. In contrast, the vortex line in the complex-valued scalar field of a superconductor is well defined and is the locus of singularity points that satisfy

$$|\psi| = 0 \text{ and } -\oint_C \nabla\theta \cdot d\mathbf{l} = 2n\pi, \tag{1}$$

where $|\psi|$ and θ are the magnitude and phase angle, respectively, of the order parameter ψ, and C is a simple closed contour in $\mathbb{R}^3$ encircling a vortex, which is small enough to avoid any other singularities of ψ. The nonzero integer n, which is usually ± 1, is defined as the chirality of the vortex line with respect to the contour C. In the rest of the paper, the term vortex means a singularity line in a complex-valued scalar field, unless otherwise noted.

We propose a vortex extraction algorithm for both structured and unstructured mesh TDGL simulations. We generalize the previous work in [25] to extract vorticies in both structured and unstructred mesh data. Singularity points are detected on mesh faces on either a tetrahedron or a hexahedron, and they are further transformed into vortex lines based on mesh connectivities.

We develop a vortex tracking algorithm for identifying the same vortex lines over time. Spatial faces of 3D hexahedral or tetrahedral cells are extruded to space-time virtual prisms that are then checked for intersection by vortices, which indicate the movement of vortices. By exploiting the face connectivities of prisms, the singularities in adjacent time frames can be related. Vortex lines in adjacent frames are thereby sewn together, unless a topological event such as a split, merge, or recombination occurs. In the tracking algorithm, the only assumption is that the simulation output varies smoothly over time. A rigorous proof of correctness of our tracking method is provided, and the experimental results demonstrate the effectiveness and robustness of the proposed algorithm. Compared with other feature tracking algorithms in general, our method is parameter free and guarantees the correctness of the results within the accuracy of the output data. While our method is related to earlier studies on tracking critical points in 2D time-varying vector [36] and tensor fields [35], significant challenges arise in our 3D TDGL data. In vector fields, critical points can be located by searching where velocity vanishes, but vortices in superconductors are localized by checking the phase jump of the contour integral (Eq. 1). The higher dimensionality of the 3D mesh is more complex than in 2D cases. Gauge invariance (discussed in Section 3) also needs to be incorporated in the algorithms for numerical reasons.

In addition, based on the vortex extraction and tracking results, we further apply and develop visualization techniques for interactive data exploration. An event diagram is provided to scientiststhat explains the changes in vortex topology. It also enables locating interesting features over time. The visual design is inspired by Storylines [32]. Important measurements such as voltage are also overlaid in the view as line charts, in order to show the relationship between the vortex events and macroscopic system behaviors. A prototype system is implemented that contains both 3D visualization and event diagrams.

To summarize, the contributions of this paper are as follows:

- A vortex detection algorithm for both structured and unstructured mesh TDGL simulations;
- A vortex tracking algorithm for visualizing and analyzing vortex dynamics and events in the datasets;
- Application of various visualization techniques for analyzing and understanding the results of the vortex detection and tracking.

The remainder of this paper is organized as follows. We summarize related work in Section 2 and then introduce our particular scientific problem and its corresponding simulation data in Section 3. Vortex extraction and tracking algorithms are detailed in Section 4. The visualization tool is presented in Section 5. Results and performance are evaluated in Section 6, followed by discussions in Section 7. In Section 8, we summarize our conclusions and briefly discuss future work.

2 Related Work

We summarize related work on complex-valued scalar field visualization and vortex visualization in fluid flow.

2.1 Complex-valued scalar field visualization

In general, a complex-valued scalar field $\psi : \mathbb{R}^n \to \mathbb{C}$ associates a complex scalar with every point in the space (usually $\mathbb{R}^2$ or $\mathbb{R}^3$). Complex fields are found in various science and engineering domains, for example, quantum mechanics, superconductivity, superfluidity, string theory, acoustics, and optics. Relatively little research, however, has been done to visualize and extract features from such fields. We categorize complex field visualization into two categories: direct and feature/topology-based visualization methods.

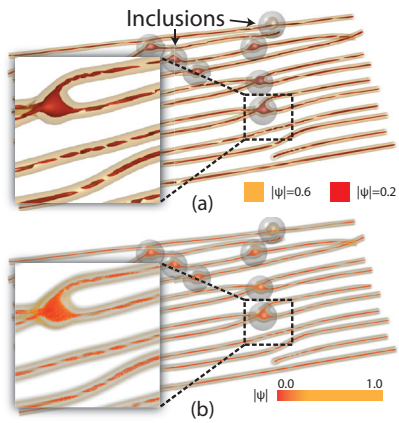

Fig. 2. (a) Isosurface and (b) volume rendering of the order parameter magnitude $|\psi|$. Fine vortex features are usually blurred in the rendering results.

Direct visualization approaches map data values into visual properties so that they can be directly interpreted.Pseudocolor plots, isosurfaces, and contours have been used to visualize quantum physics data [13]. 2D vector field visualization methods, such as arrow plots and line integral convolution (LIC) [6], can be used to visualize complex fields, but their extension to 3D creates occlusion problems. Production tools such as the Application Visualization System (AVS) [23], ParaView [1], and VisIt [8] have been used to facilitate such visualizations. In previous studies of TDGL simulations [18, 12], vortices have been visualized by heat maps or isosurfaces of the magnitude $|\psi|$. These methods are problematic, however, as the datasets become larger and contain more features. Isosurfaces and volume rendering, as in Fig. 2, usually blur the fine features of a vortex and merge when two vortices are in close proximity. Additionally, $|\psi|$ usually is suppressed within inclusions such that vortices cannot be visualized inside them.

An alternative is to use feature- and topology-based visualization to extract important structures and regions in the datasets, for example, valleys and ridges in complex-valued acoustical fields [20]. Vortices correspond to singularities in ψ, isolated points and lines in 2D and

3D, respectively. The extraction of singularity points has been studied in complex optical fields [10, 21]. For TDGL datasets, Phillips et al. [25] proposed a method to extract singularity lines from structured Cartesian grid TDGL simulation data. By interpreting the data as a graph, individual singularity points are connected to form lines. We extend this work with a tracking algorithm for both structured and unstructured mesh data. This algorithm is also related to 2D vector field and tensor field singularity tracking [36, 35, 37], where 2D faces are extended to space-time prisms in order to check whether the features pass through the boundaries. In our work, however, we must account for application-specific mesh and feature complexities that are present in 3D TDGL simulations such as vortex chirality, gauge transformations, and a 3D mesh complex, as well as the large data scales of our application's datasets.

2.2 Vortex visualization in fluid flow

Vortex visualization is a well-studied topic in flow visualization, and a comprehensive literature review on this topic can be found in [16] and [26]. In general, vortices are characterized by swirling motion centers in fluid flow; however, this process can be mathematically defined in various ways. Several definitions have been proposed, such as extrema of λ_2 [14, 30], vorticity magnitude [39], and acceleration magnitude [17].

The two main vortex extraction approaches in fluid flow are region based and line based, which extract the vortices as regions and core lines, respectively. In region-based methods, vortices are located by thresholding the aforementioned characteristic scalar fields. Using related ideas, Jiang et al. [15] proposed a swirling region detection method based on combinatorial topology, and Otto and Theisel [22] introduced uncertain vortex regions. In comparison, in line-based methods, the center of vortical motions are extracted as lines. Most line-based approaches can be generalized as extracting parallel vector descriptors from a dataset [24]. For example, a vortex core line can be defined by a locus of points, where velocity is parallel to vorticity [27, 31], or as the extrema lines of pressures, where pressure gradient is parallel to vorticity [2]. Alternatively, predictor-corrector methods [3] can be used to locate vortex lines in an iterative manner.

Vortex tracking is necessary in order to analyze unsteady flow fields. For vortex regions, the same techniques used in scalar field feature tracking can be used, such as scale-space methods [4] and connected regions [29]. For vortex core lines, Theisel et al. [33] proposed parallel vector surfaces to track them in space-time, which is built on the feature flow fields (FFF) framework [34]. Similar techniques are used to track the vortices as cores of swirling motions [38].

An essential difference exists between fluid flow vortices and vortices in a superconductor. The former are a classical mechanics phenomenon, while the latter are an emergent quantum physics phenomenon. Vortices in fluid flow are usually extracted by a local operator (e.g., extrema or parallel vectors), but vortices in superconductors are localized by the contour integral (Eq. 1), which is a non-local operator. Unlike fluid flow, we cannot determine whether a point is on a vortex line by merely checking whether $|\psi| = 0$. Thus, local operator based vortex extraction and tracking frameworks in fluid flow, such as parallel vectors [24] and FFF [34], cannot be directly used for our problem. In this paper, we extract vortices and their movements by examining phase jumps over mesh faces and space-time edges and then connect them with a graph-based algorithm.

3 TDGL Superconductor Simulation Data

In this section we introduce the properties of TDGL superconductor simulation data. The notation used in this paper is listed in Table 1.

The input data in our study comes from two TDGL implementations, *GLGPU* and *Condor*. The differences between the two simulation models are listed in Table 2. GLGPU is based on a structured rectangular grid, using general-purpose graphics hardware. Currently, the spatial size of problem that can be modeled with GLGPU is limited by the amount of GPU memory; however, the total amount of the time-varying output data can be arbitrarily large. The output of GLGPU is

Table 1. Table of notation.

Symbol	Meaning
ψ	Complex-valued order parameter field
$\|\psi\|, \theta$	Magnitude and phase angle of ψ
$\mathbf{A}$	Magnetic vector potential
$\mathbf{B}$	Magnetic field (curl of $\mathbf{A}$, $\mathbf{B} = \nabla \times \mathbf{A}$)
∇	Gradient operator in 3D, $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})^T$
$\hat{\nabla}$	Gradient operator in 4D, $\hat{\nabla} = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}, \frac{\partial}{\partial t})^T$
$\mathcal{E}, \mathcal{F}, \mathcal{C}$	Sets of mesh edges, faces, and cells
$\mathcal{L}_{ee}, \mathcal{L}_{ef}, \mathcal{L}_{ff}$	Edge-edge, edge-face, and face-face links in mesh graphs
$\mathcal{F}_p^i$	Set of punctured faces in time frame i
$\mathcal{E}_p^{i,i+1}$	Set of intersected edges between time frame i and $i+1$
$\mathcal{F}_p$	Set of $\mathcal{F}_p^i$ for all time frames
$\mathcal{E}_p$	Set of $\mathcal{E}_p^{i,i+1}$ for all time frames
$\mathcal{V}^i = \{V_k^i\}$	Set of vortex graphs in time frame i
$\mathcal{V}_{(k)} = \{V_{(k)}^i\}$	Tracked vortex graphs with the global ID k over time
$\mathcal{S}^{i_0;i_1} = \{S_k^{i_0;i_1}\}$	Set of vortex sheet graphs from time frame i_0 to i_1

Table 2. Comparison between GLGPU and Condor simulation models.

	GLGPU	Condor
Mesh	2D/3D Regular Cartesian	3D Unstructured
Mesh elements	Hexahedra	Tetrahedra
Discretization	Finite-difference	Finite-element
Parallelism	GPU	MPI & libMesh on CPUs
Data format	BDAT	ExodusII
Magnetic vector potential	Analytically defined	Vertex values

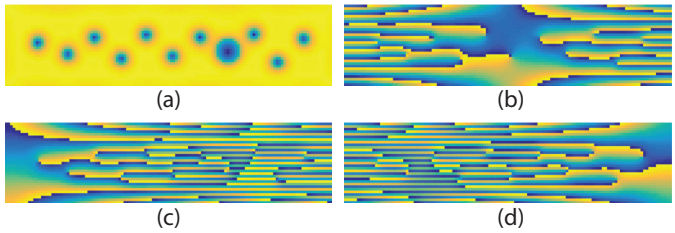

Fig. 3. Examples of gauge transformation: (a) magnitude $|\psi|$, (b) phase field θ in the input data, (c) gauge-transformed phase field $\tilde{\theta}$, (d) another gauge-transformed phase field $\tilde{\theta}'$.

custom-formatted binary files (BDAT). Condor is based on an unstructured mesh, using MPI-based parallelism in supercomputers. It is built on the libMesh [19] finite-element library, and tetrahedra are the basic mesh elements in the grid. The mesh as well as the Condor solution data is stored in the ExodusII[1] format. LibMesh provides a high-level API to load mesh and solutions into memory.

The discretizations of the GLGPU and Condor grids are finite difference and finite element, respectively. In GLGPU data, values inside the mesh elements are estimated by trilinear interpolation. In Condor, piecewise linear approximation is used in the finite-element simulation.

In general, the numerical solution of TDGL models is the tuple $(\psi, \mathbf{A})$, where ψ is the order parameter, and $\mathbf{A}$ is the magnetic vector potential. Both variables are used in our analysis. The tuple $(\psi, \mathbf{A})$ is not a unique description of the state of the superconductor but rather one of an infinite set of tuples that constitute the *gauge freedom* of the solution. Any property of the data needs to be measured in a *gauge-invariant* way or calculated in a way that is not dependent on which specific tuple is used. For example, while the magnitude of ψ (Fig. 3(a)) is gauge invariant, the phase of ψ (Fig. 3(b)) is not. The two phase maps shown in Fig. 3(c) and 3(d) are equally legitimate. In order to calculate the phase change around a closed contour in a gauge-invariant manner [25], the calculation should be transformed as follows:

$$\int_L \nabla\theta \cdot d\mathbf{l} = \int_L (\nabla\theta - \mathbf{A}) \cdot d\mathbf{l} + \int_L \mathbf{A} \cdot d\mathbf{l}, \qquad (2)$$

where L is the integral path, θ is the phase angle of the order parameter ψ, and $\mathbf{A}$ is the magnetic vector potential. When L is a closed

[1] http://sourceforge.net/projects/exodusii/

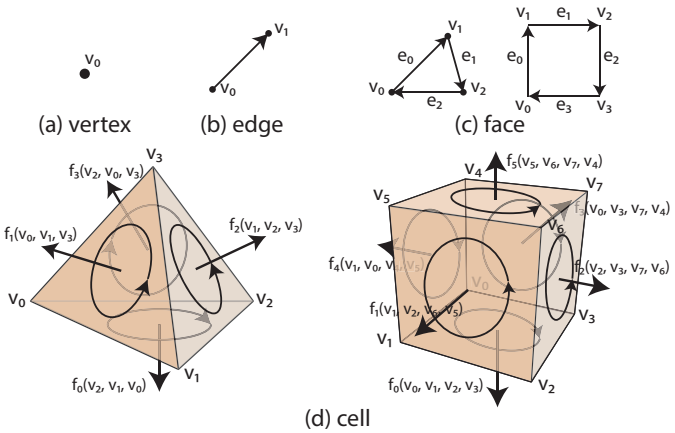

Fig. 4. Mesh elements in TDGL simulations. For convenience, the face normals always point outward. The vertices in each face follow a counterclockwise order.

curve, the Stokes theorem allows the contour integral of $\mathbf{A}$ to be calculated as the flux of magnetic field $\mathbf{B}$, that is, $\int \mathbf{B} \cdot d\mathbf{a}$. For the finite-element-based Condor code, however, the magnetic flux is nontrivial to compute because there are multiple gradients of $\mathbf{A}$ on vertices and edges in the mesh element. Hence, the contour integral is calculated directly instead.

While both codes currently assume a uniform magnetic field ($\mathbf{B}$) in the domain, the value of magnetic vector potential ($\mathbf{A}$) is provided in different ways. For the GLGPU code, the magnetic vector potential is analytically defined. For Condor, in contrast, $\mathbf{A}$ is stored as a value on each vertex in the data file. The latter is in anticipation of implementing nonuniform magnetic fields in the future.

System status and parameters such as voltage V and external current J_{ext} are stored for each time frame. Also, simulation configurations need to be handled correctly in the visualization and analysis. (See [12] for details.) In addition, the locations of material inclusions embedded in the field are also available if they are used in the simulations. Fig. 1 shows the geometries of inclusions as transparent spheres.

For convenience, we define four types of mesh elements in Fig. 4 based on the dimensionality: cells, faces, edges, and vertices. Cells are basic units in the mesh that contain several vertices; they are tetrahedra or hexahedra in our data. Faces are sides of cells, and edges are sides of faces. In the data structures of the next section, edges and faces each are stored just once. Based on the ordering of vertices describing the element, each edge has an implicit direction, and each face has a winding direction determined by the right-hand rule.

4 Graph-Based Vortex Extraction and Tracking

The vortex extraction algorithm locates the vortex lines at individual time frames, and the vortex tracking algorithm correlates these lines across time frames with consistent and unique IDs. Formally, each *vortex line* is an ordered locus of singularity points defined by Equation 1. The tracking results are essentially *vortex surfaces* that contain a locus of vortex lines at different times, as illustrated in Fig. 5. In our graph-based algorithms, vortex lines and vortex surfaces are represented as *vortex graphs* and *(vortex) sheet graphs*, respectively.

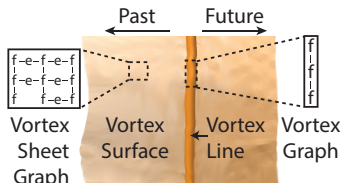

Fig. 5. Vortex line and its vortex surface. The surface is the trajectory of the vortex line over time.

The pipeline of the extraction and tracking algorithms is illustrated in Fig. 6. Both algorithms are based on the mesh discretization used in the simulations. The mesh is presumed to be fine enough to capture

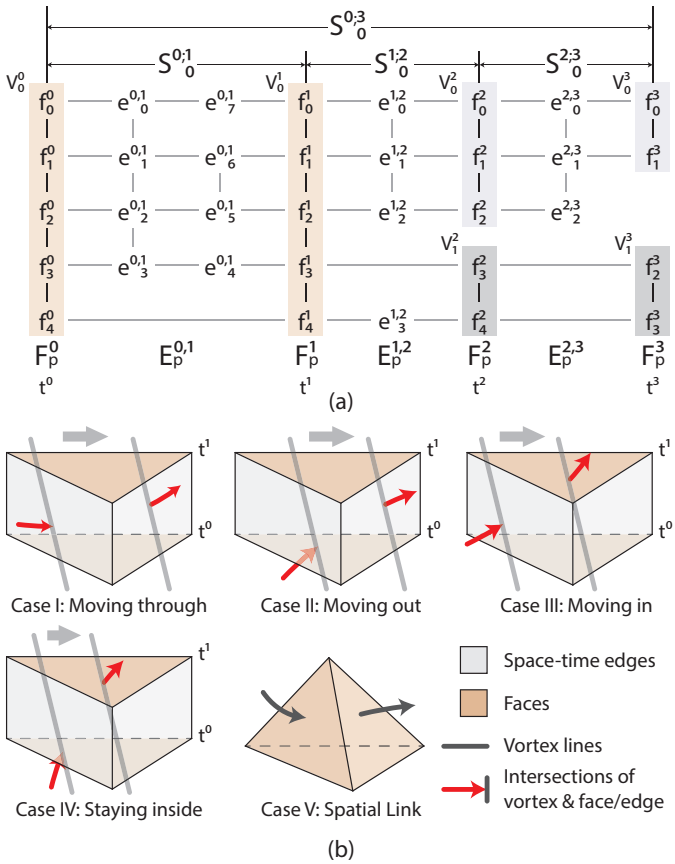

Fig. 7. (a) Illustration of the vortex sheet graph and (b) five types of links in the vortex sheet graph (b).

all important physical phenomena of the TDGL model, and the simulation output is assumed to be continuous between grid points over space and time. A moving (or stationary) singularity is detected by examining mesh elements over time. Specifically, vortices are detected and tracked by examining *spatial faces* and *space-time edges* as follows. At a given time frame, vortices are located by checking each mesh face to see whether it is punctured by a vortex line. The movement of a vortex line is detected by checking each space-time edge to see whether it is intersected at an intermediate time between two adjacent time frames. The *punctured (space) faces* and *intersected (space-time) edges* are stored in $\mathcal{F}_p$ and $\mathcal{E}_p$, respectively, for further analysis.

The key part of the pipeline is the construction of vortex graphs and sheet graphs, which are graph-based discretizations of vortex lines and vortex surfaces. A vortex graph V_k^i connects a subset of punctured faces $\mathcal{F}_p^i$ in time frame i. A set of punctured faces (usually two) are connected if they belong to the same mesh cell. This presumption is based on Lemma 1 (see the Appendix) that a closed volume (a cell in this case) in $\mathbb{R}^3$ has an equal number of vortex entry and exit points. Based on this principle, vortex lines are generated by tracing the connected nodes in a vortex graph. A vortex sheet $S_k^{i_0,i_1}$ connects all punctured faces $\cup_{i_0}^{i_1} \mathcal{F}_p^i$ and intersected edges $\cup_{i_0}^{i_1-1} \mathcal{E}_p^{i,i+1}$ in time period i_0 to i_1 (Fig. 7(a)). Two elements are connected if they are in the same space-time prism, which is a space-time extension of a spatial face f. A connection indicates that a vortex line has intersected the face f over the time interval by moving in, out, or through, or staying inside the face (Fig. 7(b)). Events can be further detected based on the extraction results.

Our analysis is based on the *mesh graph* $\mathcal{G}$ in Fig. 8, whose nodes include all edges and faces in the mesh and whose links are their connections.[2] The following subsections present detailed descriptions of

[2]To avoid ambiguity, we use the terms "links" instead of "edges" and "nodes" instead of "vertices" in graphs.

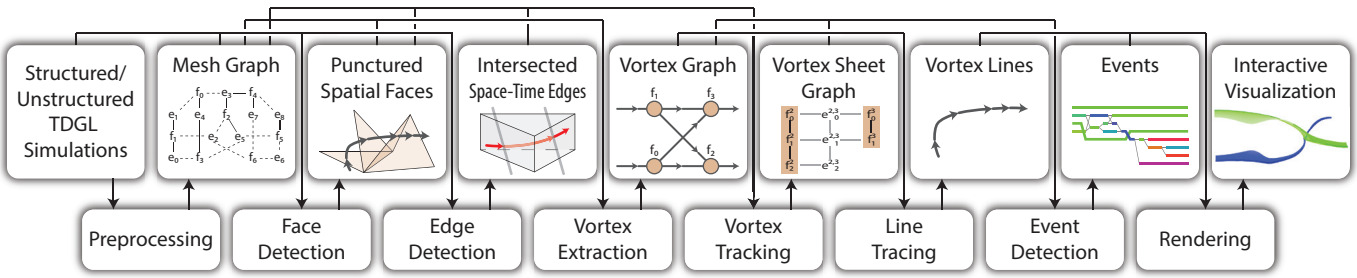

Fig. 6. Pipeline of the graph-based vortex extraction and tracking algorithms. The mesh graph is constructed from the input data, and then punctured faces and intersected edges are extracted. Vortex graphs and vortex sheet graphs are constructed, which are further transformed into vortex lines with global IDs over time. The vertex lines and the events are further interactively visualized.

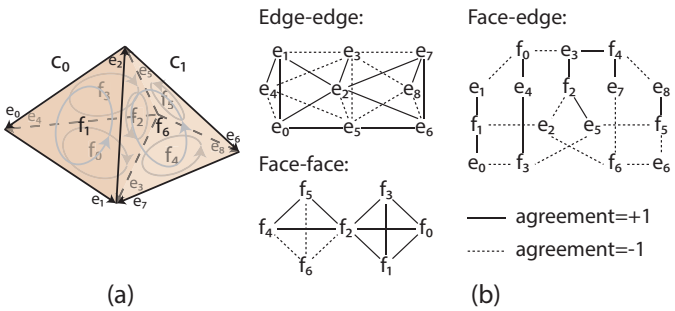

Fig. 8. Illustration of the mesh graph $\mathscr{G}$: (a) an unstructured mesh with two tetrahedra; (b) node-link diagram for $\mathcal{L}_{ee}$, $\mathcal{L}_{ef}$, $\mathcal{L}_{ff}$.

the construction of a mesh graph, detection of punctured faces and intersected edges, and the construction of a vortex graphs and vortex sheet graphs.

4.1 Mesh graph construction

A mesh graph is the basic data structure in our graph-based vortex extraction and tracking algorithms. Fig. 8 illustrates an example of the mesh graph for a tetrahedral mesh consisting of two tetrahedra. Formally, a mesh graph is the undirected and edge-weighted graph $\mathscr{G} = (\mathscr{N}, \mathscr{L})$. The set of nodes $\mathscr{N} = \{\mathcal{E}, \mathcal{F}\}$ is the set of all mesh edges $\mathcal{E}$ and mesh faces $\mathcal{F}$. The set of links $\mathscr{L} = \{\mathcal{L}_{ee}, \mathcal{L}_{ef}, \mathcal{L}_{ff}\}$ is the set of all edge-edge links $\mathcal{L}_{ee}$, edge-face links $\mathcal{L}_{ef}$, and face-face links $\mathcal{L}_{ff}$. Two edges are linked if they belong to the same face; an edge is linked to a face if the edge belongs to face, and two faces are linked if they belong to the same cell. Edges and faces each are stored with an implicit direction. Each link in $\mathscr{L}$ has a weight of $+1$ or -1 indicating directional agreement or disagreement between two connected elements. Directional agreement between faces, for example, means their winding directions both point in or out of the cell. For example, in Fig. 8, the link between edge e_1 and face f_1 has the weight of $+1$, but the link between edge e_7 and face f_6 has the weight of -1.

The mesh graph $\mathscr{G}$ is stored as adjacency lists. Each edge keeps lists of `sibling_edge_ids` and `parent_face_ids`. Their directional agreements are stored in `sibling_edge_agreement` and `parent_face_agreement`, respectively. Similarly, each face keeps lists of `child_edge_ids`, `child_edge_agreement`, `sibling_face_ids`, and `sibling_face_agreement`.

The explicit construction of the mesh graph is necessary only for unstructured meshes. For structured Cartesian meshes, the regular pattern of the mesh makes these data structures implicit and trivial to generate on the fly. For unstructured meshes, however, finite-element libraries such as libMesh provide explicit access only to cells. If iterating over all cells, faces and edges will be repeatedly accessed, leading to an inefficient use of computation resources. Therefore, we first preprocess the mesh and construct the mesh graph. In this way, the parent, child, and sibling mesh element relationships are established once and reused.

4.2 Punctured faces and intersected space-time edges detection

The calculation of closed contour integrals can locate the singularities in the complex field defined over space and time. We use the paths around mesh faces and space-time edges as contours to detect singularities. If and only if a face is intersected by a vortex line in a single time frame, the edges of the face encircle a singularity. Similarly, if and only if a vortex line intersects an edge at an intermediate time between two time frames, a time contour can be constructed that also encircles a singularity. The time contour comprises the edge in space at the two time frames and two edges in time created by extending the end points of the space edge through time. An approximate intersection point or intersection time can be further solved by using interpolation.

To locate intersection points for both faces and space-time edges, we extend Equation 1 to 4D as follows,

$$n = -\frac{1}{2\pi} \oint_C \hat{\nabla}\theta \cdot d\hat{\mathbf{l}}, \tag{3}$$

where C is a space-time contour, $\hat{\nabla}$ is the gradient operator, and $d\hat{\mathbf{l}}$ is the infinitesimal line segment of C. Notice that Equation 3 is equivalent to Equation 1 when the time dimension of C is constant. If n is a nonzero integer (usually ± 1 in our study), the contour C encircles a singularity point. The sign of n is the chiral direction of the singularity with respect to the normal of the contour.

In the discrete case, where contour C is formed by m connected line segments, the contour integral in Equation 3 is broken into a sum of line integrals, which are further converted to the sum of phase shift on each line segment of C:

$$n = -\frac{1}{2\pi} \sum_{i=0}^{m-1} \int_{L_{i,j}} \hat{\nabla}\theta \cdot d\hat{\mathbf{l}} = -\frac{1}{2\pi} \sum_{i=0}^{m-1} \Delta\theta_{i,j}, \tag{4}$$

where $j = (i+1) \mod m$.

$$\Delta\theta_{i,j} = \mod(\theta_j - \theta_i + \pi, 2\pi) - \pi, \tag{5}$$

where θ_i is the phase angle of ψ on the ith vertex of C. The modulo operation maps $\theta_{i,j}$ into the range of $[-\pi, \pi)$. In the TDGL model, for numerical reasons, a gauge transformation (Equation 2) needs to be performed along the contour in order to compute the phase shift between two arbitrary locations in the same time frame. If the time is constant for path $L_{i,j}$, Equation 5 is transformed to

$$\Delta\theta_{i,j} = \mod\left(\theta_j - \theta_i - \int_{L_{i,j}} \mathbf{A}\cdot d\mathbf{l} + \pi, 2\pi\right) + \int_{L_{i,j}} \mathbf{A}\cdot d\mathbf{l} - \pi. \tag{6}$$

In the TDGL equations implemented on a mesh, values along a mesh edge can be estimated by linear interpolation, so we use $\int_{L_{i,j}} \mathbf{A}\cdot d\mathbf{l} = \frac{\mathbf{A}_i + \mathbf{A}_j}{2} \cdot (\mathbf{x}_j - \mathbf{x}_i)$, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are coordinates of vertices i and j, respectively. $\mathbf{A}_i$ and $\mathbf{A}_j$ are the values of the magnetic vector potential, which are known at every mesh vertex.

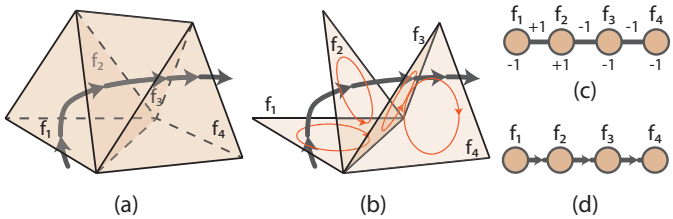

Fig. 9. Illustration of vortex extraction algorithm: (a) punctured cells in the mesh, (b) punctured faces in the mesh, (c) the vortex graph, (d) the vortex line.

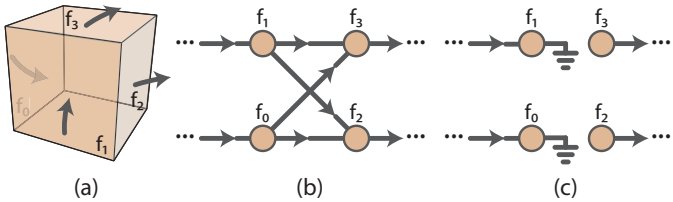

Fig. 10. Rare case creating ambiguity in vortex extraction: (a) two vortices puncture the same cell; (b) the vortex graph of the vortices; (c) the vortex graph is split into four separate lines.

Algorithm 1 The graph-based vortex graph (left) and vortex sheet graph (right) construction

$\mathcal{V}^i \leftarrow \varnothing$
$\mathcal{N}_l \leftarrow \mathcal{F}^p_{t_i}$
$\mathcal{L}_l \leftarrow \mathcal{L}_{ff}$
while not $\mathcal{N}_l$.empty() **do**
  $v \leftarrow$ an arbitrary element in $\mathcal{V}_l$
  $V \leftarrow \text{BFS}(\mathcal{N}_l, \mathcal{L}_l, v)$
  $\mathcal{V}^i$.add(V)
  $\mathcal{N}_l \leftarrow \mathcal{N}_l \setminus V$
end while

$S^{i_0;i_1} \leftarrow \varnothing$
$\mathcal{N}_s \leftarrow \{\cup_{i_0}^{i_1-1}\mathcal{E}_p^{i,i+1}, \cup_{i_0}^{i_1}\mathcal{F}^i_p\}$
$\mathcal{L}_s \leftarrow \{\mathcal{L}_{ee}, \mathcal{L}_{ef}, \mathcal{L}_{ff}\}$
while not $\mathcal{N}_s$.empty() **do**
  $v \leftarrow$ an arbitrary element in $\mathcal{N}_s$
  $S \leftarrow \text{BFS}(\mathcal{N}_s, \text{ADJ}_s, v)$
  $\mathcal{S}^{i_0;i_1}$.add(S)
  $\mathcal{N}_s \leftarrow \mathcal{N}_s \setminus S$
end while

Once a singularity is located on a face or space-time edge, the location where $|\psi| = 0$ can be approximated by solving a system where the real part $\Re(\psi)$ and imaginary part $\Im(\psi)$ simultaneously equal zero. For a linearly interpolated triangular face, the punctured point can be located by solving a linear system; for a bilinearly approximated quadrilateral face, we employ a generalized eigensolver to find the zero point [25]. The sets of intersected faces $\mathcal{F}_p$ and intersected space-time edges $\mathcal{E}_p$ are retained for further analysis.

4.3 Vortex extraction

The vortex extraction algorithm constructs the vortex graph V^i_k, and then the vortex lines are subgraphs of $\mathcal{G}$. Formally, a vortex graph V^i_k is one of the connected components in the graph $\mathcal{V}^i(\mathcal{F}^i_p, \mathcal{L}_{ff})$, where the nodes $\mathcal{F}^i_p$ are the punctured faces in time frame i and $\mathcal{L}_{ff}$ is the set of face-face links in the mesh graph $\mathcal{G}$.

The graph we construct here is the *edge-to-vertex dual graph* of the graph constructed in [25] to simplify certain data structures for unstructured meshes. Thus the meaning of a node and link in our graph is different from that in [25]. The pseudocode of vortex graph construction is listed in Algorithm 1 (left), and the algorithm is illustrated in Fig. 9. The connected components are extracted by either breadth-first search (BFS) or depth-first search (DFS). The time complexity for the search is $O(|\mathcal{F}^i_p| + |\mathcal{L}_{ff}|)$.

The rationale for the vortex line construction is based on Lemma 1 and Lemma 2. For any cell in the mesh, which is a closed volume in $\mathbb{R}^3$, there must be equal numbers of entry and exit points if it is punctured by a vortex line. Our simulation models contain at most one puncture point per face. The likelihood that a puncture point falls exactly on an edge is negligible. Thus, there are always equal numbers of faces of a cell whose outward normals have a positive or negative dot product relative to the chiral direction of the vortex. In most cases, our data has no more than two punctured faces with opposite dot products in a punctured cell, thus ensuring that the two puncture

Algorithm 2 Find adjacent nodes in a vortex sheet graph.

$\mathcal{F}_p = \{\mathcal{F}^0_p, \mathcal{F}^1_p, \ldots, \mathcal{F}^{n-1}_p\}, \mathcal{E}_p = \{\mathcal{E}_p^{0,1}, \mathcal{E}_p^{1,2}, \mathcal{E}_p^{n-2,n-1}\}$
function $\text{ADJ}_S(\mathcal{F}_p, \mathcal{E}_p, \mathcal{G}, v))$
  $A \leftarrow \varnothing$
  if $v \in \mathcal{E}_p$ **then** $e'^{i,i+1} \leftarrow v$ ▷ v is an intersected edge
    for all $e \in \mathcal{L}_{ee}(e'^{i,i+1})$ **do**
      if $e \in \mathcal{E}_p^{i,i+1}$ **then** A.add($e^{i,i+1}$) ▷ Case I eg. $e_0^{0,1} \rightarrow e_1^{0,1}$
      end if
    end for
    for all $f \in \mathcal{L}_{ef}(e'^{i,i+1})$ **do**
      if $f \in \mathcal{F}^i_p$ **then** A.add(f^i) ▷ Case II eg. $e_0^{0,1} \rightarrow f_0^0$
      else if $f \in \mathcal{F}^{i+1}_p$ **then** A.add(f^{i+1}) ▷ Case III eg. $e_7^{0,1} \rightarrow f_0^1$
      end if
    end for
  else $f'^i \leftarrow v$ ▷ v is a punctured face
    for all $e \in \mathcal{L}_{ef}(f'^i)$ **do**
      if $e \in \mathcal{E}_p^{i,i+1}$ **then** A.add($e^{i,i+1}$) ▷ Case II eg. $f_0^1 \rightarrow e_0^{1,2}$
      else if $e \in \mathcal{E}_p^{i-1,i}$ **then** A.add($e^{i,i-1}$) ▷ Case III eg. $f_0^1 \rightarrow e_7^{0,1}$
      end if
    end for
    for all $f \in \mathcal{L}_{ff}(f'^i)$ **do**
      if $f \in \mathcal{F}^{i-1}_p$ **then** A.add(f^{i-1}) ▷ Case IV eg. $f_3^2 \rightarrow f_3^1$
      else if $f \in \mathcal{F}^i_p$ **then** A.add(f^i) ▷ Case V eg. $f_3^2 \rightarrow f_4^2$
      else if $f \in \mathcal{F}^{i+1}_p$ **then** A.add(f^{i+1}) ▷ Case IV eg. $f_3^2 \rightarrow f_2^3$
      end if
    end for
  end if
return A
end function

points belong to the same vortex line. In rare cases (40 out of 9.7 million cells in `Unstable_BX` dataset for all frames), a cell has multiple pairs of single-punctured faces (Fig. 10). Physically this corresponds to two vortices in close proximity, but we cannot distinguish the two vortices from the data. We treat them with the same vortex graph in the algorithm instead.

Vortex lines are constructed by connecting the puncture points, or nodes, in the graph. The order of traced nodes follows the chiral direction of the vortex, following links that connect "ins" (negative dot products) to "outs" (positive dot products). When a cell has more than two puncture points, the traced path from a node can have more than one link to follow. We break the vortex line into multiple lines at this node, but we group all the lines associated with the multiple puncture points of the cell as belonging to the same vortex ID.

4.4 Vortex tracking

Vortex tracking is based on the construction of vortex sheets over time. Formally, a vortex sheet $S^{i_0;i_1}_k$ is a connected component in the graph $\mathcal{S}^{i_0;i_1}(\{\cup_{i_0}^{i_1}\mathcal{F}^i_p, \cup_{i_0}^{i_1-1}\mathcal{E}^i_p\}, \{\mathcal{L}_{ee}, \mathcal{L}_{ef}, \mathcal{L}_{ff}\})$. The nodes in this graph are punctured faces $f^i \in \mathcal{F}^i_p$ and intersected edges $e^i \in \mathcal{E}^i_p$ in the time period $i_0 \le i \le i_1$. Additional rules apply for the links in this graph, which are detailed in Algorithm 2.

The pseudocode for the construction of a vortex sheet appears in Algorithm 1 (right). The basic idea is to partition the elements in $\mathcal{F}^p$ and $\mathcal{E}^p$ into a set of connected components and label them with different IDs for further processing. We do so by finding each connected component in the graph $\mathcal{S}^{i_0;i_1}$ by BFS. The time complexity of vortex sheet construction is $O(\sum_{i_0}^{i_1}|\mathcal{F}^i_p| + \sum_{i_0}^{i_1-1}|\mathcal{E}^{i,i+1}_p|)$.

The correctness of the vortex sheet construction algorithm is founded on Lemma 3 and Lemma 4. Similar to a volume in $\mathbb{R}^3$, a space-time volume in $\mathbb{R}^2 \times \mathbb{R}$ will also have two punctured "faces." The faces in the prism can be faces in the two time frames, or the "virtual faces," that is, space-time edges. A $2n\pi$ phase jump in a virtual face means that the vortex line intersected the edge at an intermediate time between two time frames.

Detecting punctured faces and intersected edges results in five types of links in our graph model, corresponding to the five cases in Fig. 7(b). If an intersected edge $e^{i,i+1}$ has a link to another edge in $\mathcal{E}^{i,i+1}_p$, then a vortex line passes through the parent face of the both

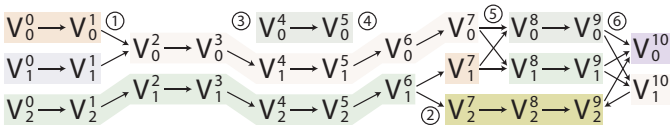

Fig. 11. Vortex graph and events: ① merging, ② splitting, ③ birth, ④ death, ⑤ recombination/crossing, and ⑥ compound.

edges without intersecting the face in either time frame (Case I). If an intersected edge $e^{i,i+1}$ connects to a punctured face, then the intersection of the vortex line has moved in or out of the face (Cases II and III). If a face f^i links to the same face in adjacent time frames (f^{i-1} or f^{i+1}), then the intersection of the vortex line remains inside this face (Case IV). Case V corresponds to the link between two faces in the same time frame. These links are identical to those generated by vortex extraction in Section 4.3 except that they are undirected.

The construction of a vortex sheet generates a simple interpretation of the continuity of a vortex line across time frames. If for each $i, (i_0 \le i \le i_1)$ there exists exactly one k_i such that $V^i_{k_i} \subset S^{i_0,i_1}_k$, then the set of vortex graphs $\{V^i_{k_i}\}$ describes the same vortex line over this time period. The tracked vortices are denoted as $\mathcal{V}_{(k)} = \{V(k)^i\}$, where k is the unique ID of the vortex over time. Some event must have occurred if these criteria are not met. Various types of events, interpreted from changes in the topology of the vortex sheets, are explained in the following subsection.

In practice, instead of loading all time frames and generating the sheet graphs $\mathcal{S}^{0;n-1}$ for all n, we need only to compute sheet graphs $\mathcal{S}^{i,i+1}$ for adjacent time frames. Thus, our out-of-core implementation requires only two adjacent time frames in memory.

4.5 Event Detection

The vortex tracking algorithm generates vortex sheets and labels the vortex lines with unique IDs if a group of lines have no topological changes over the time. Otherwise, an event must have occurred. We describe here a graph-based event detection method to help scientists understand the changing topology of vortex lines.

The event detection is based on the *vortex event graph* illustrated in Fig. 11. Formally, the event graph is defined as $\mathcal{G}_e = (\{V^i_k\}, \mathcal{L}_e)$, where the nodes are the vortex graphs in all time frames and the links, connecting one vortex graph to another, $\mathcal{L}_e$ are indirectly defined by vortex sheets. $\mathcal{L}_e(V^i_k, V^{i+1}_{k'})$ exists if and only if for $i_0 \le i \le i_1 - 1$, $V^i_k \subset S^{i_0,i_1}_{k''}$ and $V^{i+1}_{k'} \subset S^{i_0,i_1}_{k''}$. In other words, two vortex graphs in adjacent time frames are connected if and only if both vortex graphs are subsets of the same vortex sheet. To minimize the memory footprint, we use vortex sheets of only two adjacent time frames in our implementation. $\mathcal{L}_e$ is stored as a series of adjacency matrices for each such time interval.

We define several basic types of events: birth, death, splitting, merging, and recombining/crossing, as exemplified in Fig. 11. Notice that the term recombination could also be called bifurcation, yet recombination is more frequently used in the study of superconductors [5]. If an event is none of the above five types, we call it a compound event. A compound event occurs if multiple splitting and merging events are in the same time interval. Technically, we cannot reconstruct the sequence of "subevents" in the compound event using the graph-based definition. Reconstructing the subevents requires a finer time discretization.

5 Visualization Tool

A visualization tool is implemented for scientists to explore TDGL datasets. The tool has two design goals: to enable users to see how vortices are distributed in space and vary over time, and to investigate how vortex dynamics (events) and energy dissipation (voltages) are related. Hence, we designed the tool with two components (shown in Fig. 1): the spatial view and the event view. The former provides the interactive 3D visualization for vortices, and the latter visualizes the events and voltages in a 2D view.

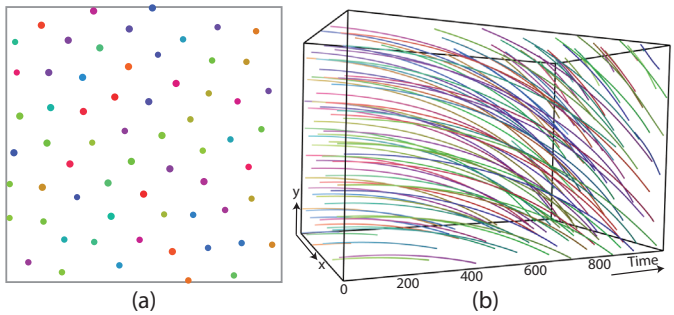

Fig. 12. 2D test data: (a) vortex cores of the first time frame (b) the vortex tracking results. The lines are the trajectories of the vortices.

5.1 Spatial View

The spatial view provides a full set of functionalities for 3D interactive visualization of both raw data and extracted vorticies. For the raw TDGL data, the order parameter magnitude can be visualized with volume rendering and isosurfaces, and the data slices can be rendered with pseudo colors. Vortices and their trajectories are visualized with tubes and surfaces, respectively. Arrows rendered on vortex lines indicate the chirality of the vorticies. Vortex surfaces are constructed by triangulating discrete points on vortex lines in adjacent frames. Together with the event diagram described in the next section, the spatial view can help scientists explore and discover key features in TDGL simulation data.

5.2 Event View

A storyline presentation [32] of the vortex event graph is provided in the event view, as shown in Fig. 1(d). Together with the line chart showing the voltages, the visualization provides an overview of vortex dynamics and their correlation with the energy dissipation. In the event view, each colored line represents a tracked vortex. The length of the line encodes the duration of the vortex. Short gray lines connect a group of vortices if they are involved in an event. Zooming allows users to browse details in a long time sequence.

The graph layout is based on the `dot` algorithm in the Graphviz library [11]. Rendering is based on Scalable Vector Graphics (SVG). The event graph $\mathcal{G}_e = (\{V^i_k\}, \mathcal{L}_e)$ is transferred to `dot` and then transformed into a layout. Graphviz ensures the minimum numbers of crossings in the layout. Nodes in the same time frames are set to the same rank in `dot` input in order to align the nodes to time frames. To keep long vortices as straight as possible in the layout, we assign higher weights to their corresponding links in the graph. Color schemes are automatically generated in order to avoid ambiguities in both the graph and 3D visualizations according to two constraints: (1) vortices in the same time frame cannot have the same colors, and (2) vortices involved in the same event cannot have the same colors. The graph layout and the color schemes are transformed into the SVG format, which can be easily rendered in the event view.

6 Results

We present three application cases of our algorithms. The specifications of the datasets and the performance are listed in Table 3. The timings are tested on a workstation with a Intel Xeon E5620 CPU (2.40 GHz) and 12 GB main memory. The implementation is C++. The code simultaneously supports the analysis of both structured and unstructured TDGL output data. The libMesh library [19] is used to manage the mesh and data I/O for the Condor unstructured TDGL simulation output. Because the same underlying finite-element framework is used for both the simulation and analysis, our implementation is compatible with future in situ parallel execution. The intermediate data products in the pipeline—mesh graphs, punctured faces, and intersected edges—are serialized and stored in files for data reuse in further analysis.

Table 3. Data specifications and the timings. n_t is the number of time frames. n_v, $|\mathcal{E}|$, $|\mathcal{F}|$, $|\mathcal{C}|$ are numbers of edges, faces and cells, respectively. $|\mathcal{L}_{ee}|$, $|\mathcal{L}_{ef}|$, and $|\mathcal{L}_{ff}|$ are the number of links in the mesh graph. $|\mathcal{F}_p|$ and $|\mathcal{E}_p|$ are the total numbers of punctured faces and intersected edges in all time frames, respectively. T_{pre}, T_{io}, T_{pe}, T_{pf}, and T_g are the timings for preprocessing, I/O, punctured face detection, intersected edge detection, and vortex/sheet graph construction, respectively. I/O takes longer for `tslab` data because of the extra overhead in libMesh.

Name	Mesh	Resolution	n_t	Size	n_v	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{F}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{L}_{ee}\rvert$	$\lvert\mathcal{L}_{ef}\rvert$	$\lvert\mathcal{L}_{ff}\rvert$	$\lvert\mathcal{F}_p\rvert$	$\lvert\mathcal{E}_p\rvert$	T_{pre}	T_{io}	T_{pf}	T_{pe}	T_g
Test2D	2D Cartesian	128^2	1K	128MB	16.4K	32.8K	16.4K	16.4K	98.4K	16.4K	N/A	64.9K	5.6K	N/A	$1.5s$	$42.4s$	$53.6s$	$0.4s$
Unstable_BX	3D Cartesian	$256 \times 128 \times 32$	3K	24GB	1.0M	3.1M	3.1M	1.0M	18.6M	6.2M	15.5M	9.7M	0.6M	N/A	$24.1s$	$3.3hr$	$2.1hr$	$236.7s$
tslab	3D Unstructred	N/A	1K	40GB	0.5M	3.7M	6.3M	3.6M	9.5M	9.5M	18.9M	3.4M	0.2M	$183s$	$1.9hr$	$1.9hr$	$1.1hr$	$10.9s$

6.1 2D structured mesh data

A 2D structured mesh dataset, `Test2D` (Fig. 12), was generated with the GLGPU simulation for testing the vortex tracking algorithm. 2D vortices are points. In the simulation, the external current increases over time; thus, the vortices are moving downward in an accelerating manner. The data has periodic boundary conditions in both the x and y direction. When a vortex crosses a periodic boundary, it reappears on the opposite side of the box. In each time frame, 65 vortices are detected, and no events occur over the time sequence. The rendering results and the supplementary video show that each vortex is consistently tracked over time.

6.2 3D structured mesh data

In Fig. 1 and the supplementary videos, we visualize the vortex dynamics of a superconductor in a periodic dissipative state created by aligning the magnetic field and current in a superconductor with inclusions (dataset `Unstable_BX`). From the event diagram and the voltage chart, we observe a strong correlation between the occurrence of events and the energy dissipation. The system oscillates between a slowly evolving, nearly stable state and a rapidly rearranging, unstable state.

Vortices are stretched in the direction of the magnetic field. Attracted to material inclusions, vortices bend to pin themselves on a nearby inclusion. If the vortices were perfectly straight in the x-direction, the external current applied along the x-axis would impose no force on them. The bending of the vortices induces a slight Lorentz force from the current, pushing the vortex along the y-axis. In frames 101 to 187, the bent part of each vortex slowly deforms in the y-direction (see vortex surfaces of #5, #6, #7, and #10 in frames 101 to 187). At frame 187, vortices #0 and #3 swap parts to create two new vortices #14 and #15 at frame 188. The motions before and after this event occur extremely fast, as shown by the vortex surfaces.

In addition to the 3D visualization of vortices, scientists would also like to see an overview of all events and investigate how these events are related to voltage changes. From the event diagram, we can see that the vortices rapidly topologically reconfigure after frame 230. At frame 231, vortex #15 bends to the boundary and splits into two new vortices #16 and #17, each attached to the boundary of the system. The two ends of the vortex attached to the boundary are now pushed helically around the system in opposite directions by the Lorentz force. The two vortex ends travel along a cross-section of the system where several inclusions are roughly aligned in the y-direction, and a series of recombination events along this plane is observed in subsequent frames. The two vortex ends meet and join again, and the system relaxes back into a quiescent state after frame 330. The line chart below the event diagram depicts the voltage spikes observed over the cycle. The intense voltage spike occurs when the ends of the vortex attach to the slab surface and are driven helically around the system.

6.3 3D unstructured mesh data

in Fig. 13 and the supplementary video, we show vortex extraction and tracking for the `tslab` dataset generated from an unstructured mesh consisting of 3.6M cells. The event diagram at the bottom of Fig. 13 begins shortly after the simulation is initialized, when the vortices are still relaxing into a low-energy configuration, corresponding to many events over the first 50 frames. The vortex surfaces shown in the upper left of Fig. 13 correspond to a time period when no events occur. In the upper right of Fig. 13, we show the vortices extracted at a particular time step.

An event of particular interest to scientists is recombination, defined by two vortices swapping parts. In the upper middle of Fig. 13, we show in greater detail before and after the recombination event at time frame 224. After the event, the lower and upper parts of vortex #39 become the lower and upper parts of #43 and #44, respectively, and the upper and lower parts of vortex #40 become the upper and lower parts of #43 and #44, respectively. The vortex surfaces on the left and right side show how vortices deform before and after recombination. To recombine, the vortices bend to a locally antiparallel configuration that increases their attraction to each other. After the recombination, the vortices, now parallel, rapidly repel each other.

7 Discussion

Vortex extraction and tracking algorithms enable scientific exploration and analysis of superconductor simulation data. Extraction of a vortex allows precise interpretation of a vortex's spatial features, regardless of its proximity to another vortex or an inclusion. Tracking allows each vortex to be identified over time and shows events that change the topology of a vortex. Event diagrams provide a visual abstraction of the events and facilitate interactive exploration. By visualizing the events of the tracking algorithm, the user is provided a quick overview of the entire time sequence. This allows subsequent investigation into the relationship between vortex dynamics and macroscale measures such as energy dissipation. Vortex tracking also enables precise characterization and measurement of local vortex motions.

For a closed volume in $\mathbb{R}^3$ (cells in our algorithms), if there are one entrance point and one exit point, the two points belong to the same vortex line unambiguously. Similarly, we observe how singularity points move over time by calculating phase jumps over space-time edges. Vortex graphs and vortex sheet graphs are constructed by graph-based algorithms, and vortices are tracked over time and labeled with global IDs. Events are detected if vortex lines have topological changes over time.

In general, the robustness of the algorithms is limited only by the resolution of the data. A space or time discretization that is too coarse can generate various artifacts, and scientists would need to rerun the simulation with finer discretization of the mesh in such cases. Since phase changes can be correctly measured only if their magnitude is less than π, if the mesh is too coarse, false positives or false negatives in the punctured face and intersected edge detection will result. We also require that no more than one vortex puncture each face. However, if more than one vortex or vortex sheet punctures the same cell by different faces (Fig. 10), we can handle the case by treating the vortices as a compound object. Similarly, in event detection, if multiple events occur involving the same set of vortices over a time increment, we cannot reconstruct the subevents that create the compound event; but we can identify the presence of a compound event. All the issues can be resolved by increasing the data resolution. A puncture point exactly intersecting a spatial edge of the mesh in a time frame, resulting in zero, two, or three punctured faces instead of one, is a highly unlikely occurrence that we have never observed.

8 Conclusions and Future Work

In this paper, we have presented a framework for vortex extraction and tracking of both structured and unstructured complex-valued TDGL simulation data. By checking singularities on mesh faces and space-time edges, we have shown how to construct vortex graphs and vortex sheet graphs that model the connectivities of singularity points in both space and time at the finest scale the dataset supports. By applying

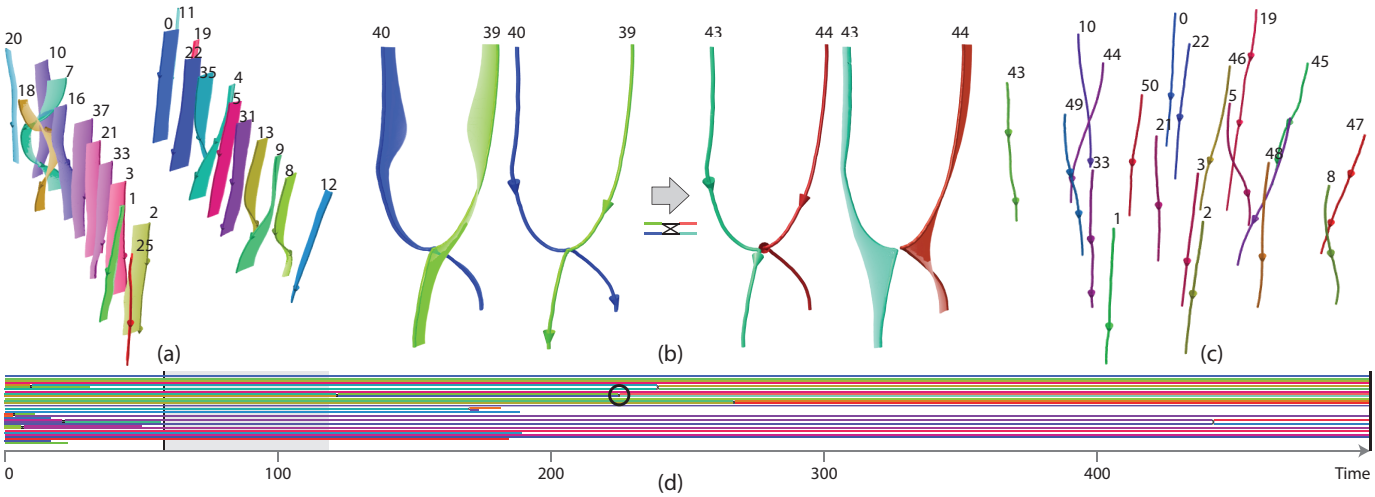

Fig. 13. Visualization results on the 3D unstructured mesh data: (a) vortex tracking results from frame 60 to 120; (b) vortex recombination event at frame 224; (c) vortex extraction results on frame 500; (d) the event diagram. The arrows on the vortex lines indicate their chiralities.

various visualization techniques, we have demonstrated how we can explore the analysis results and help scientists understand vortex dynamics and macroscale superconductor behaviors.

Future work will entail supporting in situ analysis and improving performance by exploiting the inherent parallelism in our algorithm. Because both the Condor simulation and our code use the same libMesh library, our framework can be integrated into TDGL simulations for in situ analysis. Currently, the most time-consuming part of the pipeline is detecting the punctured faces and intersected edges. However, this part of the algorithm should parallelize easily. Additionally, the vortex extraction and tracking framework may be extended to more complicated meshes, for example, adaptive mesh refinement. In event visualization, challenges remain to effectively visualize complicated events for very long time sequences, which could be addressed by new visual analysis technologies.

The framework developed here is applicable to any scientific application investigating topological defects in complex scalar fields discretized over time and space. For example, the proposed methods could be used for analyzing other complex-valued Ginzburg-Landau simulations such as superfluidity, Bose-Einstein condensation, strings in field theory, topological defects in liquid crystals, and complicated nonlinear fluid dynamics.

A Properties of Vortex Sets and Detection Lemmas

In this appendix we discuss several fundamental properties of the vortex sets of solutions ψ of Ginzburg-Landau equations (GL) on a smooth compact domain $D \subset \mathbb{R}^3$. These properties follow from the behavior of the complex parameter ψ that can be derived from the general theory of GL equations but whose proof is outside the scope of this discussion. In particular, we assume that ψ is sufficiently smooth as a simultaneous function of space and time and that for a nontrivial magnetic field at any given time ψ vanishes at most on subsets of D of topological dimension 1, roughly speaking, on curves. By definition vortices are precisely the *branch curves* for θ, where $\theta = \mathrm{Arg}(\psi)$, and the relevant techniques for reasoning about vortices are analogous to analytic continuation of functions of complex variable. In particular, the phase is single valued on any simply connected domain outside the vortex set but multi-valued on any domain that links[3] a subset of the branch points of θ. This multivaluedness is equivalent to a nontrivial increment of the phase along a closed contour γ linking the branch point set: $\Delta\theta = \int_\gamma \nabla\theta \cdot dx$. These phase integrals are topological invariants: their value does not change as long the underlying contour is deformed without changing its linking relationship with the vortex (i.e., branching) set. We summarize these properties as the following lemma:

Lemma 1. *Singularities (vortices) of the order parameter ψ are the branching curves of its phase $\theta = Arg(\psi)$. These curves either are closed or terminate on the boundary of D.*

Because vortices are extended lines that do not terminate in the interior of D, they must "puncture" any generic (i.e., up to an arbitrarily small perturbation of the mesh), smooth compact surface an even number of times: if the line "enters," it must "exit" the surface. That is, if we assume that a compact smooth surface S bounding an open set intersects the vortex line *transversely* (i.e., not tangentially) away from a branch point, we can state the following vortex conservation law, where the puncture points are *signed* by the chirality (direction) of the vortex curve:

Lemma 2. *The sum of signed puncture points on a generic S is zero.*

Because ψ is sufficiently smooth as a function of both space and time, the space-time branching curve $\mathcal{C} \subset \mathbb{R}^2 \times \mathbb{R}$ in 3D space-time is precisely the set of spatial branch points parameterized by time. That is, each time slice $\mathcal{C} \cap \mathbb{R}^2 \times \{t_0\}$ is the set of branch points of ψ at time t_0. Similarly, the space-time branch surface $\mathcal{S} \subset \mathbb{R}^3 \times \mathbb{R}$ in 4D space-time is composed exactly of the spatial branch curves at different times. It immediately follows that the values of phase integrals along contours in space-time do not depend on the contour orientation—whether the contour is confined to a fixed time-slice $t = t_0$ (horizontal) or crosses time slices (has a vertical component); the integral value depends only on the contour's linking relationship with the branch set. Therefore, we have the following lemmas.

Lemma 3. *The line integral over a space-time face is $2\pi n$ with $n \neq 0$, iff there is a vortex line intersecting the corresponding spatial edge in an intermediate time $t_0 \leq t \leq t_1$.*

Lemma 4. *The phase integral over any contour inside a space-time face over a spatial edge vanishes iff at no intermediate time does a puncture point cross the edge.*

An illustration of how a moving vortex leads to Lemmas 3 and 4 as well as a longer discussion of the behavior of phase integrals and branch sets is provided in the supplemental material.

[3] Set A links set B if closed contours are lying in A that cannot be deformed to a point without cutting B.

Acknowledgments

We thank Chunhui Liu for useful discussions. This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357. This work is also supported by the U.S. Department of Energy, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program.

REFERENCES

[1] J. Ahrens, B. Geveci, and C. Law. Paraview: An end-user tool for large data visualization. In C. D. Hansen and C. R. Johnson, editors, *The Visualization Handbook*. Elsevier, 2005.

[2] D. C. Banks and B. A. Singer. Vortex tubes in turbulent flows: Identification, representation, reconstruction. In *Proc. IEEE Visualization '94*, pages 132–139, 1994.

[3] D. C. Banks and B. A. Singer. A predictor-corrector technique for visualizing unsteady flow. *IEEE Trans. Vis. Comput. Graph.*, 1(2):151–163, 1995.

[4] D. Bauer and R. Peikert. Vortex tracking in scale-space. In *VisSym'02: Proc. Symp. Data Visualization*, pages 233–240, 2002.

[5] M. Bou-Diab, M. J. W. Dodgson, and G. Blatter. Vortex collisions: Crossing or recombination? *Phys. Rev. Lett.*, 86(22):5132–5135, 2001.

[6] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. In *Proc. SIGGRAPH '93*, pages 263–270, 1993.

[7] X. H. Chao, B. Y. Zhu, A. V. Silhanek, and V. V. Moshchalkov. Current-induced giant vortex and asymmetric vortex confinement in microstructured superconductors. *Physics Review B*, 80(054506):1–6, 2009.

[8] H. R. Childs, E. Brugger, K. S. Bonnell, J. S. Meredith, M. Miller, B. Whitlock, and N. Max. A contract based system for large data visualization. In *Proc. IEEE Visualization '05*, pages 191–198, 2005.

[9] Q. Du. Numerical approximations of the Ginzburg-Landau models for superconductivity. *Journal of Mathematical Physics*, 46(095109):1–22, 2005.

[10] R. Dndliker, I. Mrki, M. Salt, and A. Nesci. Measuring optical phase singularities at subwavelength resolution. *Journal of Optics A: Pure and Applied Optics*, 6(5):S189–S196, 2004.

[11] E. R. Gansner and S. C. North. An open graph visualization system and its applications to software engineering. *Software: Practice and Experience*, 30(11):1203–1233, 2000.

[12] A. Glatz, H. L. L. Roberts, I. S. Aranson, and K. Levin. Nucleation of spontaneous vortices in trapped Fermi gases undergoing a BCS-BEC crossover. *Physics Review B*, 180501(84):1–4, 2011.

[13] H.-C. Hege, M. Koppitz, F. Marquardt, C. McDonald, and C. Mielack. Visual analysis of quantum physics data. In A. D. Bandrauk and M. Ivanov, editors, *Quantum Dynamic Imaging*, pages 71–87. Springer, 2011.

[14] J. Jeong and F. Hussain. On the identification of a vortex. *Journal of Fluid Mechanics*, 285:69–94, 1995.

[15] M. Jiang, R. Machiraju, and D. Thompson. A novel approach to vortex core region detection. In *VisSym'02: Proc. Symp. Data Visualization*, pages 217–225, 2002.

[16] M. Jiang, R. Machiraju, and D. Thompson. Detection and visualization of vortices. In C. D. Hansen and C. R. Johnson, editors, *The Visualization Handbook*. Elsevier, 2005.

[17] J. Kasten, J. Reininghaus, I. Hotz, and H.-C. Hege. Two-dimensional time-dependent vortex regions based on the acceleration magnitude. *IEEE Trans. Vis. Comput. Graph.*, 17(12):2080–2087, 2011.

[18] S. Kim, C.-R. Hu, and M. J. Andrews. Steady-state and equilibrium vortex configurations, transitions, and evolution in a mesoscopic superconducting cylinder. *Physics Review B*, 69(094521):1–18, 2004.

[19] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey. `libMesh`: A C++ library for parallel adaptive mesh refinement/coarsening simulations. *Engineering with Computers*, 22(3–4):237–254, 2006.

[20] H. Obermaier, J. Mohring, E. Deines, M. Hering-Bertram, and H. Hagen. On mesh-free valley surface extraction with application to low frequency sound simulation. *IEEE Trans. Vis. Comput. Graph.*, 18(2):270–282, 2012.

[21] K. O'Holleran, F. Flossmann, M. R. Dennis, and M. J. Padgett. Methodology for imaging the 3D structure of singularities in scalar and vector optical fields. *Journal of Optics A: Pure and Applied Optics*, 11(094020):1–7, 2009.

[22] M. Otto and H. Theisel. Vortex analysis in uncertain vector fields. *Comput. Graph. Forum*, 31(3):1035–1044, 2012.

[23] J. Pauschenwein and B. Thaller. Visualizing quantum-mechanical wave functions in three dimensions with AVS. *Computers in Physics*, 10(6):558–566, 1996.

[24] R. Peikert and M. Roth. The "parallel vectors" operator - a vector field visualization primitive. In *Proc. of IEEE Visualization '99*, pages 263–270, 1999.

[25] C. L. Phillips, T. Peterka, D. Karpeyev, and A. Glatz. Detecting vortices in superconductors: Extracting one-dimensional topological singularities from a discretized complex scalar field. *Physics Review E*, 91(023311):1–12, 2015.

[26] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The state of the art in flow visualisation: Feature extraction and tracking. *Comput. Graph. Forum*, 22(4):1–17, 2003.

[27] M. Roth and R. Peikert. Flow visualization for turbomachinery design. In *Proc. IEEE Visualization '96*, pages 381–384, 1996.

[28] I. A. Sadovskyy, A. E. Koshelev, C. L. Phillips, D. A. Karpeev, and A. Glatz. Stable large-scale solver for Ginzburg-Landau equations for superconductors. *arXiv: 1409.8340 [cond-mat.supr-con]*, 2014.

[29] D. Silver and X. Wang. Tracking scalar features in unstructured datasets. In *Proc. IEEE Visualization '98*, pages 79–86, 1998.

[30] S. Stegmaier and T. Ertl. A graphics hardware-based vortex detection and visualization system. In *Proc. IEEE Visualization '04*, pages 195–202, 2004.

[31] D. Sujudi and R. Haimes. Identification of swirling flow in 3D vector fields. Technical Report AIAA-95-1715, American Institute of Aeronautics and Astronautics, 1995.

[32] Y. Tanahashi and K. Ma. Design considerations for optimizing storyline visualizations. *IEEE Trans. Vis. Comput. Graph.*, 18(12):2679–2688, 2012.

[33] H. Theisel, J. Sahner, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Extraction of parallel vector surfaces in 3D time-dependent fields and application to vortex core line tracking. In *Proc. IEEE Visualization '05*, page 80, 2005.

[34] H. Theisel and H.-P. Seidel. Feature flow fields. In *VisSym'03: Proc. Symp. Data Visualization*, pages 141–148, 2003.

[35] X. Tricoche, G. Scheuermann, and H. Hagen. Tensor topology tracking: A visualization method for time-dependent 2D symmetric tensor fields. *Comput. Graph. Forum*, 20(3):461–470, 2001.

[36] X. Tricoche, T. Wischgoll, G. Scheuermann, and H. Hagen. Topology tracking for the visualization of time-dependent two-dimensional flows. *Computers & Graphics*, 26(2):249–257, 2002.

[37] B. Wang, P. Rosen, P. Skraba, H. Bhatia, and V. Pascucci. Visualizing robustness of critical points for 2D time-varying vector fields. *Comput. Graph. Forum*, 32(3):221–230, 2013.

[38] T. Weinkauf, J. Sahner, H. Theisel, and H.-C. Hege. Cores of swirling particle motion in unsteady flows. *IEEE Trans. Vis. Comput. Graph.*, 13(6):1759–1766, 2007.

[39] N. J. Zabusky, O. N. Boratav, R. B. Pelz, M. Gao, D. Silver, and S. P. Cooper. Emergence of coherent patterns of vortex stretching during reconnection: A scattering paradigm. *Phys. Rev. Lett.*, 67(18):2469–2472, 1991.