Morse Decomposition of 3D Piecewise Linear Vector Fields

Marzieh Berenjkoub, Guoning Chen; Univeristy of Houston; Houston, Texas

Abstract

Morse decomposition has been shown a reliable way to compute and represent vector field topology. Its computation first converts the original vector field into a directed graph representation, so that flow recurrent dynamics (i.e., Morse sets) can be identified as some strongly connected components of the graph. In this paper, we present a framework that enables the user to efficiently compute Morse decompositions of 3D piecewise linear vector fields defined on regular grids. Specifically, we extend the 2D adaptive edge sampling technique to 3D for the outer approximation computation of the image of any 3D cell for the construction of the directed graph. To achieve finer decomposition, a hierarchical refinement framework is applied to procedurally increase the integration steps and subdivide the underlying grids that contain certain Morse sets. To improve the computational performance, we implement our Morse decomposition framework using CUDA. We have applied our framework to a number of analytic and real-world 3D steady vector fields to demonstrate its utility.

1. Introduction

Vector fields are one of the omnipotent tools for the study of a wide range of continuous dynamical systems that describe the behaviors of gas and liquids under different circumstances. Applications that involve vector fields and their analysis include automobile and aircraft engineering, climate study, oceanography, combustion physics and chemistry, earthquake engineering, and medical practice, among others. An effective and compact way to represent these complicated data is to compute their topology [13, 1]. The topology of a steady vector field is defined as the flow recurrent features, including fixed points and periodic orbits, and their connectivity [7].

Conventional vector field topology, also referred to as the *differential topology*, is computed based on the characterization of streamlines– solutions to the ordinary differential equation that describes the vector field, which is numerically unstable. To address this, Chen et al. [2] introduced the Morse decomposition as a stable representation of the vector field topology. Different from the differential topology, Morse decomposition represents the flow recurrent dynamics by the individual disjoint sub-regions of the flow domain, called *Morse sets*, that enclose these flow recurrent features. This coarse representation offers an additional room for topology to tolerate certain amount of error or perturbation, resulting in a more stable representation compared to the differential topology. Since its introduction, Morse decomposition has been extended to analyze piecewise constant vector fields [3] and 3D vector fields defined on unstructured meshes [4]. However, there is still little work on the computation of Morse decomposition of vector fields defined on 3D regular grids.

In this paper, we introduce an effective framework that enables the efficient computation of Morse decompositions of 3D *piecewise linear* vector fields that are defined on regular grids.

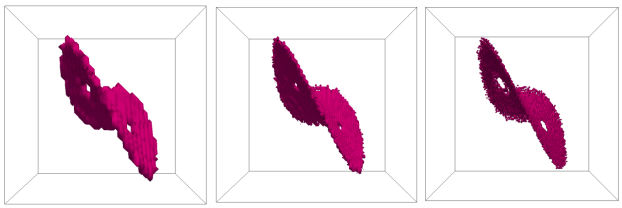

Figure 1. *Morse sets (magenta) of the Lorenz system in three different resolutions. In this case, we started with a regular* $32 \times 32 \times 32$ *cubical grid (left) on the domain* $[-30,30] \times [-30,30] \times [-10,50]$*, and gradually doubled the resolution (middle and right).* $\tau = 400$*,* 800 *and* 1600*, respectively.*

Similar to the previous approaches, the first step of our framework is to convert the original vector field into a directed graph by tracking the image of each 3D cell of the regular grid based on the vector field defined on it (i.e., flow map estimation). To accurately estimate this image, we extend the adaptive edge sampling technique introduced in [2] and make use of the configuration of the regular grid to develop a 3D adaptive face sampling strategy. This leads to an accurate construction of the directed graph. After obtaining the graph, flow recurrent dynamic features can be identified as the strongly connected components of the graph.

One challenge for Morse decomposition computation is to determine an ideal integration time for the flow map estimation, which can accurately capture the flow behavior while achieving efficient calculation. To address that, we extend the hierarchical refinement framework for 2D vector fields [20] to 3D flows, which enables the Morse decomposition to start with a smaller integration time. This integration time will be increased gradually at regions that may contain flow recurrent dynamics to refine the obtained results. In addition, our refinement framework borrows ideas from the recently introduced image-space Morse decomposition for 2D vector fields [5], and procedurally sub-divides the 3D cells to refine the boundary of the obtained Morse sets. Figure 1 provides an example of such a refinement to the Morse set detected from the Lorenz system.

Since the input vector fields are defined on regular grids, a CUDA implementation of our framework is possible, which largely improves the computation performance of our framework. This is much needed in practice when handling large scale 3D vector fields. To our best knowledge, this is the first work that addresses Morse decomposition of piecewise linear vector fields defined on 3D regular grids. It is also the first time that a CUDA implementation of Morse decomposition is presented. We have applied our framework to a number of analytic and real-world 3D data sets to demonstrate its utility.

The rest of the paper is organized as follows. Section 2 discusses the related work on vector field topology. Section 3 reviews the Morse decomposition. Section 4 provides the detail

of our 3D flow combinatorialization based on the adaptive face sampling. Section 5 describes our pipeline of hierarchical Morse decomposition. Section 6 details the implementation of our algorithm using CUDA. Experimental results for analytical and simulation data are provided in Section 7, followed by the conclusion and possible future work in Section 8.

2. Related Work

There exists a large number of algorithms and techniques for the visualization and analysis of vector-valued data. A series of survey papers [8, 9] provide a comprehensive review of this vigorous research area. In the following, we review only the most related work to the presented research.

Helman and Hesselink were the first to introduce vector field topology to the visualization community [13]. Specifically, they defined a topological skeleton that is comprised of first-order fixed points and their connectivity. This topological skeleton has been extended to handle the boundary features and higher-order fixed points [14], respectively. Wischgoll and Scheuermann [15] introduced a technique for periodic orbit detection from 2D steady vector fields, which is later extended to 3D [24].

Chen et al. introduced an efficient method for detecting periodic orbits from 2D/2.5D vector fields and defined a more complete vector field topology by including periodic orbits into the topology [7]. Theisel et al. proposed a saddle-saddle connector for 3D vector field topology visualization to reduce the occlusion issue[15]. Extracting higher-order of critical points has been introduced by Weinkauf et al. to assist the simplification of 3D vector field topological representation[16]. Considering the problem of instability in previous methods, Chen et al. introduced the first stable and discrete representation of vector field topology based on Morse decomposition[2]. This framework has been applied to 3D vector fields defined on unstructured grids [4] and extended to construct a hierarchical representation of the flow structure [20], respectively. Different from their work, we handle 3D vector fields defined on regular grids, which enables us to develop some efficient algorithms and implementations to compute Morse decomposition for large scale data in pratice. Recently, Szymczak and Zhang proposed a stable Morse decomposition framework for piecewise constant vector fields on surfaces[6]. Later, Szymczak et al. presented an algorithm for computing nearly recurrent components for 3D piecewise constant (PC) vector fields defined on regular grids [3]. Note that piecewise constant vector fields assume a constant vector value within each cell, which changes the nature of the original vector fields. Different from their setting, we compute Morse decomposition directly based on the original piecewise linear vector fields.

3. Background

In this section, we will briefly review some important concepts of vector fields and Morse decomposition.

Consider a 3-manifold $\mathbb{M} \subset \mathbb{R}^3$, a general vector field $\mathbf{u}$ can be expressed as an ordinary different equation $\dot{\mathbf{x}} = \mathbf{u}(\mathbf{x},t)$ or a map $\varphi : \mathbb{R} \times \mathbb{M} \to \mathbb{R}^3$, satisfying $\varphi_{t_0}^{t_0}(\mathbf{x}) = \mathbf{x_0}$ and $\varphi_{t_0}^{t+s}(\mathbf{x}) = \varphi_s^{t+s}(\varphi_{t_0}^s(\mathbf{x})) = \varphi_t^{t+s}(\varphi_{t_0}^t(\mathbf{x}))$. This flow map describes the spatial correlation of points through *trajectories* (or flow paths) starting at time t_0: $\varphi_{t_0}^t(\mathbf{x})$. In steady flows, a point $\mathbf{x}_0 \in \mathbb{M}$ is a *fixed point* if $\varphi(t,\mathbf{x}_0) = \mathbf{x}_0$ for all $t \in \mathbb{R}$. That is, $\mathbf{u}(\mathbf{x}_0,t) = 0$. $\mathbf{x}$ is a *periodic point* if there exists $T > 0$ such that $\varphi(T,\mathbf{x}) = \mathbf{x}$. The trajectory of

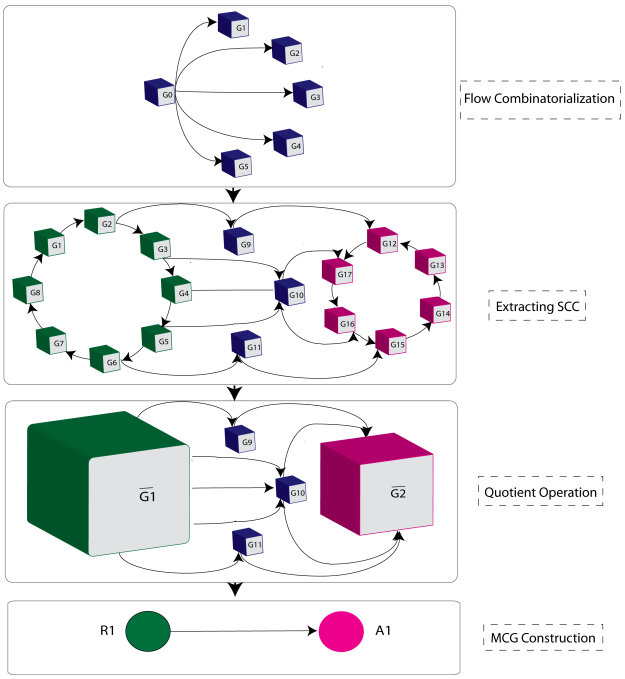

Figure 2. *The Morse decomposition computation pipeline.*

a periodic point is called a *periodic orbit*. Both fixed points and periodic orbits are examples of *flow recurrent features*. The connectivity of these flow recurrent features represents the qualitative (or structural) information of the vector fields, which we refer to as *vector field topology*.

Morse Decomposition A Morse decomposition is a collection of disjoint closed sets, called Morse sets. Together, they contain all the recurrent dynamics of the flow induced by the vector field. More precisely, sets M_i, $i \in \{1,2,\ldots,N\}$ form a Morse decomposition if and only if the trajectory of any point is either (i) entirely contained in one of the Morse sets or (ii) is contained in some Morse set M_i for large enough negative times and in some other Morse set M_j, with $j > i$, for large enough positive times. Intuitively, (ii) means that the trajectory of any point outside the Morse sets can only move from a set with lower subscript to a set with a higher subscript. (ii) excludes any recurrent dynamics outside the Morse sets, making it gradient-like. In practice, the partial order between Morse sets can be represented as an acyclic directed graph called *Morse connection graph*, or MCG.

In [2], Chen et al. described a computation pipeline for Morse decompositions of the given 2D vector fields. In this pipeline, the input vector field is firstly converted into a directed graph, denoted by $\mathscr{F}$, through a numerical computation, called flow combinatorialization. The nodes of $\mathscr{F}$ are the individual triangles of the mesh where the vector field is defined. The directed edges indicate the flow mapping relations between triangles. For instance, if there is a directed edge $T_1 \to T_2$, the particles released at triangle T_1 can be advected by the flow and reaches T_2 over certain time τ. In other words, $\mathscr{F}$ encodes the dynamics of the flow at a combinatorial level. From $\mathscr{F}$, Morse sets can be identified as the strongly connected components of $\mathscr{F}$ with non-trivial Con-

ley index. The connectivity between the identified Morse sets can then be extracted by path searching between their corresponding strongly connected components in $\mathscr{F}$. It has been shown that this pipeline can be extended to 3D as illustrated in Figure 2.

Algorithm 1 Adaptive sampling on a face algorithm

procedure ADAPTIVE_FACE_SAMPLING(V,f, τ,k)
Input: V:vector field; f:current face; τ:user specified integral time; k: the level of adaptive face sampling.
Output: V_o:the index of edges that will be added to the graph $\mathscr{F}_\tau$ in Algorithm2(After all GPU computation done).
Local variables:
f_1', f_2', f_3', f_4':the four sub-faces of f;
v_1, v_2, v_3, v_4:the four vertices of f;
e_1, e_2, e_3, e_4:the four end points of f;
L: the current level of adaptive face sampling.
Begin
$L = L + 1$
if (L > k ||
check_neighborhood_condition(e_1, e_2, e_3, e_4))
return;
else
$v_{c1} = add\_particle(v_1, v_2)$;
trace(v_{m1}, τ);
$v_{c2} = add\_particle(v_2, v_3)$;
trace(v_{m2}, τ);
$v_{c3} = add\_particle(v_3, v_4)$;
trace(v_{m3}, τ);
$v_{c4} = add\_particle(v_4, v_1)$;
trace(v_{m4}, τ);
$v_{cc} = add\_particle\text{–}center(f)$;
trace(v_{cc}, τ);
check_edge_end_points();
f_1' = build_new_face($v_1, v_{c1}, v_{cc}, v_{c4}$);
call Adaptive_face_sampling(V,f_1',τ,L);
f_2' = build_new_face($v_{c1}, v_2, v_{c2}, v_{cc}$);
call Adaptive_face_sampling(V,f_2',τ,L);
f_3' = build_new_face($v_{cc}, v_{c2}, v_3, v_{c3}$);
call Adaptive_face_sampling(V,f_3',τ,L);
f_4' = build_new_face($v_{c4}, v_{cc}, v_{c3}, v_4$);
call Adaptive_face_sampling(V,f_4',τ,L);
end procedure

4. Flow Combinatorialization

As described earlier, the most important step in Morse decomposition computation is the flow combinatorialization. In this work, we employ the idea of τ-maps for the computation of flow combinatorialization. In the rest of our discussion, we assume that the underlying computation domain is represented by a regular grid. Vector values are defined at the vertices and tri-linear interpolation is used to obtain values on the edges and inside the grid.

The purpose of flow combinatorilization is to construct the directed edges $e_k = (V_i \to V_j)$ (V_i and V_j are two voxels) in $\mathscr{F}_\tau$, which accurately encodes the flow map induced by the vector field. As shown in the previous work [2], the directed edges starting from each node in $\mathscr{F}_\tau$, i.e., corresponding to a 3D cell (or a voxel) V, can be obtained by locating all the 3D cells, $\{V_j\}$, that intersect with the image of V, denoted by $\varphi_{t_0}^{t_0+\tau}(V)$ or simply $\varphi_\tau(V)$. That is, $e_k = (V \to V_j) \in \mathscr{F}_\tau \iff V_j \cap \varphi_\tau(V) \neq \emptyset$.This set of 3D cells constitutes the *outer approximation* of $\varphi_\tau(V)$. Hence, to obtain an accurate directed graph,$\mathscr{F}_\tau$, it is essential to compute an accurate outer approximation of the image of each 3D cell of the given grid. A naive approach for the outer approximation estimation is to advect the densely placed particles within each 3D cell over time τ and locate the set of cells that enclose the end positions of these particles. However, such a naive approach will usually result in disconnected Morse sets due to inaccurate estimation of the flow map (see an example shown in Figure 3 top). To address that, we extend the 2D adaptive edge sampling strategy [2] and propose an adaptive face sampling process described as follows.

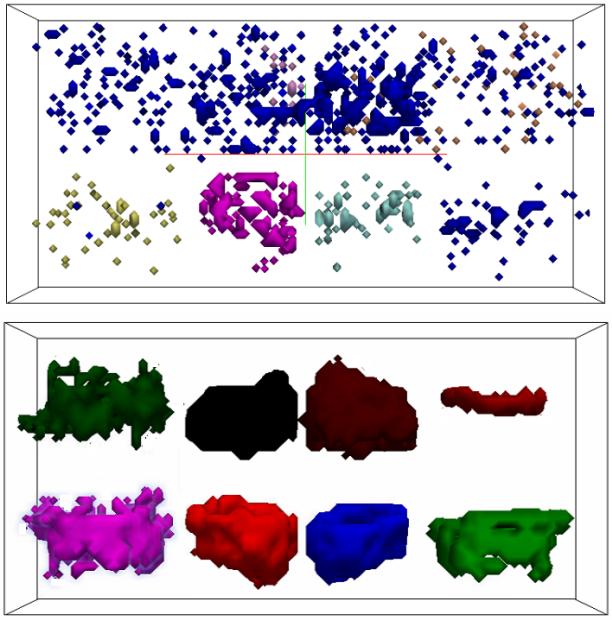

Figure 3. *Morse decomposition results of the Benard data with $64 \times 16 \times 32$ cubic cells, using $\tau = 1200$. Different colors indicate different Morse sets. The Morse sets computed without using adaptive face sampling (top) are typically disconnected, while they are well connected with the proposed adaptive sampling strategy (bottom).*

Adaptive Sampling For Faces

Consider a voxel V and its boundary ∂V. Let $\varphi_\tau(\partial V)$ be the image of the ∂V under the flow φ over time τ. It suffices to have an accurate outer approximation of $\varphi_\tau(\partial V)$ to obtain the outer approximation of $\varphi_\tau(V)$. The interior of $\varphi_\tau(V)$ will be identified via a backward mapping [2]. To obtain the outer approximation of $\varphi_\tau(\partial V)$, we propose an adaptive face sampling strategy for each V. This adaptive sampling is based on the observation that the image of a connected object under a continuous map remains connected. Consequently, the basic idea of our strategy is to preserve the connectivity of $\varphi_\tau(\partial V)$ during mapping.

Specifically, we are interested in finding the set of connected voxels that contains the image of each face of a voxel. To com-

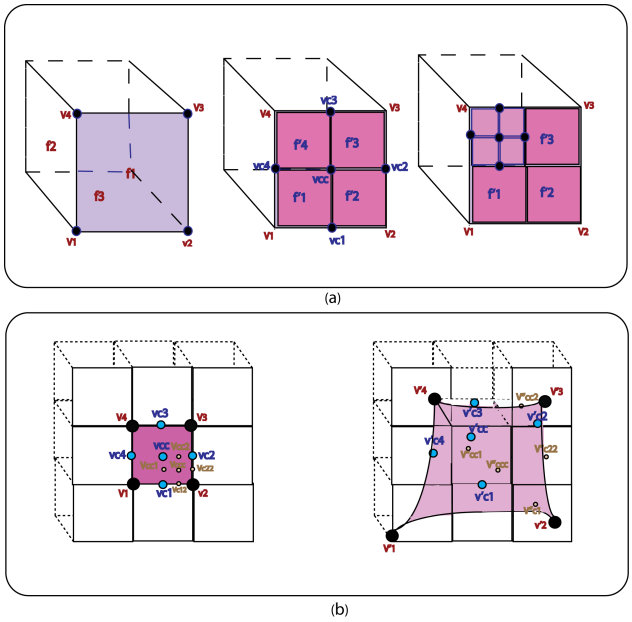

Figure 4. *Illustration of the adaptive face sampling strategy on regular grid. Top images show an example of face subdivision based on the connectivity of the image of face f_1. The bottom images illustrate an outer approximation (e.g., the shaded region) of a face (i.e., the red quad in the left).*

pute this set of connected voxels, we first release particles from the four vertices (or corners) of each face f_i. If the voxels that contain the end positions of these four particles are not identical or neighboring to each other (i.e., they are not connecting), more seeds will be inserted between the previous seeds recursively. We achieve that by subdividing f_i into four equal quads. We consider each new quad as a new face $f_{ij}^{(k)}$ and apply the same process above, i.e., placing particles at the corners of each $f_{ij}^{(k)}$ (excluding those corners that have been considered in the previous iteration) and validating the connectivity of the voxels that enclose their images. This recursive process continues until we locate a set of connected voxels that completely encloses the image of f_i. In theory, this process will converge given a finite τ and a continuous flow. In practice, we set a maximum recursive level k to prevent the over-stack exception because of the memory limitation of GPU. In all our experiments, we set k up to 4. Figure 4 provides an example to illustrate our adaptive face sampling. For a face f_1, initially four particles are placed at the four vertices v_i and advected by the flow over time τ (left image of Figure 4(a)). If the two voxels containing the images of the two vertices v_1 and v_2 are neither the same nor neighbors, we then divide f_1 to four faces in a way that v_{cc} located at the center and v_{c1} to v_{c4} are located in the middle of each edge (middle image of Figure 4(a)). We trace streamlines from the new faces and determine whether the voxels containing the images of faces f_{11} and f_{12} are connected or not. If they are not, it means that we need more samples on face (f_{11}). Therefore, we split the face segment (f_{11}) to four equal faces and recursively call the algorithm until a set of connected voxels is found. Algorithm 1 provides a pseudo-code of our adaptive face sampling. Figure 3 (bottom) shows the Morse decomposition result with the proposed adaptive face sampling. Note that in our experiments, τ is estimated by the number of integration steps.

The complete flow combinatorialization is computed in both forward and backward flow directions. Particularly, we first estimate the $\mathscr{F}_\tau$ in the forward flow direction using the above adaptive face sampling process and add the obtained directed edges accordingly. Then, we estimate $\mathscr{F}_\tau$ in the backward flow direction using the same adaptive face sampling process and add the corresponding edges by avoiding the repeated edges. In addition to the above boundary estimation, we also seed particles at the centers of the individual cells and compute their images and add the corresponding edges while avoiding redundancy [2].

5. Hierarchical Refinement

Even with the above flow combinatorialization computation, it still remains a challenge to determine an ideal τ to compute the Morse decompositions with sufficiently fine Morse sets and fast computation. Chen et al. [2] have shown that in general, the larger the τ the finer the decomposition result will be but with slower computation and larger error (e.g., disconnected Morse sets). In practice, users have to select different τ's in order to get the desired result. To address that, Chen et al. [20] proposed a hierarchical refinement framework which enables to compute a Morse decomposition using a smaller τ at first, then gradually refines the results by using a larger τ within the Morse set regions detected in the previous computation. We adapt their framework and combine it with an additional cell subdivision process to refine the boundaries of the resulting Morse sets, whose smoothness is known constrained by the resolution of the underlying grid.

Specifically, our hierarchical refinement method consists of two stages. In the first stage, we gradually increase τ for the refinement of the detected Morse set. This is based on the work by Chen et al., which demonstrates that computing the flow combinatorialization does not require a constant τ value everywhere in the domain [20]. This refinement process enables us to semi-automatically determine an ideal τ for the extraction of finer Morse sets within a local region given its flow characteristics. In the second stage, we sub-divide the voxels that fall in the resulting Morse sets from the first stage and re-compute the local Morse decomposition within each Morse set region with a higher grid resolution. This enables us to partially improve the smoothness of the Morse set boundaries for visualization, which is in a much similar spirit of the image-space Morse decomposition framework [5]. In particular, our refinement framework can be described as follows.

1) An initial Morse decomposition is computed using a small τ. 2) For each obtained Morse set (we denote it as M_i), if its size (i.e., total number of voxels within it) is larger than a user-specified threshold, we increase τ_i as $\tau_i' = 2\tau_i$ and recompute the Morse decomposition within a sub-volume corresponding to M_i. 3) If the newly obtained Morse sets are all well-connected individually, we put them into a queue Q, and set $\tau_i = \tau_i'$, and repeat the step 2) with a Morse set from Q. Otherwise, if the extracted Morse sets become disconnected or we reached the threshold for adaptive face sampling (i.e., $k > 4$), go back to the previous level of this Morse set M_i and set $\tau_i' = (\tau_i + \tau_i')/2$, then go to step 2). 4) After all Morse sets converge using steps 2) and 3), we refine the grid within the obtained Morse sets by doubling their resolution. For each Morse set M_i and its corresponding τ_i, we use $2\tau_i$ to compute a new Morse decomposition within it. If the newly

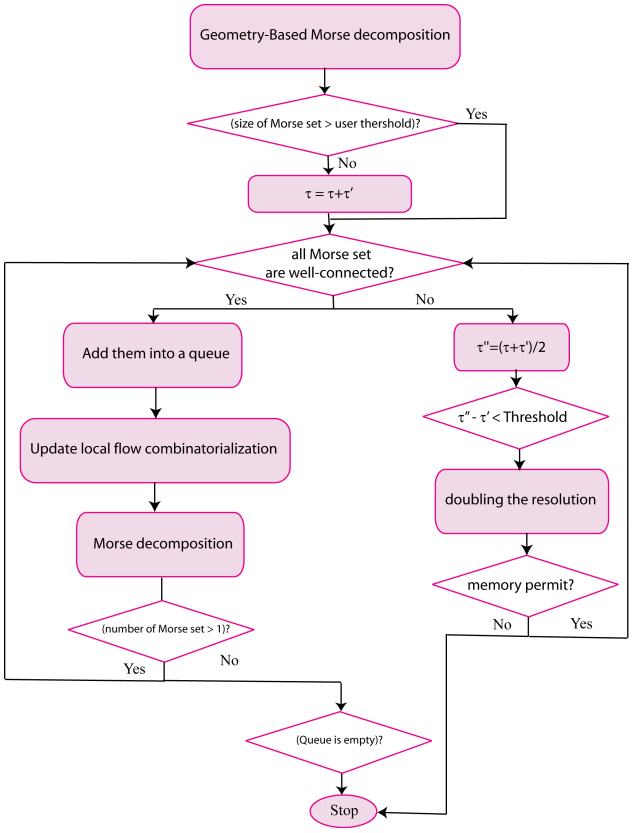

Figure 5. *The pipeline of the proposed hierarchical refinement framework of Morse decompositions of vector fields.*

obtained Morse sets are all well-connected individually, we put them into Q, and repeat step 4) to increase its resolution unless the memory doesn't permit. Otherwise, we set $\tau_i' = (\tau_i + \tau_i')/2$ and recompute the Morse decomposition within M_i and go to step 4).

To explain how the hierarchical framework searches for an ideal τ, we use the Tornado data as an example. Figure 7 shows the result of Tornado with different numbers of integration steps. We start from $\tau = 150$ and increase it by doubling it using the hierarchical refinement framework. The third image in Figure 7 shows the result which is obtained by $\tau = 600$. The resulting Morse set is disconnected. We then perform a binary search for the ideal τ, as illustrated in the pipeline shown in Figure 5. The ideal τ for this data is 450. We then subdivide the cells falling in this Morse set in attempt to further refine the Morse set. However, the resulting Morse set with a finer resolution does not improve much for this example.

6. CUDA Implementation

Even with the previously described hierarchical framework, the computation cost for the Morse decomposition of 3D vector fields can still be very high. To further accelerate the computation, we explore the GPU and CUDA implementation of our framework. As the estimation of the outer approximation of each 3D cell (or voxel) is independent of the others, it is natural to process them in parallel. In the higher-level, one can assign a thread for the outer approximation computation for each cell. In the lower-level, the outer approximation is extracted via the tracing of particles seeded at each cell. Ideally, if the advection of each particles is independent of the others, a thread can be assigned to it to achieve the maximum parallelization. However, considering the proposed adaptive face sampling strategy, not all the required particles and their initial positions are known at the beginning. This makes their parallel computation challenging. To address that, we proceed as follows (see Algorithm 2).

Algorithm 2 An efficient GPU outer approximation computation

procedure CONSTRUCT_EDGE_MAP(V,B τ,k)
Input: V:vector field; B:boundary coordinates;
Output: f_τ:completed graph;
Local variables: h:array of vertex coordinates; e: array of end points coordinate; h_p:array of seed point coordinates; e_p: array of coordinates of ending positions of seed points; Q is a queue for saving Morse sets;
Begin
  Generate_Vertex_Coordinates(h);
  for all h $\in B$
    do in parallel
      $e \leftarrow Trace\_Streamlines\_GPU(h, \tau, v)$;
    end in parallel
  end for
  Add_Edges(h,e,f_τ);
  Call Morse_Decomposition(f_τ);
  Current_Morse_Sets = Save_Morse_Set();
  do
   for all h $\in$ *Current_Morse_Sets*
    do in parallel
      $(h_p, e_p) \leftarrow Adaptive\_Face\_Sampling\_GPU(h, e, \tau, k)$;
    end in parallel
   end for
    Add_Edges(h_p,e_p,f_τ);
    Call Morse_Decomposition(f_τ);
    Current_Morse_Sets = Save_Morse_Set();
    for all $x \in$ *Current_Morse_*Sets **do**
      $if(!Check\_Connectivity(x))$; Add x to Q;
    end for
  while (!isEmpty(Q))
End
end procedure

At first, streamlines are computed concurrently by executing a CUDA thread per vertex of the grid for the initialization of flow combinatorialization. These streamlines and their end positions will be accessed multiple times in the subsequent face adaptive sampling and connectivity determination. Function *trace_Streamlines_GPU* in Algorithm 2 accomplishes this initialization. An important array for this initialization is h that contains the coordinates of the vertices. It is passed to the GPU memory.

After computing the streamlines starting from vertices, the ending positions of the streamlines will be saved at e. Based on these end positions, the corresponding directed edges are added to the graph. Specifically, since one vertex is shared by six voxels, six directed edges starting from these six voxels will be added for each streamline. In the next step, we perform the face sampling

Table 1: Performance (in seconds) and memory usage(in MB) of the Morse decomposition of the Lorenz data with different integration steps and different resolution ($k = 2$).

Res	32	64	128
τ	Memory / Time	Memory / Time	Memory / Time
100	1,162.7 / 24.611	3,463.2 / 148.981	23,199.0 / 1199.119
200	1,163.3 / 45.47	3,420.3 / 332.292	23,200.7 / 2,572.517
400	1,164.0 / 84.866	3,430.9 / 674.945	23,201.6 / 5,722.202
800	1,164.8 / 153.059	3,440.1 / 1,236.084	23,823.3 / 10,153.4
1600	1,165.0 / 193.171	3,452.9 / 1775.597	23,204.8 / 15,092.574

(i.e., *Adaptive_Face_Sampling_GPU* function in Algorithm 2), in which we need the ending positions of the vertices of the six faces of each voxel which are computed in the last step. Again, since each vertex is shared by 6 faces, we access the ending position and starting position of this voxel using four threads. One thread will cover three faces and the other three faces will be covered by other three threads running for other vertices.

During the adaptive sampling process, the connectivity of the voxels that contain the end positions of the streamlines starting from a given sub-face can be achieved by checking the adjacency of the indices of these voxels. To reduce repetition in the computation, in the implementation of the *Adaptive_Face_Sampling_GPU* function, we only consider the three faces whose normals are in the forward flow direction, and the other three faces will be covered by other threads assigned to other adjacent voxels. Because in each level of face sampling we need to compute the previous level completely, each face must be split sequentially. The algorithm of this adaptive face sampling process has been described in Figure 4 and Algorithm 1.

Since it is impossible to run a particle per thread because the number of required levels of adaptive sampling as well as the required particles are unknown, we allocate the GPU memory to the size of maximum number of particles in all levels that the algorithm may need. As we add one particle to the middle point of each edge of every (sub-)face and one to its center, each (sub-)face will place totally 5 particle. If we set $k = 4$, the total number of all the (sub-)faces involved in this adaptive sampling process will be $341 = \sum_{k=0}^{4} 4^k$. Thus, the maximum number of particles needed for processing a face will be $5 \times 341 = 1705$.

7. Results

We have applied our 3D Morse decomposition framework to a number of analytic and real-world flow data. Figures 1 and 6 show the obtained Morse sets for the *Lorenz system*. This vector field is defined on a $32 \times 32 \times 32$ grid with $\sigma = 10$, $b = 8/3$ and $\rho = 28$ in the domain $[-30;30] \times [-30;30] \times [-10;50]$, which is originally presented in [21]. From the result, we see our method successfully locates the Morse set that encloses a saddle-like periodic orbit. Figures 3b and 8 show the result for the *Benard-Rayleigh Convection* data [22] using one-level cell subdivision. We have subsampled the data set to $64 \times 16 \times 32$ within the domain $[-16;16] \times [-4;4] \times [-8;8]$. Our result clearly shows the eight Morse sets that match the configuration of the Benard flow. Figure 9 shows the flow behind a *Square-Cylinder* Data [23].The dimension of this data is $96 \times 32 \times 24$ in the domain $[-20;12] \times [-4;4] \times [0;6]$. By subtracting the average (or mean) velocity from the flow, we can observe some interesting swirling structure, which is highlighted by the identified Morse set. Figure 7 shows the flow of *Tornado* Data [25]. This vector field is defined on a $32 \times 32 \times 32$ grid and in the domain $[-30;30] \times [-30;30] \times [-10;50]$. Our result identifies one connected Morse set that corresponds to the core of the tornado.

To study the performance of the proposed Morse decomposition, we conduct a number of experiments with the above data sets. There are a number of factors that may affect the performance of our pipeline including its computational time and memory consumption. In the following experiments, we particularly concentrate on four factors: 1) the resolution of the grid, *res* 2) number of integration steps τ, 3) number of levels of adaptive face sampling k and 4) number of levels of refinement L. Tables 1 and 2 provide the performance report of our experiments, respectively. All the performance information is measured on a PC with Intel Xeron 2.6GHz 2 processor and 128GB RAM and a Quadro K4000 Graphic Card.

Different integration step τ and resolution *res* To see how different resolution affect the performance of our framework, we compute the Morse decompositions of the Lorenz system with three resolutions of 32, 64 and 128, respectively. We set the number of levels of adaptive sampling k to 2 and start the τ from 100 and increase it up to the $1,600$. Figure 6 shows the result of Lorenz with different integration steps. The computation times for $\tau = 200$, 400, 800 and 1600 are 45.47, 84.866, 153.059 and 193.171 seconds, respectively, and the memory foot-prints are at most 1,164.0MB for all of them. Figure 6 and Table 1 provide the results of Lorenz with different integration steps and resolutions.

Different numbers of levels of adaptive sampling k To see how different numbers of levels of the proposed adaptive sampling algorithm affect the performance of our framework, we compute the Morse decomposition of all four data sets with different levels of adaptive sampling. We use the sub-sampled data for our computation which is mentioned at the beginning of this section. As described earlier, due to the hardware constraint, we set the number of levels of adaptive sampling, k, up to 4. From these results, we see that the computation time is a few seconds for all data sets when $k = 0$. When k is increased, the computation times are increased accordingly. However, the increase of the computation time need not be linear with respect to k. This is in large due to the different flow configuration of different data. Among different data sets, the computation time is varying. This is mostly because of different resolutions of the data and different parameter settings for the computation. For example, the cylinder data has the highest resolution, therefore, its computation time is higher than the others when using a similar τ. Another example is

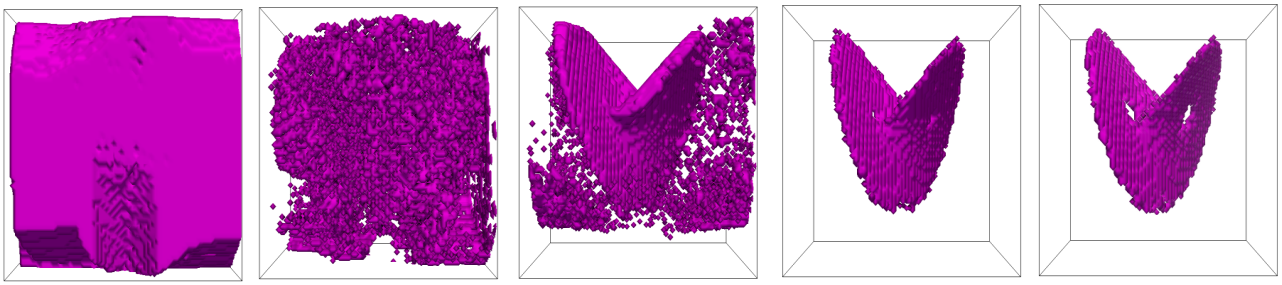

Figure 6. *The Morse set of the Lorenz system with resolution of* $32 \times 32 \times 32$*. The results are obtained using* $\tau = 100$*, 200, 400, 800 and 1600, respectively. The computation times are 24, 45, 84, 153 and 193 seconds, respectively.*

Table 2: Performance of the Morse decomposition for all four data sets with different levels of adaptive sampling. Both τ and resolution are fixed for the individual data set. Specifically, for Lorenz: τ=200 and *res* is $32 \times 32 \times 32$; for Tornado:τ=100 and *res* is $32 \times 32 \times 32$; for Benard:τ=1200 and *res* is $64 \times 16 \times 32$; for cylinder:τ=200 and *res* is $96 \times 32 \times 24$.

Dataset	k / Measurements	0	1	2	3	4
Lorenz	Time (s)	1.453	21.768	24.42	52.330	89.366
	Memory (MB)	398.0	1,156.3	1,156.2	1,159.1	1,162.5
Tornado	Time (s)	1.543	31.768	32.815	100.367	252.005
	Memory (MB)	398.6	1,154.3	1,154.4	1,155.1	1,155.6
Benard	Time (s)	2.531	241.519	242.261	976.892	2996.884
	Memory (MB)	400.1	1,157.7	1,157.6	1,163.0	1,171.5
Cylinder	Time (s)	1.859	78.209	79.522	283.086	742.849
	Memory (MB)	187.7	1,886.5	1,886.5	1,889.1	1,900.2

the Benard data which has a highly convoluted flow configuration. Therefore, a larger τ (i.e., $1,200$) is needed in order to isolate the eight vortex systems. Consequently, it has the longest computation time in general. In terms of the physical memory consumption, it is easy to understand that the larger the data set (with larger number of cells) the more memory it will consume. For instance, the cylinder data typically demands more memory space than the other data sets. In terms of the level of adaptive sampling, when $k = 0$, the memory use is relatively small, while it becomes much larger when $k > 0$ due to the enabling of the adaptive sampling. However, the memory consumption does not change much if k is increased. This can be explained by the fact that we initialize the memory for the storage of the graph by assuming an average 16 particles for each voxel in our implementation.

Another interesting observation from our experiments is that for some data set, after k is larger than certain value, the improvement of the resulting Morse decomposition (i.e., the connectivity and smoothness of the Morse sets) is not obvious. For example, for the Tornado data, we found a connected Morse set consisting of $2,701$ voxels using $k = 1$. When we set $k = 2,3,4$, the obtained Morse sets contain $2,901$, $3,006$ and $3,091$ voxels, respectively.

Different levels of cell subdivision An important factor that affects the possible levels of cell subdivision is the GPU memory constraint. Since we have no a-priory information for the number of particles, we have to allocate the possible maximum number for the returning array of indexes. It's also dependent on the dimension of the data set, i.e. for Lorenz data in resolution of $64 \times 64 \times 64$, we can have adaptive sampling up to 3 levels, and for Benard data in resolution of $128 \times 32 \times 64$, the maximum level of refinement is 2.

Figures 1 and 9 show the examples of the effect of cell subdivision to the decomposition. Generally, the deeper the level of refinement is used, the smoother and more coherent would the results become.

CPU vs. GPU There is a substantial performance gain with the GPU version as shown by the comparison in Table 3. The GPU version is 7.30x to 12.87x faster than the single threaded CPU implementation. The reductions in the run-time are from 843.274 to 115.436 seconds for the Lorenz, 13,295.7 to 1,032.509 seconds for the Benard, 1,096.402 to 98.761 seconds for the Tornado and 881.951 to 2,193.303 seconds for the Square-Cylinder. Those correspond to 7.8, 12.8, 35.2 and 4.49 times speed-up of the GPU over the CPU, respectively. The smallest speed up is for Cylinder and the largest one is for Benard. The reason is because of the longer integration step and also the complexity of the Benard flow, which needs to go through deeper levels of adaptive sampling in order for the extracted Morse sets to be converged.

8. Conclusion

In this paper, we have presented a new framework for Morse decomposition of piecewise linear vector field defined on 3D regular grids. Most vector fields of interest in science and engineering are three-dimensional. Our approach allows one to analyze them directly, without being restricted to slices or other two-dimensional domains. Specifically, we present a hierarchical Morse decomposition framework that enables the automatic se-

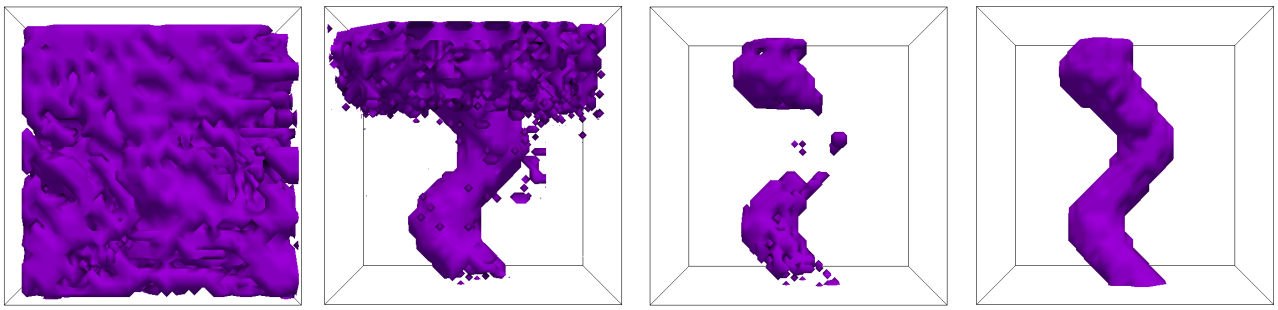

Figure 7. *Results for the Tornado data set using the proposed hierarchical framework. The results shown in the first three images are generated with* $k = 3$ *and* $\tau = 150$, 300, *and* 600, *respectively. We see the Morse set become disconnected with* $\tau = 600$. *We then apply the binary search for the ideal* τ *which turns out to be* 450 *(the right image). The computation times are 19.715, 58.419, 86.427, and 98.761 seconds, respectively.*

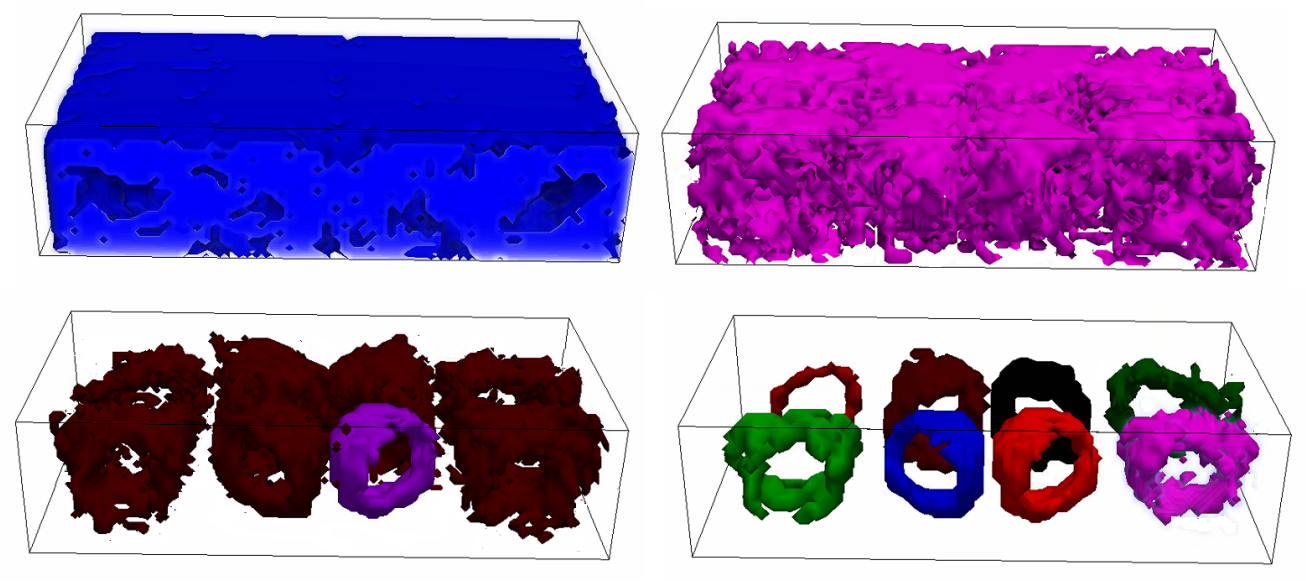

Figure 8. *Results for the Bernard-Rayleigh convection data set using the sub-sampled data set. The result on the top left is generated using* τ *= 1200 and* $k = 1$. *The top right result is obtained using* $\tau = 2400$ *and* $k = 2$. *The bottom left image is generated using* $\tau = 4800$ *and* $k = 3$, *which contains a disconnected Morse set. We then apply the binary search and determine the ideal* τ *for those local regions that may contain flow recurrent dynamics. The bottom right image shows the converged result with eight isolated Morse sets (in different colors). The* τ *used for these Morse sets range from* $2,400$ *to* $3,600$. *The computation times are 143.2, 287.34, 976.53 and 1032.51 seconds, respectively.*

Table 3: The comparison of the performance (in terms of the computation time (in seconds) and memory consumption (in MB)) of a GPU implementation versus a CPU implementation of the proposed pipeline with the Lorenz, Tornado, and Benard data sets. The last column is the speed-up gained from the GPU implementation.

	GPU	CPU	
Dataset	Memory / Time	Memory / Time	**Speed-up**
Lorenz	1,233.8 / 115.436	1,341.9 / 843.274	**7.30x**
Benard	1,367.8 / 1,032.509	1,481.8 / 13,295.7	**12.87x**
Tornado	1,328.7 / 98.761	1,419.7 / 1,096.402	**11.10x**
Cylinder	9,288.9 / 881.951	9,858.1 / 2,193.303	**4.49x**

lection of a proper τ for the flow map estimation at different flow regions in order to generate Morse decomposition with desired fineness. We also introduce an adaptive face sampling strategy for the accurate estimation of the outer approximation of each cell during the flow combinatorialization computation. Finally, we integrate the cell subdivision strategy into our computation pipeline to refine the boundaries of the obtained Morse sets. Our pipeline has been implemented using CUDA and applied to a number of 3D data sets to demonstrate its efficacy.

There are some limitations of the current framework. In particular, for the flow behind a square cylinder data set, our method cannot sufficiently refine the obtained Morse set due to the limited level of adaptive face sampling (i.e., k). In addition, the proposed CUDA implementation may not be optimal, as it does not utilize the advantage of shared memory mechanism. Furthermore, the memory allocation for particles is performed in the beginning of the algorithm and remains unchanged, which is sub-optimal. In the future, we plan to address all these issues.

Acknowledgment

We thank Tino Weinkauf and Robert Laramee for the data. This research was supported by NSF IIS-1352722.

Author Biography

Marzieh Berenjkoub is a Ph.D. student at the Department of Computer Science at the University of Houston. She earned her M.S degree in Computer Science from University of Tabriz, Iran. Her research interests include visualization, computer graphics, medical imaging, and human-computer interaction.

Guoning Chen is an Assistant Professor at the Department of Computer Science at the University of Houston. He earned his Ph.D. degree in Computer Science from Oregon State University in 2009. His research interests include scientific data analysis and visualization, geometric modeling, geometry processing, and physically-based simulation. He is a member of IEEE and ACM.

References

[1] Laramee, R.S. Laramee and H. Hauser and L. Zhao and F. H. Post. "Topology-based flow visualization, the state of the art." Topology-based methods in visualization (TopoInVis2005). Springer Berlin Heidelberg, 2007. 1-19.

[2] Chen, Guoning, Konstantin Mischaikow, Robert S. Laramee, and Eugene Zhang. "Efficient Morse decompositions of vector fields." IEEE Transactions on Visualization and Computer Graphics, 14(4): 848-862, 2008.

[3] Szymczak, Andrzej, and Nicholas BrunhartLupo. "Nearly recurrent components in 3D piecewise constant vector fields." Computer Graphics Forum. 31(3): 1115-1124. 2012.

[4] Reich, Wieland, Dominic Schneider, Christian Heine, Alexander Wiebel, Guoning Chen, and Gerik Scheuermann. "Combinatorial vector field topology in 3 dimensions." 4th Workshop on Topology-Based Methods in Data Analysis and Visualization (TopoInVis2011). 2011.

[5] Chen, Guoning, and Shuyu Xu. "An image-space Morse decomposition for 2D vector fields." IS&T/SPIE Visualization and Data Analysis (VDA) conference, February, 2015.

[6] Szymczak, Andrzej, and Eugene Zhang. "Robust Morse decompositions of piecewise constant vector fields." IEEE Transactions on Visualization and Computer Graphics 18(6): 938-951, 2012.

[7] Chen, Guoning, Konstantin Mischaikow, Robert S. Laramee, Pawel Pilarczyk, and Eugene Zhang. "Vector field editing and periodic orbit extraction using Morse decomposition." IEEE Transactions on Visualization and Computer Graphics, 13(4): 769-785, 2007.

[8] Post, Frits H., Benjamin Vrolijk, Helwig Hauser, Robert S. Laramee, and Helmut Doleisch. "The state of the art in flow visualization: Feature extraction and tracking." Computer Graphics Forum. 22(4): 775-792, Blackwell Publishing, Inc, 2003.

[9] Laramee, Robert S., Helwig Hauser, Helmut Doleisch, Benjamin Vrolijk, Frits H. Post, and Daniel Weiskopf. "The State of the Art in Flow Visualization: Dense and TextureBased Techniques." In Computer Graphics Forum, 23(2): 203-221. Blackwell Publishing Ltd., 2004.

[10] McLoughlin, Tony, Robert S. Laramee, Ronald Peikert, Frits H. Post, and Min Chen. "Over Two Decades of IntegrationBased, Geometric Flow Visualization." Computer Graphics Forum. 29(6): 1807-1829, Blackwell Publishing Ltd, 2010.

[11] Pobitzer, Armin, Ronald Peikert, Raphael Fuchs, Benjamin Schindler, Alexander Kuhn, Holger Theisel, Kreimir Matkovi, and Helwig Hauser. "The State of the Art in TopologyBased Visualization of Unsteady Flow." Computer Graphics Forum. 30(6): 1789-1811, Blackwell Publishing Ltd, 2011.

[12] Edmunds, Matt, Robert S. Laramee, Guoning Chen, Nelson Max, Eugene Zhang, and Colin Ware. "Surface-based flow visualization." Computers & Graphics 36(8): 974-990, 2012.

[13] Helman, James, and Lanbertus Hesselink. "Representation and display of vector field topology in fluid flow data sets." Computer 8: 27-36, 1989.

[14] Scheuermann, Gerik, Bernd Hamann, Kenneth I. Joy, and Wolfgang Kollmann. "Visualizing local vector field topology." Journal of Electronic Imaging 9(4): 356-367, 2000.

[15] Wischgoll, Thomas, and Gerik Scheuermann. "Detection and visualization of closed streamlines in planar flows."IEEE Transactions on Visualization and Computer Graphics, 7(2): 165-172, 2001.

[16] Ebert, D., P. Brunet, and I. Navazo. "Locating closed streamlines in 3D vector fields." methods 16: 19, 2002.

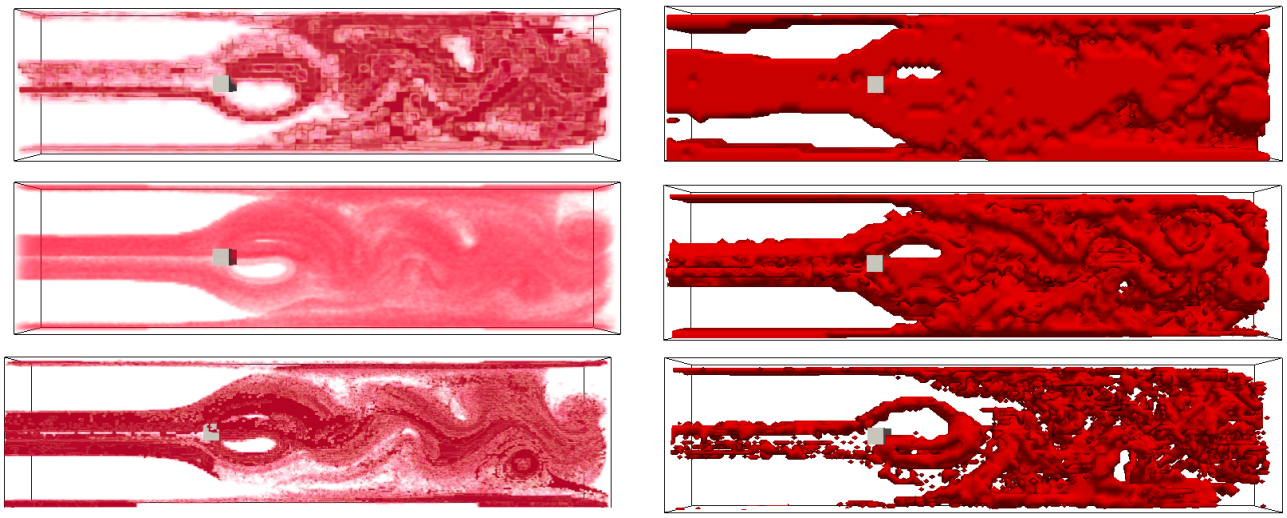

Figure 9. *Results of the flow behind a square cylinder data set. The images in the left column show the Morse decompositions of the flow with increasing resolutions (from top to bottom). Specifically, the upper-left image visualizes the Morse set (in red) obtained with resolution* $96 \times 32 \times 24$ *,* $\tau = 900$ *and* $k = 4$*; the middle-left image shows the Morse set with resolution* $192 \times 64 \times 48$*,* $\tau = 1800$ *and* $k = 2$*; the bottom-left image is the Morse set with resolution* $384 \times 128 \times 96$ *,* $\tau = 3600$ *and* $k = 1$*. The computation times for these three results are* 1150.98*,* 1621.2*, and* 2456.24 *seconds, respectively. The images in the right column shows the Morse decomposition results of the cylinder flow at time step* 2028 *with* $\tau = 200$*,* 400*, and* 800*, respectively. The resolution of the data for these results is* $96 \times 32 \times 24$ *and* $k = 1, 2$ *and* 4*. The computation times are* 161.31*,* 326.4 *and* 519.574 *seconds, respectively. Note that for all these results, only one Morse set is identified.*

[17] Theisel, Holger, Tino Weinkauf, Hans-Christian Hege, and Hans-Peter Seidel. "Grid-independent Detection of Closed Stream Lines in 2D Vector Fields." VMV. Vol. 4. 2004.

[18] Theisel, Holger, Tino Weinkauf, Hans-Christian Hege, and Hans-Peter Seidel. "Saddle connectors-an approach to visualizing the topological skeleton of complex 3D vector fields." In proceedings of IEEE Visualization, 2003.

[19] Weinkauf, Tino, et al. "Extracting higher order critical points and topological simplification of 3D vector fields." In proceedings of IEEE Visualization, 2005.

[20] Chen, Guoning, Qingqing Deng, Andrzej Szymczak, Robert S. Laramee, and Eugene Zhang. "Morse set classification and hierarchical refinement using Conley index." IEEE Transactions on Visualization and Computer Graphics, 18(5) (2012): 767-782, 2012.

[21] Lorenz, Edward N. "Deterministic nonperiodic flow." Journal of the atmospheric sciences 20.2 (1963): 130-141.

[22] Weiskopf, Daniel, Tobias Schafhitzel, and Thomas Ertl. "Texture-based visualization of unsteady 3d flow by real-time advection and volumetric illumination." Visualization and Computer Graphics, IEEE Transactions on 13.3 (2007): 569-582.

[23] Camarri, Simone, Angelo Iollo, Marcelo Buffoni, and Maria Vittoria Salvetti. "Simulation of the three-dimensional flow around a square cylinder between parallel walls at moderate reynolds numbers." AIMETA 2005 (2006): 1000-1012.

[24] Thomas Wischgoll, Gerik Scheuermann: 3D Loop Detection and Visualization in Vector Fields, Visualization and Mathematics III, Springer-Verlag, Heidelberg, Germany, 2003, pp. 151-160

[25] Falk, Martin, and Daniel Weiskopf. "Output-sensitive 3D line integral convolution." Visualization and Computer Graphics, IEEE Transactions on 14.4 (2008): 820-834.