Vision, Modeling, and Visualization (2014)
J. Bender, A. Kuijper, T. von Landesberger, H. Theisel and P. Urban (Eds.)

Hierarchical Hashing for Pattern Search in 3D Vector Fields

Zhongjie Wang[1], Hans Peter Seidel[1], Tino Weinkauf[2]

[1]MPI Informatics, Fraunhofer IGD, Saarbrücken, Germany
[2]KTH Royal Institute of Technology, Stockholm, Sweden

Abstract

The expressiveness of many visualization methods for 3D vector fields is often limited by occlusion, i.e., interesting flow patterns hide each other or are hidden by laminar flow. Automatic detection of patterns in 3D vector fields has gained attention recently, since it allows to highlight user-defined patterns and separate the wheat from the chaff. We propose an algorithm which is able to detect 3D flow patterns of arbitrary extent in a robust manner. We encode the local flow behavior in scale space using a sequence of hierarchical base descriptors, which are pre-computed and hashed into a number of hash tables. This ensures a fast fetching of similar occurrences in the flow and requires only a constant number of table lookups. In contrast to many previous approaches, our method supports patterns of arbitrary shape and extent. We achieve this by assembling these patterns using several smaller spheres. The results are independent of translation, rotation, and scaling. Our experiments show that our approach encompasses the state of the art with respect to both the computational costs and the accuracy.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1 Introduction

The visualization and analysis of vector fields play an important role in various disciplines. In the past decade, feature-based flow analysis has achieved impressive results. Amongst these methods are a few approaches for finding patterns in a 2D or 3D flow: given a flow pattern template either from the flow itself or from other resources, users are enabled to find similar structures in the flow.

The current state-of-the-art of pattern-based flow analysis still faces some challenges. We observe two major issues:

(I) Templates can often not describe interesting flow features appropriately due to their shape, which is defined by one single geometric object such as a rectangle or a box. This is often not sufficient, since various meaningful flow behaviors have irregular extents.

(II) In order to find similar occurrences of the pattern in the flow, the existing approaches use a linear comparison, i.e., they go through all possible locations, orientations, and scale factors. In our observation, this is rather slow for 3D vector fields.

In this paper, we present a fast and accurate pattern matching approach for 3D flow fields that overcomes these two issues. First, we allow the user to define a template by arranging a number of spheres with arbitrary locations and radii. Flow features with irregular extents can now be described and searched for. In fact, our definition encompasses patterns with more than one connected component.

Second, we propose a hierarchical hashing and matching algorithm which can achieve a pattern search in 3D vector fields in a few seconds with an affordable memory cost. We achieve this using our novel hierarchical hashing approach. It is based on rotation-invariant base descriptors that describe the local flow behavior at different levels of a scale space hierarchy. We hash these descriptors in an efficient and robust manner, and store these hashes in a number of hash tables. This leads to a fast query of similar occurrences within a constant number of table lookups. The similarity of the entire pattern is then essentially computed as the weighted sum of base descriptors, which are queried in the corresponding locations and scales. Our method retrieves patterns independent of rotation, scaling and translation.

The paper is organized as follows. Section 2 reviews related work. In Section 3 we introduce the concept of a scale space of a 3D vector field and how we sample it using base descriptors. Furthermore, we explain the hashing of these descriptors. Section 4 uses this for the actual pattern search in 3D flows, where the patterns are defined as arrangements of spheres. In Section 5 we evaluate the accuracy and robustness of our method. Section 6 showcases results using different flow data sets. We conclude with a discussion in Section 7.

2 Related Work

The research about pattern-based analysis of flow fields can be categorized into spatial and line-based pattern analysis. Our method falls into the former category. We will discuss these two different approaches in the following.

2.1 Spatial Pattern Analysis

Spatial pattern analysis refers to methods that work directly on the domain of the vector field.

Critical points can be considered as the smallest patterns. Topological analysis of vector fields has a long tradition [TWHS03]. Several methods exist to track topological structures in time-dependent flows [TWSH02, GTS04, WTGP10]. Vortex structures can be seen as flow patterns as well. Their definition varies greatly from the swirling behavior of particle trajectories [WSTH07] to features in derived scalar quantities [SWH05].

Topological methods and vortex methods have in common that the developer of the method defines the feature, and not the user. This is very reasonable and suitable in many scenarios. However, giving the user the flexibility to define what constitutes a feature can lead to a more application-specific focus in feature-based data analysis. This has been already demonstrated for *local* features with the highly successful SimVis approach [DGH03]. Here, the user is able to brush local feature criteria. Patterns are not supported, i.e., the criteria do not work on a region of the data, nor can shape or orientation be constrained.

Several methods exist that support user-defined patterns. Heiberg et al. [HEWK03] define a set of axis-symmetric patterns as masks, then convolve the vector field by these masks with several discretized rotations. For each possible location, the characteristic is computed by spectral decomposition of the tensor field constructed from the convolution values. Ebling et al. [ES03, ES05, ES06] follow the same idea, but use Clifford algebra to accelerate the convolution process. The methods in this group performs rather slowly due to the convolution and the sampling of rotations.

Instead of using a set of filters to summarize the rotation space by discretization, Schlemmer et al. [SHM*07] compute moment-invariant descriptors for circular flow masks to obtain invariance against scaling, translation, and rotation. Recently, Bujack et al. [BHSE14] improved the method of [SHM*07] by normalizing the above-mentioned moments. Both methods work only for 2D vector fields. Bujack et al. [BKH*15] later extend their method into 3D vector fields.

All spatial pattern matching methods above support only patterns that are defined by a single geometric shape, either a rectangle/box or a circle. In contrast to the above methods, our approach works for 3D flows, performs fast, and allows the definition of flow features with irregular extents.

2.2 Pattern Analysis using Flow Trajectories

Instead of finding similar patterns directly from the vector field, some approaches generate a large number of flow trajectories first, and analyze them to reveal flow features.

Rössl and Theisel [RT12] perform spectral decomposition to find a low dimensional similarity between stream lines. Li et al. [LWS13] propose a spatially sensitive bag-of-features method to describe a stream line by several stream line properties. McLoughlin et al. [MJL*13] use a sequence of intrinsic properties to characterize a stream line. By changing the size of the convolution kernel, they can search for lines with different level of details. Lu et al. [LCL*13] propose a distribution-based segmentation algorithm to split long stream lines into segments, then compare the similarity of stream lines based on the correlation of segments. Wei et al. [WWYM10] achieves field line search by comparing 2D sketches with the projections of 3D field lines. Wang et al. [WESW14] propose a globally invariant segmentation algorithm. Tao et al. [TWS14] resamples streamlines based on local features, and shapes the steamline comparison problem as string comparison. [BPM*13, SGSM08, SS06] use predicates to discover flow patterns.

3 Hierarchical Description, Hashing and Indexing

In the following, we discuss a hierarchical description of the flow, its encoding using base descriptors, as well as their hashing and indexing. Loosely spoken, we transform a 3D vector field into a data structure suitable for fast querying of flow patterns.

3.1 Scale Space

We consider a steady 3D vector field $\mathbf{v}(\mathbf{x})$ over the domain $D \subseteq \mathbb{R}^3$. The scale space of $\mathbf{v}$ is an ordered set $\mathcal{V} = [\mathbf{v}_0, \ldots, \mathbf{v}_n]$ of vector fields derived from $\mathbf{v}$ with decreasing complexity and resolution. It is realized by alternating the process of Gaussian smoothing and resolution reduction as shown in Figure 1. It allows us to describe the features of the flow at different scales and is the key to making the entire algorithm scale-independent.

3.2 Base Descriptors

A base descriptor encodes the local flow behavior at a certain level of the scale space. A base descriptor $\mathcal{B}(\mathbf{p}_\ell)$ is located at the grid position $\mathbf{p}_\ell$ in level ℓ of the scale space. We equip this position with a local coordinate system. This will allow us to compare different base descriptors in a rotation-invariant manner. We choose the orthonormal Frenet-Serret frames $(\mathbf{t}(\mathbf{p}_\ell), \mathbf{n}(\mathbf{p}_\ell), \mathbf{b}(\mathbf{p}_\ell))$, where $\mathbf{t}(\mathbf{p}_\ell)$ is the tangent, $\mathbf{n}(\mathbf{p}_\ell)$ is the normal, and the binormal $\mathbf{b}(\mathbf{p}_\ell) = \mathbf{t}(\mathbf{p}_\ell) \times \mathbf{n}(\mathbf{p}_\ell)$. Hence, this describes the local linearized behavior of the tangent curve through $\mathbf{p}_\ell$. We encode the local flow behavior using normalized flow samples of $\mathbf{v}_\ell$ at the six neighbors around $\mathbf{p}_\ell$

$$(\mathbf{p}_\ell \pm u_\ell \mathbf{t}) \qquad (\mathbf{p}_\ell \pm u_\ell \mathbf{n}) \qquad (\mathbf{p}_\ell \pm u_\ell \mathbf{b}), \tag{1}$$

where u_ℓ refers to the voxel size in level ℓ. This gives us the six normalized flow samples $\tilde{\mathbf{v}}_1, \ldots, \tilde{\mathbf{v}}_6$.

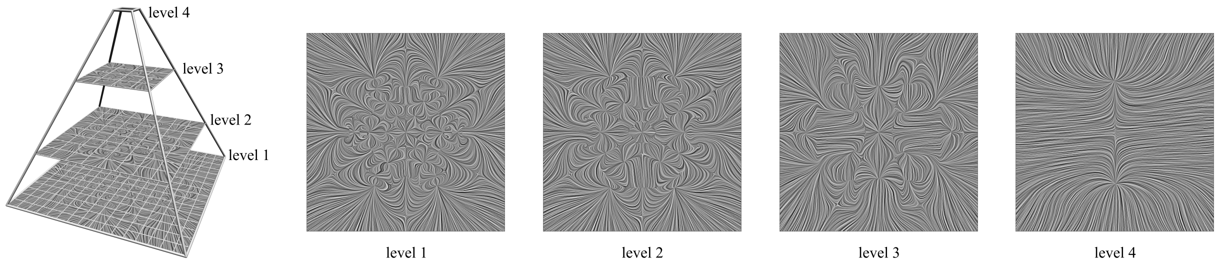


Figure 1: *2D illustration of the scale space of a vector field* **v***. The scale space consists of a number of derived vector fields, where every level is a smoothed version of the previous level at half the resolution. Our algorithm works with 3D vector fields.*

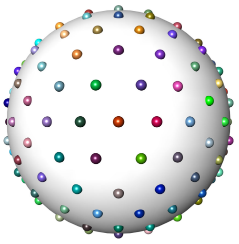


Figure 2: *Equidistant point sampling on the unit sphere. The Voronoi cells of the shown points are the bins for hashing orientations.*

Rotation invariance is achieved by relating the local coordinate system to the global one by computing a rotation matrix $\mathcal{R}(\mathbf{p}_\ell)$ (cf. Arun et al. [AHB87]):

$$(\mathbf{x},\mathbf{y},\mathbf{z}) = \mathcal{R}(\mathbf{p}_\ell)\cdot(\mathbf{t}(\mathbf{p}_\ell),\mathbf{n}(\mathbf{p}_\ell),\mathbf{b}(\mathbf{p}_\ell))\,. \tag{2}$$

After applying $\mathcal{R}(\mathbf{p}_\ell)$ to each of the six samples, we finally obtain the normalized base descriptor by concatenating them in a single vector

$$\mathcal{B}(\mathbf{p}_\ell) = (\mathcal{R}\tilde{\mathbf{v}}_1,\ldots,\mathcal{R}\tilde{\mathbf{v}}_6). \tag{3}$$

3.3 Base Descriptor Hashing

Hashing is used to speed up the search for similar descriptors. The general idea is to quantize similar descriptors into bins. A search is then only a matter of retrieving the right bin.

Base descriptors $\mathcal{B}(\mathbf{p}_\ell)$. consist of unit-length vectors, i.e., orientations. We use the unit sphere for hashing by segmenting it into a number of equally sized cells, or bins. Each orientation in the base descriptor is mapped onto the sphere, where it falls into a bin. The index of this bin is recorded. In other words, this transforms $\mathcal{B}(\mathbf{p}_\ell)$ into a vector of six indices.

The cells/bins are the Voronoi cells of an equidistant point sampling on the unit sphere. In our implementation, we obtain this by starting from an icosahedron, We subdivide it to obtain a higher resolution while still maintaining an as-equidistant-as-possible point sampling. Figure 2 shows the result that we use in our implementation. It gives rise to 162 bins, which allows us to discriminate two orientations if the angle between them is larger than 17°. Furthermore, we add an extra *null-cell* to gather disappeared orientations, e.g., due to a singular point. A *null-cell* is not a neighbor of any other bin.

Classic hashing fails to discriminate similar items at bin boundaries. We employ two strategies to deal with this: *Locality Sensitive Hashing* (LSH) [IM98] and its extension *multi-probe Locality Sensitive Hashing* [LJW*07].

LSH [IM98] mitigates the problem by employing several hashing functions. If two items are hashed into the same bin by at least one of those hashing functions, then they are considered to be similar. In our case, this means that we create several randomly rotated copies of our hashing sphere. If two orientations end up in the same Voronoi cell on at least one of those spheres, then these orientations are considered similar.

The multi-probe extension of LSH [LJW*07] affects the querying stage. When querying using a particular base descriptor $\mathcal{B}(\mathbf{p}_\ell)$, we do not only return its respective bin, but also the neighboring bins. The benefit of multi-probe LSH is that it can use fewer hashing functions and still achieve the same discrimination quality as the original LSH. This makes it faster. We refer to the literature for more details.

Let τ be the number of hashing spheres. This leads to 6τ indices for a base descriptor. These codes are pre-computed at each grid point and each level of $\mathcal{V}$. They are stored in tables where a hashing index points to a set of matching descriptors. We have 6τ tables for each hashing sphere and each of the six orientations.

3.4 Base Descriptor Querying and Comparison

Given a base descriptor $\mathcal{B}(\mathbf{p}_\ell)$, we find all similar base descriptors in the domain as follows. We map an orientation in $\mathcal{B}(\mathbf{p}_\ell)$ to the unit sphere. It falls into different bins on the different hashing spheres. A lookup in the respective tables yields τ sets of matching descriptors. The *union* of these sets holds all matching descriptors with respect to this one orientation. We perform this for all other orientations in $\mathcal{B}(\mathbf{p}_\ell)$ as well and get six sets. Their *intersection* yields the set of all base descriptors where all orientations are similar to the queried one. Two base descriptors $\mathcal{B}_i$ and $\mathcal{B}_j$ are considered to be equal if either they both can be mutually queried, or they are directly compared and considered to be similar. We define the binary cost of two base descriptors as

$$\mathcal{E}_B(\mathcal{B}_i,\mathcal{B}_j) = \begin{cases} 0 & \mathcal{B}_i = \mathcal{B}_j \\ 1 & \mathcal{B}_i \neq \mathcal{B}_j \end{cases}. \tag{4}$$

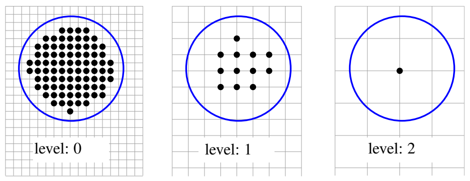


Figure 3: *2D illustration of a sphere descriptor (blue circle) consisting of a number of base descriptors (black dots) at different levels of the scale space.*

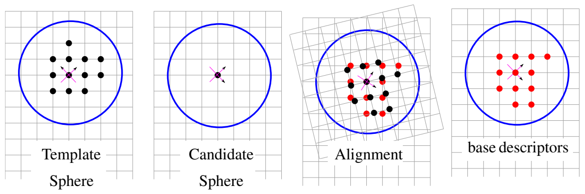


Figure 4: *2D illustration of finding a candidate sphere and its corresponding base descriptors at a specific scale level. both first base descriptors $\mathcal{B}_0$ marked with a coordinate system are aligned. The red dots represent the closest integer grid point for the rotated base descriptors of the template sphere.*

4 Pattern Definition and Search

We describe a flow pattern as a layout of spheres in the flow, i.e., the pattern is defined by selecting spherical parts of the domain. We want to find other occurrences of this pattern where the flow behavior within the spheres is similar and the spheres themselves form a similar layout. To do so, we will first discuss our definition of flow behavior within a sphere, and how to query for it in the flow. Then we show how we enforce a similar layout of spheres.

4.1 Sphere Descriptors

Consider a sphere with a certain origin $\mathbf{o}$ and radius r in the domain of the vector field $\mathbf{v}(\mathbf{x})$. It covers a set of grid points of the original vector field $\mathbf{v}(\mathbf{x})$ as well as its derived versions in the scale space $\mathcal{V}$. We sort the base descriptors located at these grid points by their levels and natural grid indices. This constitutes a sphere descriptor

$$\mathcal{D}(\mathbf{o},r) = \{\mathcal{B}(\mathbf{p}_\ell) : \ ||\mathbf{p}-\mathbf{o}|| < r, \ \mathbf{p} \in \mathcal{N}_d(\mathbf{p}_\ell)\}\,, \quad (5)$$

where $\mathcal{N}_d$ is the neighborhood along dimension d. In short, a sphere descriptor $\mathcal{D}$ consists of a sequence of base descriptors $\mathcal{B}(\mathbf{p}_\ell)$ covered by the sphere. A 2D illustration of a sphere descriptor is given in Figure 3. Note how the sphere covers less grid points in higher levels of the hierarchy, since the resolution is coarser there.

Given a sphere descriptor $\mathcal{D}$, we are interested in finding similar occurrences below a certain cost threshold. The query is processed in a coarse-to-fine framework, i.e., we start at the highest level in scale space. Let $\mathcal{B}_0$ be the first base descriptor in $\mathcal{D}$ at the coarsest level of the sphere descriptor. We find a set of possibly matching candidate spheres $\mathcal{D}'$ by querying all base descriptors similar to $\mathcal{B}_0$. Let us denote a member of this set as $\mathcal{B}'_0$.

We continue by matching neighboring base descriptors as follows: we first transform the coordinates of $\mathcal{B}_i \in \mathcal{D}$ into the local coordinate system of the sphere's first base descriptor $\mathcal{B}_0$. This is done by applying the rotation matrix $\mathcal{R}_0$ (cf. (2)). We denote the new coordinates of $\mathcal{B}_i$ as $\mathbf{p}_i$. Note that the content of the descriptors does not change. For a candidate sphere $\mathcal{D}'$, we look up the same location $\mathbf{p}_i$ in the local coordinate system of $\mathcal{B}'_0$, and get the base descriptor at the closest integer grid point as the corresponding base descriptor $\mathcal{B}'_i$. If the computed integer grid point is out of the domain, we simply mark the base descriptor as not similar. Figure 4 gives a 2D illustration of the base descriptor matching process.

When matching spheres, we give different weights w_ℓ to the base descriptors depending on their level in scale space. Note that the volume of a voxel in a level $\ell+1$ is 8 times bigger than the volume of a voxel at level ℓ. This leads to the following weights for base descriptors:

$$w(\ell+1) = 8\, w(\ell)\,. \quad (6)$$

We compute the cost between a sphere descriptor $\mathcal{D}$ and its candidate $\mathcal{D}'$ by accumulating the cost of all the base descriptors

$$\mathcal{E}_D(\mathcal{D},\, \mathcal{D}') = \sum_i w_i \mathcal{E}_B(\mathcal{B}_i,\, \mathcal{B}'_i)\,. \quad (7)$$

Since the cost $\mathcal{E}_B$ of two base descriptors is a binary value (see (4)), the largest possible sphere matching cost $\mathcal{E}_D$ is the sum of all weights of the sphere's base descriptors. The smallest matching cost is 0. This allows us to normalize the cost and define a normalized similarity measure

$$\mathcal{S}_D(\mathcal{D},\, \mathcal{D}') = \frac{\sum_i w_i - \mathcal{E}_D(\mathcal{D},\, \mathcal{D}')}{\sum_i w_i}\,, \quad (8)$$

where i is the index of the base descriptor, and $\mathcal{S}$ is within the range of $[0,\ 1]$. Especially, $\mathcal{S}=0$ if $\mathcal{D}$ and $\mathcal{D}'$ are entirely different, while $\mathcal{S}=1$ if $\mathcal{D}$ and $\mathcal{D}'$ match up completely.

4.2 Sphere Layout Filtering

Consider $\mathcal{P}$ as a template pattern defined by m sphere descriptors $\mathcal{D}^i$. This is the flow pattern that the user wants to find in a flow. Using the sphere descriptor matching from the previous section, we can find matching spheres for each individual $\mathcal{D}^i$, but this would neglect the arrangement or layout of these spheres. In this section, we introduce our approach to finding the layouts of spheres from the set of all matching spheres. A sphere in the flow matches to a sphere in the pattern, if their first base descriptors match. Such a match may have a low similarity $\mathcal{S}_D$, but we account for that when computing the similarity of the entire pattern below.

We define the local coordinate system Γ^i of each sphere descriptor $\mathcal{D}^i(\mathbf{o}^i, r^i)$ as the local coordinate system of its first base descriptor $\mathcal{B}^i_0$. It provides a stable orientation, since it is anchored at the coarsest level in scale space. We also define the level of each sphere descriptor Λ^i as the level of $\mathcal{B}^i_0$. Furthermore, we define the center of a template pattern

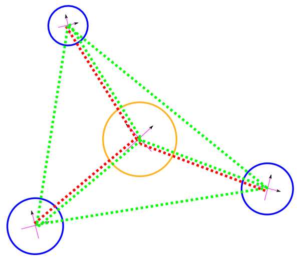


Figure 5: *Layout verification illustrated in 2D. Four circles define a template pattern in 2D. The yellow circle in the middle is the central circle as it is closest to the center of the pattern. Red dotted lines indicate the pairwise verification of scaling and rotation. The green dotted lines indicate the pairwise verification of translation.*

$\mathcal{P}$ by weighting the origins of all its spheres

$$\mathcal{O} = \frac{\sum_{i=1}^{\mathbf{m}} r^i \mathbf{o}^i}{\sum_{i=1}^{\mathbf{m}} r^i} \quad . \tag{9}$$

We designate one of the spheres in $\mathcal{P}$ as the *center sphere* $\mathcal{D}^c$, namely the one whose origin is closest to $\mathcal{O}$. Furthermore, we denote $\mathcal{P}'$ as candidate pattern, $\mathcal{D}^{i'}$ are its sphere descriptors, and $\mathcal{D}^{c'}$ is its center sphere descriptor.

A candidate pattern is produced by selecting a candidate sphere for each of the spheres in the template $\mathcal{P}$. The layout of the spheres is verified with respect to scaling, rotation, and translation by the following constraints (see also Figure 5):

- **Scaling** The level difference of $\mathcal{D}^{i'}$ and $\mathcal{D}^{c'}$ should be the same as that of $\mathcal{D}^i$ and $\mathcal{D}^c$, which can be written as
$$\Lambda^i - \Lambda^c = \Lambda^{i'} - \Lambda^{c'} \quad . \tag{10}$$
- **Rotation** The coordinate systems Γ^i and $\Gamma^{i'}$ should be similar in their own coordinate system of the center sphere, i.e.Γ^c and $\Gamma^{c'}$. The constraint is defined as
$$\measuredangle(\mathcal{R}_c\Gamma^i_j, \mathcal{R}_c{}'\Gamma^{i'}_j) < \theta, j \in \{1,2,3\} \;, \tag{11}$$
where j represents the index of three axes, i.e., tangent, normal, and binormal. $\mathcal{R}_c$ and $\mathcal{R}_c{}'$ are the rotation matrices which rotate the world coordinate system to their center sphere coordinate system. θ is the discrimination angle of the hashing sphere as discussed in Section 3.3.
- **Translation** A distance ratio threshold λ is introduced to constrain the distance deviation for any pair of spheres as
$$\forall i,j \left| \frac{\left|\left|\mathbf{o}^i - \mathbf{o}^j\right|\right| - \left|\left|\mathbf{o}^{i'} - \mathbf{o}^{j'}\right|\right|}{\left|\left|\mathbf{o}^i - \mathbf{o}^j\right|\right|} \right| < \lambda \;, \tag{12}$$
where i and j are the indices of spheres.

If a combination of spheres satisfies all three constraints, we accept this combination as a match of the template pattern, and its similarity value is computed as the average of the similarities of the single spheres

$$\mathcal{S}_P(\mathcal{P}, \mathcal{P}') = \frac{1}{m}\sum_{i=1}^{m} \mathcal{S}_D(\mathcal{D}^i, \mathcal{D}^{i'}) \quad . \tag{13}$$

5 Evaluation and Discussion

In the following, we discuss and evaluate our method. We render the detected patterns by coloring the parts of stream lines running through these areas. The color transitions from red to white to indicate the pattern similarity $\mathcal{S}_P$, i.e., high similarity is indicated in red, low similarity is shown in white. In regions where no pattern has been detected, we display fainted stream lines.

5.1 Parameter Overview

Our pattern matching approach contains two parameters that can be adjusted by the user: the distance ratio threshold λ from (12), and a threshold on the pattern similarity value $\mathcal{S}_P$ from (13). Larger values for the threshold on $\mathcal{S}_P$ reduce the number of found patterns. Smaller values of λ make the template being searched for more rigid.

Let us explain their behavior using an example from the BENZENE data set, shown in Figure 6. We select a template pattern with three spheres. The middle sphere contains laminar flow, while the other two contains a source each. We show the matching results with different parameters in the form of a grid in Figure 6. Along the horizontal axis $\mathcal{S}_P$ is increased from 0.7 to 0.9, and along the vertical axis λ spans from 0.1 to 0.2. We can see in the upper left corner that a small threshold on $\mathcal{S}_P$ and a large λ lead to a massive number of matches. To further filter the results, we can either increase the threshold on $\mathcal{S}_P$ or reduce λ. As observed, both ways achieve similar effects. For an unknown data set, we recommended to choose a small threshold on $\mathcal{S}_P$ and a large λ at the beginning. Then users see a superset and can approach a smaller set by tuning $\mathcal{S}_P$ and λ.

Other parameters are on an algorithmic level and not exposed to the user. In fact, we fixed them in our implementation as well. They include the number of Voronoi cells on the hashing sphere (Section 3.3), and the number of such (randomly rotated) hashing spheres. We fixed the former to 162, and the latter to 20.

5.2 Sensitivity to Vortex Orientation

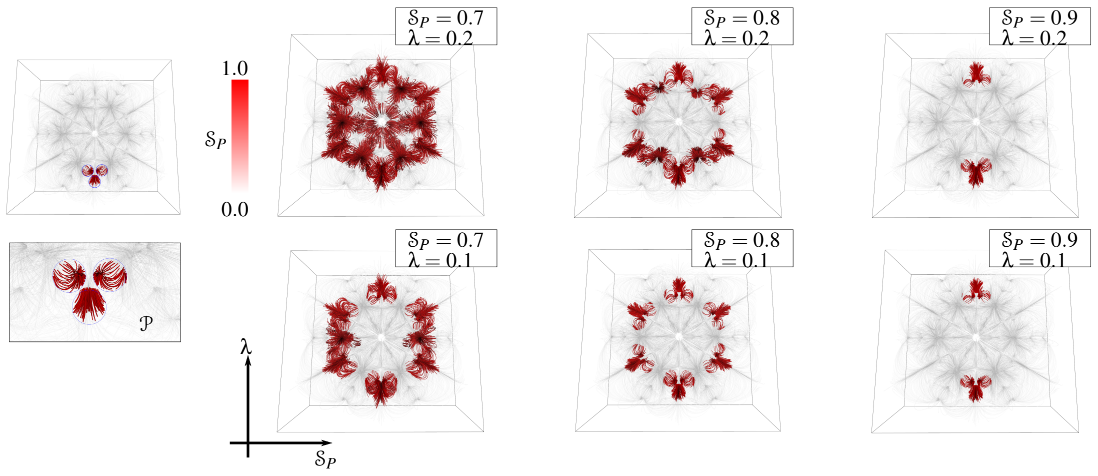
Our method is sensitive to the orientation of swirling flow, i.e., it detects whether the flow swirls in clockwise or counterclockwise orientation. Figure 11 shows this at the von Kármán vortex street in the flow behind a square cylinder.

5.3 Robustness to Transformations and Noise

We evaluate the robustness of our approach to different transformations using the analytical 3D flow

$$\mathbf{v}(x,y,z) = \begin{pmatrix} y \\ (x-0.5)(x+0.5) \\ (z-0.5)(z+0.5+2y) \end{pmatrix} , \tag{14}$$

which has four critical points at the locations $[\pm 0.5, 0, \pm 0.5]$. As the reference pattern, we select four spheres with a radius



$\mathcal{S}_P = 0.7$ $\lambda = 0.2$
$\mathcal{S}_P = 0.8$ $\lambda = 0.2$
$\mathcal{S}_P = 0.9$ $\lambda = 0.2$
1.0
$\mathcal{S}_P$
0.0
$\mathcal{S}_P = 0.7$ $\lambda = 0.1$
$\mathcal{S}_P = 0.8$ $\lambda = 0.1$
$\mathcal{S}_P = 0.9$ $\lambda = 0.1$
$\mathcal{P}$
λ
$\mathcal{S}_P$

Figure 6: *Pattern search in the* BENZENE *data set using different combination of parameters. We perform pattern search using the template pattern with three spheres on the left. The results are demonstrated in the coordinates of parameter combinations. We choose $\mathcal{S}_P \in \{0.7,\ 0.8,\ 0.9\}$ and $\lambda \in \{0.1,\ 0.2\}$.*

of 0.5 around the critical points. After the transformation, we record the similarity to the ground truth. Figure 7 shows the results.

For all experiments, the similarity increases with the volume resolution, since this allows for a better pattern alignment. For *Translation* and *Rotation*, we conducted two tests: (i) translation/rotation along a number of randomly selected axes (solid lines in the plots), (ii) translation/rotation along a coordinate axis (dashed lines). The latter indicates full similarity when the grid point arrangement is perfect. A similar observation can be made in the plot for *Scaling*. For the *Noise* validation, we pollute our pattern by adding white noise to each component of the flow. The level of noise is based on the range of component-wise magnitude inside the reference pattern. For the *Deformation* validation, we scale the pattern in one of three dimensions, and record the average similarity.

5.4 Timings

Table 1 summarizes the timings for all the experiments. All the timings are measured in single thread processing. Several factors influence the processing time. First, if a massive number of similar occurrences exist in a data set, then searching becomes slow. Second, as discussed in Section 5.1, a small threshold on the pattern similarity $\mathcal{S}_P$ as well as a big distance ratio threshold λ can also increase the processing time. The timings for hashing table generation is comparatively slow. It needs couple of minutes for preprocessing a big data set. However, it is still acceptable as this process only needs to run once for each data set.

6 Results

Figures 8 and 9 show further results from the BENZENE data set. Note how a pattern consisting of a single sphere in Figure 8 is able to capture the 6-fold rotational symmetry of this data set. The pattern in Figure 9 has a higher complexity and is irregular. It consists of four spheres. This example shows how our method provides great flexibility when defining flow patterns. In Figure 10, we perform pattern search in the RAYLEIGH-BÉNARD flow. The template pattern consists of two spheres describing a narrowing spiral in 3D. The RAYLEIGH-BÉNARD flow has eight vortices. Two of them rotate downward in clockwise manner, two others rotate downward in counterclockwise manner. The other four have the

Data set	Dimensions	Spheres	$\mathcal{S}_P$	λ	Timing (*sec.*)
BENZENE	$129 \times 129 \times 65$	1	0.73	–	3.7
BENZENE	$129 \times 129 \times 65$	3	$0.7 \sim 0.9$	0.1, 0.2	≈ 24
BENZENE	$129 \times 129 \times 65$	4	0.7	0.1	16
BÉNARD	$257 \times 65 \times 129$	2	0.8	0.1	10
CYLINDER	$257 \times 129 \times 65$	5	0.85	0.08	44
DELTAWING	$257 \times 129 \times 65$	2	0.7	0.1	40

Table 1: *Timings. For each experiment, we list the dimensions of the data set, number of spheres in the template pattern, the pattern similarity threshold $\mathcal{S}_P$, the distance ratio threshold λ, and the timing measured in single thread processing.*

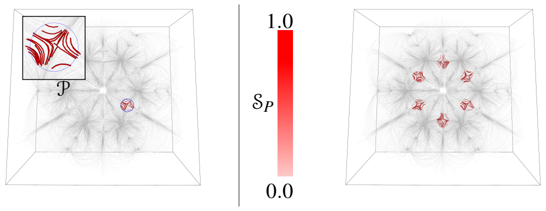


Figure 8: *Detection of the rotational symmetry in the* BENZENE *data set using a saddle-like template pattern.*

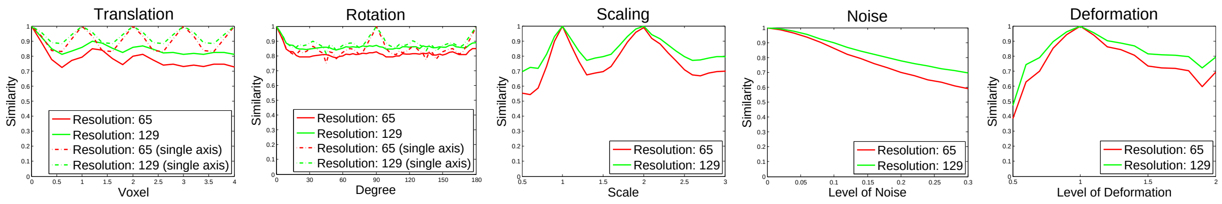


Figure 7: *Robustness validation. All validations are conducted with resolutions of* 65^3 *and* 129^3*. The dashed curves in the translation and rotation figures indicate a transformation along/around a coordinate axis, e.g., x-axis.*

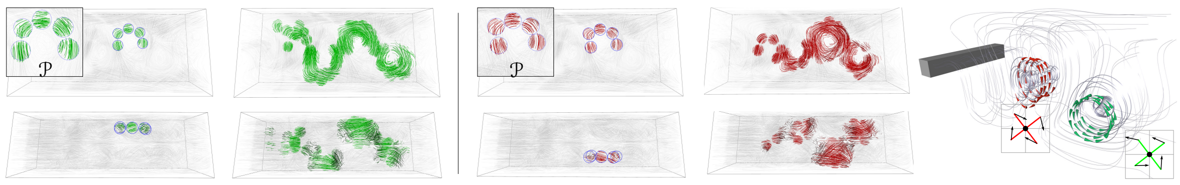


Figure 11: *Pattern search in the* CYLINDER *flow. The template pattern* $\mathcal{P}$ *consists of five spheres describing an extended arc. Since base descriptors are sensitive to the orientation of swirling flow. This enables us to distinguish between clockwise and counterclockwise rotating vortices.*

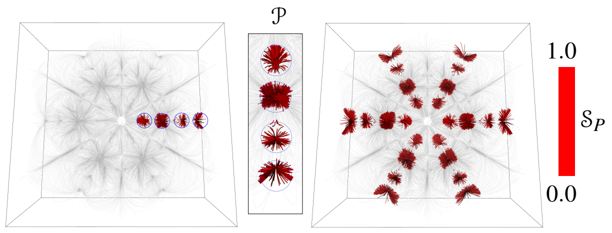


Figure 9: *Pattern search in the* BENZENE *data set. The template pattern* $\mathcal{P}$ *links four spheres. Each of them consists a singularity inside.*

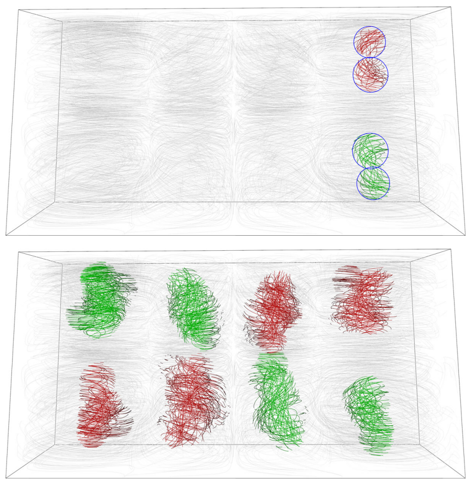


Figure 10: *Pattern search in the* RAYLEIGH-BÉNARD *flow. The template pattern* $\mathcal{P}$ *uses two spheres to describe a narrowing spiral.*

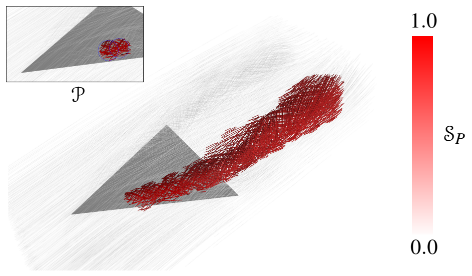


Figure 12: *Pattern search in the* DELTAWING *flow. We select two small nearby spheres to describe a small segment of a vortex core. The result shows that we detect occurrences at different scales.*

same behavior, but upwards. As the result shows, our method is able to distinguish between these differently oriented vortex structures. See also the corresponding discussion in Section 5.2. In Figure 11, we test two similar symmetric arc with different orientations. The result nicely shows the symmetric results which reflects the different orientations of swirling in the flow. Figure 12 illustrates an example in the DELTAWING flow. This is a flow around a jet. The most distinct features are two gradually expanding vortices above the wing. We can see from the figure that the detections are continuous, and their sizes are gradually increasing. This implies that our algorithm works well with continuously changing scales.

7 Conclusion, Discussion and Future Work

In this paper, we propose a hashing-based pattern search algorithm in 3D vector fields, which is invariant against translation, rotation, and scaling. The first contribution of this paper is to allow for template patterns with irregular extent. This is by arranging a number of spheres in 3D space. This way of defining templates is flexible and users are able to intuitively cover flow features with arbitrary extents. The second contribution is the hierarchical hashing strategy used to find similar patterns, which gives rise to the good performance of the algorithm. Although the proposed algorithm can obtain good results most of the time, it still has some limitations. For example, it works best on vector fields with cubic voxels. This is because of the rotational alignment when comparing sphere descriptors, as illustrated in Figure 4. However, we can virtually create such a grid over the domain if the data set has highly non-uniform voxels. We think a possible direction for further improvement is to define an interpolation method between hashing codes. With the help of code interpolation, we can query the neighboring descriptor quickly, and also get rid off the limitation mentioned above.

References

[AHB87] ARUN K. S., HUANG T. S., BLOSTEIN S. D.: Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence 9*, 5 (1987), 698–700. 3

[BHSE14] BUJACK R., HOTZ I., SCHEUERMANN G., E.HITZER: Moment invariants for 2d flow fields using normalization. In *Proc. IEEE Pacific Visualization 2014* (2014). 2

[BKH*15] BUJACK R., KASTEN J., HOTZ I., SCHEUERMANN G., HITZER E.: Moment invariants for 3d flow fields via normalization. In *IEEE Pacific Visualization Symposium, PacificVis 2015 in Hangzhou, China* (2015). 2

[BPM*13] BORN S., PFEIFLE M., MARKL M., GUTBERLET M., SCHEUERMANN G.: Visual analysis of cardiac 4d mri blood flow using line predicates. *IEEE Transactions on Visualization and Computer Graphics 19* (2013), 900–912. 2

[DGH03] DOLEISCH H., GASSER M., HAUSER H.: Interactive feature specification for focus+context visualization of complex simulation data. In *Proc. VisSym 03* (2003), pp. 239–248. 2

[ES03] EBLING J., SCHEUERMANN G.: Clifford convolution and pattern matching on vector fields. In *Proc. IEEE Visualization* (2003), pp. 193–200. 2

[ES05] EBLING J., SCHEUERMANN G.: *Clifford Convolution And Pattern Matching On Irregular Grids*. Springer, Heidelberg, 2005, pp. 231–248. 2

[ES06] EBLING J., SCHEUERMANN G.: Segmentation of flow fields using pattern matching. In *Proc. EuroVis* (2006), pp. 147–154. 2

[GTS04] GARTH C., TRICOCHE X., SCHEUERMANN G.: Tracking of vector field singularities in unstructured 3D time-dependent datasets. In *Proc. IEEE Visualization 2004* (2004), pp. 329–336. 2

[HEWK03] HEIBERG E., EBBERS T., WIGSTROM L., KARLSSON M.: Three-dimensional flow characterization using vector pattern matching. *Visualization and Computer Graphics, IEEE Transactions on 9*, 3 (2003), 313–319. 2

[IM98] INDYK P., MOTWANI R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing* (1998), ACM, pp. 604–613. 3

[LCL*13] LU K., CHAUDHURI A., LEE T., SHEN H., WONG P. C.: Exploring vector fields with distribution-based streamline analysis. In *IEEE Pacific Visualization 2013* (2013). 2

[LJW*07] LV Q., JOSEPHSON W., WANG Z., CHARIKAR M., LI K.: Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd international conference on Very large data bases* (2007), VLDB Endowment, pp. 950–961. 3

[LWS13] LI Y., WANG C., SHENE C.: Streamline similarity analysis using bag-of-features. In *Proc. SPIE* (2013), vol. 9017, pp. 90170N–90170N–12. 2

[MJL*13] MCLOUGHLIN T., JONES M. W., LARAMEE R. S., MALKI R., MASTERS I., HANSEN C. D.: Similarity measures for enhancing interactive streamline seeding. *IEEE Transactions on Visualization and Computer Graphics 19*, 8 (2013), 1342–1353. 2

[RT12] RÖSSL C., THEISEL H.: Streamline embedding for 3d vector field exploration. *IEEE Transactions on Visualization and Computer Graphics 18*, 3 (2012), 407–420. 2

[SGSM08] SALZBRUNN T., GARTH C., SCHEUERMANN G., MEYER J.: Pathline predicates and unsteady flow structures. *The Visual Computer 24*, 12 (2008), 1039–1051. 2

[SHM*07] SCHLEMMER M., HERINGER M., MORR F., HOTZ I., HERING-BERTRAM M., GARTH C., KOLLMANN W., HAMANN B., HAGEN H.: Moment invariants for the analysis of 2d flow fields. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1743–1750. 2

[SS06] SALZBRUNN T., SCHEUERMANN G.: Streamline predicates. *IEEE Transactions on Visualization and Computer Graphics 12*, 6 (2006), 1601–1612. 2

[SWH05] SAHNER J., WEINKAUF T., HEGE H.-C.: Galilean invariant extraction and iconic representation of vortex core lines. In *Proc. Eurographics / IEEE VGTC Symposium on Visualization (EuroVis '05)* (Leeds, UK, June 2005), K. Brodlie D. Duke K. J., (Ed.), pp. 151–160. 2

[TWHS03] THEISEL H., WEINKAUF T., HEGE H.-C., SEIDEL H.-P.: Saddle connectors - an approach to visualizing the topological skeleton of complex 3D vector fields. In *Proc. IEEE Visualization 2003* (2003), pp. 225–232. 2

[TWS14] TAO J., WANG C., SHENE C.: Flowstring: Partial streamline matching using shape invariant similarity measure for exploratory flow visualization. In *Proc. IEEE Pacific Visualization* (March 2014). 2

[TWSH02] TRICOCHE X., WISCHGOLL T., SCHEUERMANN G., HAGEN H.: Topology tracking for the visualization of time-dependent two-dimensional flows. *Computers & Graphics 26* (2002), 249–257. 2

[WESW14] WANG Z., ESTURO J. M., SEIDEL H.-P., WEINKAUF T.: Pattern search in flows based on similarity of stream line segments. In *Proc. Vision, Modeling and Visualization* (2014). 2

[WSTH07] WEINKAUF T., SAHNER J., THEISEL H., HEGE H.-C.: Cores of swirling particle motion in unsteady flows. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2007) 13*, 6 (11-12 2007), 1759–1766. 2

[WTGP10] WEINKAUF T., THEISEL H., GELDER A. V., PANG A.: Stable feature flow fields. *IEEE Transactions on Visualization and Computer Graphics* (2010). 2

[WWYM10] WEI J., WANG C., YU H., MA K.: A sketch-based interface for classifying and visualizing vector fields. In *Proc. IEEE Pacific Visualization* (2010), pp. 129–136. 2