Pattern Search for the Visualization of Scalar, Vector, and Line Fields

Zhongjie Wang

Dissertation
zur Erlangung des Grades
des *Doktors der Ingenieurswissenschaften (Dr. -Ing.)*
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

September 2015
Saarbrücken, Germany

Datum des Kolloquiums / Date of Defense
December 18, 2015

Dekan / Dean
Prof. Dr. Markus Bläser
Universität des Saarlandes, Saarbrücken

Vorsitzender / Chair
Prof. Dr. Philipp Slusallek

Prüfer / Examiners
Prof. Dr. Hans-Peter Seidel
Prof. Dr. Tino Weinkauf
Prof. Dr. Christoph Garth

Protokollant / Recorder
Dr. Renjie Chen

Abstract

The main topic of this thesis is pattern search in data sets for the purpose of visual data analysis. By giving a reference pattern, pattern search aims to discover similar occurrences in a data set with invariance to translation, rotation and scaling. To address this problem, we developed algorithms dealing with different types of data: scalar fields, vector fields, and line fields.

For scalar fields, we use the SIFT algorithm (Scale-Invariant Feature Transform) to find a sparse sampling of prominent features in the data with invariance to translation, rotation, and scaling. Then, the user can define a pattern as a set of SIFT features by e.g. brushing a region of interest. Finally, we locate and rank matching patterns in the entire data set. Due to the sparsity and accuracy of SIFT features, we achieve fast and memory-saving pattern query in large scale scalar fields.

For vector fields, we propose a hashing strategy in scale space to accelerate the convolution-based pattern query. 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.

For line fields, we present a stream line segmentation algorithm to split long stream lines into globally-consistent segments, which provides similar segmentations for similar flow structures. It gives the benefit of isolating a pattern from long and dense stream lines, so that our patterns can be defined sparsely and have a significant extent, i.e., they are integration-based and not local. This allows for a greater flexibility in defining features of interest. For user-defined patterns of curve segments, our algorithm finds similar ones that are invariant to similarity transformations.

Additionally, we present a method for shape recovery from multiple views. This semi-automatic method fits a template mesh to high-resolution normal data. In contrast to existing 3D reconstruction approaches, we accelerate the data acquisition time by omitting the structured light scanning step of obtaining low frequency 3D information.

Kurzfassung

Das Hauptthema dieser Arbeit ist die Mustersuche in Datensätzen zur visuellen Datenanalyse. Durch die Vorgabe eines Referenzmusters versucht die Mustersuche ähnliche Vorkommen in einem Datensatz mit Translations-, Rotations- und Skalierungsinvarianz zu entdecken. In diesem Zusammenhang haben wir Algorithmen entwickelt, die sich mit verschiedenen Arten von Daten befassen: Skalarfelder, Vektorfelder und Linienfelder.

Bei Skalarfeldern benutzen wir den SIFT-Algorithmus (Scale-Invariant Feature Transform), um ein spärliches Abtasten von markanten Merkmalen in Daten mit Translations-, Rotations- und Skalierungsinvarianz zu finden. Danach kann der Benutzer ein Muster als Menge von SIFT-Merkmalspunkten definieren, zum Beispiel durch Markieren einer interessierenden Region. Schließlich lokalisieren wir passende Muster im gesamten Datensatz und stufen sie ein. Aufgrund der spärlichen Verteilung und der Genauigkeit von SIFT-Merkmalspunkten erreichen wir eine schnelle und speichersparende Musterabfrage in großen Skalarfeldern.

Für Vektorfelder schlagen wir eine Hashing-Strategie zur Beschleunigung der faltungsbasierten Musterabfrage im Skalenraum vor. Wir kodieren das lokale Flussverhalten im Skalenraum durch eine Sequenz von hierarchischen Basisdeskriptoren, welche vorberechnet und als Zahlen in einer Hashtabelle gespeichert sind. Dies stellt eine schnelle Abfrage von ähnlichen Vorkommen im Fluss sicher und benötigt lediglich eine konstante Anzahl von Nachschlageoperationen in der Tabelle.

Für Linienfelder präsentieren wir einen Algorithmus zur Segmentierung von Stromlinien, um lange Stromlinen in global konsistente Segmente aufzuteilen. Dies erlaubt eine größere Flexibilität bei der Definition von Mustern. Für vom Benutzer definierte Muster von Kurvensegmenten findet unser Algorithmus ähnliche Kurvensegmente, die unter Ähnlichkeitstransformationen invariant sind.

Zusätzlich präsentieren wir eine Methode zur Rekonstruktion von Formen aus mehreren Ansichten. Diese halbautomatische Methode passt ein Template an hochauflösende Normalendaten an. Im Gegensatz zu existierenden 3D-Rekonstruktionsverfahren beschleunigen wir die Datenaufnahme, indem wir auf die Streifenprojektion verzichten, um niederfrequente 3D Informationen zu gewinnen.

Acknowledgement

First of all, I would like to express my sincere gratitude to my supervisor Prof. Dr. Tino Weinkauf for introducing me to the research of flow visualization. He always shows his endless patience in supporting and encouraging me to explore all my ideas, and also offers selfless guidances to help me staying in the right lane. Discussions with him are always inspirational. He uses his deep knowledge and optimistic characteristic to help me going forward. Without him, the thesis would not be possible.

I would also like to express my gratitude to Prof. Dr. Thorsten Thormählen. He introduced me to the Computer Graphics Department of Max Planck Institute for Informatics. He also uses his rich experiences to build my research skills, and always offers me detailed teachings which are very important for a freshman in academia.

I am also thankful to Prof. Dr. Hans-Peter Seidel. Thank him for arranging such a great research environment in MPII. Many high-quality internal and external talks tremendously broaden my vision in the whole discipline.

I also cannot forget to thank all the co-workers who made unconditional helps and contributions in my work. They are Dr. Janick Martinez Esturo, Himangshu Saikia, Gregorio Palmas, and Martin Grochulla. Without you, my work would be much harder and slower.

Many thanks also go to the administrative staff members, Sabine Budde and Ellen Fries from MPII, as well as Jennifer Gerling and Stephanie Jörg from IMPRS. They use their professional skills and many patience to help me arranging a lot of office work.

I would also like to express my thanks to all the researchers in MPII. They are Dr. David Günther, Dr. Alan Brunton, and Evgeny Levinkov from the visualization group in MPII, and also my officemates, Xiaokun Wu, Mohammed Shaheen, and Abhimitra Meka, and also many more who I cannot list their names all here. They are very smart, and their attitudes to work always encourage me to achieve higher goals.

Finally, I would like to thank my parents, Bing Wang and Jianfei Zhong, and my parents-in-law, Zhongchao Li and Benzhen Tan. They always stand with me and unconditionally support and encourage me throughout these study years. At last, I would like to thank my wife, Shujie. She is my source of power to overcome all the difficulties in my life.

Contents

Chapter 1

Introduction

Visualization focuses on the study of representing data using different visual elements. It plays an important role in understanding the data acquired from other disciplines, e.g., physics, chemistry, and biology. By enhancing the features or correlations in the data regarding some criteria, people can visually examine the inertial characteristics of the data. For example, the widely used visualization tools, e.g., line plot and pie graph, are very useful for examining the trend of a quantity and the composition of a subject. However, with the trend of data explosion, the quantity and the complexity of data have increased dramatically. It becomes more and more difficult to understand the data by using traditional visualization tools. Therefore, the requirement of information extraction and summarization quickly gains the focus of visualization research. A large number of feature extraction methods have been proposed over the last two decades.

However, different types of data always lead to different ways of feature extraction. In many research communities, data exist in different types. For example, language processing focuses on sequential signals, image processing focuses on images or videos, and geometry processing focuses on meshes. In this thesis, we focus on the application of pattern search in three types of data sets: scalar fields, vector fields, and line fields. These three types of data are also widely used in different research domains in visualization. Specifically, scalar fields exist in, for example, temperature or pressure analysis. Electromagnetics and flow analysis often focus on the vector fields. Line fields are widely used in streamline or trajectory analysis.

1.1 Motivation

The motivation of this thesis is the application of pattern search in flow visualization. Pattern search is a heavily discussed topic in computer vision, while in the visualization community, it is not widely used yet. The introduction of more advanced techniques of pattern search into visualization can strengthen the ability of pattern based similarity analysis. Since the flow data exists in different types, we do not directly inherit the methods from computer vision, but only follow its methodology and propose different algorithms which are suitable for different types of data that are most common with fluid simulations, i.e., scalar fields, vector fields, and line fields.

Additionally, instead of using the classic RGB images or videos, we also investigate the way of using normal maps (2D vector fields) into surface recovery. Vector fields have been widely used in optical flow analysis already. Comparing with the classic RGB images or videos (scalar fields), it gives more surface information, i.e., surface orientations (vector fields), which is more reliable than the color information. Instead of 3D reconstruction from images or scanners, in this thesis, we model the surface recovery problem as a template fitting problem, which solely relies on the normal data. Because normal maps encodes the surface information, so that the surface orientation can be iteratively optimized from multiple views.

1.2 Structure and Overview

In the following, we give an overview of the structure of the thesis. In Chapter 3, Chapter 4, and Chapter 5, we propose three algorithms for pattern search in scalar fields, vector fields, and line fields respectively. In the following, we briefly introduce these three algorithms.

In Chapter 3, we present an approach to pattern matching in 3D multi-field scalar data. Existing pattern matching algorithms work on single scalar or vector fields only, yet many numerical simulations output multi-field data where only a joint analysis of multiple fields describes the underlying phenomenon fully. Our method takes this into account by bundling information from multiple fields into the description of a pattern. First, we extract a sparse set of features for each 3D scalar field using the 3D SIFT algorithm (Scale-Invariant Feature Transform). This allows for a memory-saving description of prominent features in the data with invariance to translation, rotation, and scaling. Second, the user defines a pattern as a set of SIFT features in multiple fields by e.g. brushing a region of interest. Third, we locate and rank matching patterns in the entire data set. Experiments show that our algorithm is efficient in terms of required memory and computational efforts.

In Chapter 4, 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.

In Chapter 5, we propose a method that allows users to define flow features in form of *patterns* represented as sparse sets of stream line segments. Our approach finds “similar” occurrences in the same or other time steps. Related approaches define patterns using dense, local stencils or support only single segments. Our patterns are defined sparsely and can have a significant extent, i.e., they are integration-based and not local. This allows for a greater flexibility in defining features of interest. Similarity is measured using intrinsic curve properties only, which enables invariance to location, orientation, and scale. Our method starts with splitting stream lines using globally-consistent segmentation criteria. It strives to maintain the visually apparent features of the flow as a collection of stream line segments. Most importantly, it provides similar segmentations for similar flow structures. For user-defined patterns of curve segments, our algorithm finds similar ones that are invariant to similarity transformations. We showcase the utility of our method using different 2D and 3D flow fields.

Additionally, in Chapter 6, we discusses a computer vision application which recovers 3D surfaces from multiple normal maps (2D vector fields). We propose a semi-automatic method to fit a template mesh to high-resolution normal data, which is generated using spherical gradient illuminations in a light stage. Template fitting is an important step to build a 3D morphable face model, which can be employed for image-based facial performance capturing. In contrast to existing 3D reconstruction approaches, we omit the structured light scanning step to obtain low-frequency 3D information. This reduces the acquisition time by over 50 percent. In our experiments the proposed algorithm is successfully applied to real faces of several subjects. Experiments with synthetic data show that the fitted face template can closely resemble the ground truth geometry.

Finally, in Chapter 7, we conclude the thesis and discuss some potential future works.

1.3 Contributions

The contributions of this thesis are summarized as follows:

- In Chapter 3:
  - A novel pattern matching method is proposed for 3D multi-field data sets, which bundles the information from different fields into the description of a pattern.
  - In 3D multi-field data sets, by working with a sparse set of prominent features, the proposed method achieves response times for pattern matching of less than a second even for large data sets.
  - In contrast to the two previous versions of the 3D SIFT algorithm [17, 70], the proposed method achieves full invariance to 3D rotation by finding a robust local coordinate system. This increases the accuracy of pattern matching.
- In Chapter 4
  - For vector fields, a hierarchical hashing and matching algorithm is proposed. It can achieve a pattern search in 3D vector fields in a few seconds with an affordable memory cost.
  - The proposed algorithm allows 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 Chapter 5
  - For line fields, an example-based pattern retrieval approach is proposed. It allows users to specify interesting flow features as *patterns* that are constructed of stream line *segments*, i.e., parts of stream lines. It supports patterns represented by *multiple* line segments, which increases the flexibility and expressiveness of the specified patterns.
  - A stream line segmentation scheme is proposed. It is based only on intrinsic curve properties. It maintains the visually apparent features of the flow as a collection of stream line segments, and provides similar segmentations for similar flow structures.
- In Chapter 6
  - A novel method to semi-automatically fit a 3D face template to normal maps is proposed. This includes three main steps: feature point registration, normal registration, and shape refinement.
  - In normal registration, a novel optimization strategy to minimize a highly non-linear function is proposed. It splits the problem to several constrained optimization steps which can be linearized and solved efficiently.

1.4 List of Publications

The work presented in this thesis has been published in the following papers:

- Z. Wang, H.-P. Seidel, T. Weinkauf. Multi-field Pattern Matching based on Sparse Feature Sampling. *IEEE Transactions on Visualization and Computer Graphics*(Proc. IEEE VIS)21(12), December 2015. In Chapter 3
- Z. Wang, H.-P. Seidel, T. Weinkauf. Hierarchical Hashing for Pattern Search in 3D Vector Fields. *Proc. Vision, Modeling and Visualization*(VMV 2015). Aachen, Germany, October 7 - 10, 2015. In Chapter 4
- Z. Wang, J. Martinez Esturo, H.-P. Seidel, T. Weinkauf. Pattern Search in Flows based on Similarity of Stream Line Segments. *Proc. Vision, Modeling and Visualization*(VMV 2014). Darmstadt, Germany, October 8 - 10, 2014. In Chapter 5
- Z. Wang, M. Grochulla, T. Thormählen, H.-P. Seidel. 3D Face Template Registration Using Normal Maps. *3rd Joint 3DIM/3DPVT Conference*(3DV 2013). University of Washington, Seattle, USA, 29-30 June 2013. In Chapter 6

Chapter 2

Related Work

In the following, we discuss the related work based on two applications of this thesis, i.e., pattern search and shape recovery.

2.1 Pattern Search

Pattern search automates the process of finding similarities in a data set. First of all, we discuss the related work in this topic based on the type of reference pattern, i.e., Region-based and feature-based pattern search.

Region-based algorithms define a pattern as a compact spatial region and aim at finding similar regions within a scalar or vector field. The challenges are two-fold. First, it is necessary to detect a similar region even if it is a translated, scaled and rotated version of the pattern. Invariance to other transformations, such as small distortions or brightness changes, is desirable. Second, the computation times are often very long, which hinders interactive approaches. Region-based pattern searching for scalar fields has mainly be developed in the computer vision community and found a large number of applications. One of the most successful approaches is *SIFT*, which stands for *Scale-Invariant Feature Transform*. It was introduced by David Lowe [46] in 1999 and refined in 2004 [48] to describe local feature points in photos (2D images) in a translation-, rotation-, and scale-invariant manner. Since then, it has been applied to a number of domains including image registration [92], object recognition [47], image stitching [8], and video tracking [4]. Generalizations to higher dimensions have been proposed by Scovanner et al. [70] and Cheung et al. [17] independently. The former provided a 3D version of the SIFT algorithm, the latter a generalization to *n*-dimensional domains. For 3D domains, they both boil down to the same algorithm. These existing approaches fail to capture rotation invariance in the sense that they only determine one axis of the 3D local coordinate system. The results are sensitive to rotations around this axis. In flow visualization, some approaches are also proposed in this topic. Ebling et al. [25–27] and Heiberg et al. [35] independently introduced pattern matching for vector fields to the visualization community. Both approaches use a convolution to compute the similarity between a pattern and a location in the field. This requires to sample all possible rotations, translations and scales, and typically leads to high running times. Moment-invariant descriptors are used by Schlemmer et al. [68] and Bujack et al. [11] to achieve pattern invariance with respect to translation, rotation, and scaling. These approaches are fast, but treat 2D vector fields only. Bujack et al. [12] later extend their method into 3D vector fields.

Feature-based approaches aim at finding similar structures by comparing features with each other. Several approaches use topology to compare structures in scalar fields. Thomas et al. [76, 77] concentrate on finding all symmetric structures in a scalar field using the contour tree. Saikia et al. [63, 64] perform a similarity search for any structurally significant region as given by a subtree in the merge tree. The other methods focus on the enhancements of features. Günther" et al. [32] and Weinkauf et al. [83] propose topological denoising methods for scalar fields. Feature Flow Field proposed by Theisel et al. [75] is also widely used in extracting and tracking features. Weinkauf et al. [87] propose an automatic correction method to resolve the numeric error when tracking the features in feature flow fields. A number of methods for vector fields deal with the comparison of stream lines. Li et al. [45] uses the bag-of-features approach to describe the characteristics of stream lines. This leads to a clustering of line fields. McLoughlin et al. [54] compute signatures for stream and path lines, which accelerates the similarity computation. Lu et al. [49] propose a distribution-based segmentation algorithm to split long stream lines into segments. Wei et al. [82] achieve field line search by comparing 2D sketches with the projections of 3D field lines. Some approaches [7, 65, 66] use predicates to discover flow patterns. Several approaches [42, 62] use the spatial distance between pairwise closest points as the similarity measure for clustering stream lines. Tao et al. [73] extract shape invariant features and compare them using a string-based algorithm. Additionally, Schulze et al. [69] propose an global selection algorith for finds an optimal sets of stream surfaces to visualize the data set.

Comparing with finding similarities in a single data set, revealing the complexity of a multi-field data set is a more challenging task. Showing all fields within the same visualization leads to massive occlusions, while individual visualizations fail to communicate the commonalities. Several approaches have been proposed to deal with these issues such as multiple coordinated views [24], correlation analysis [67], or the rather recent topological approaches of Joint Contour Nets [15] and Pareto Optimality [38]. In all these approaches, finding similarities between different parts of the data is essentially a manual process.

2.2 Shape Recovery

In the past few years, many algorithms for recovering 3D facial geometry have been proposed. This section gives an overview of the most related work to our method.

Marker-based Performance Capture Marker-based performance capture systems are still the most widely adopted solutions for facial performance capture in the industry and have achieved great success in the commercial world. As markers at certain semantically significant locations are attached to the face, it is relatively easy to fit a template mesh to the marker data. The advantage of this technique is that it is fast, robust to noise, and easy to deploy. However, the captured detail is quite low, as measurements are only available at the marker positions.

Photometric Stereo Photometric stereo [91] is an active illumination approach used for surface normals recovery. The normals provide derivative information of the 3D surface and can be used to generate accurate high-frequency local geometry [57]. Recent light stage developments [1, 19] adopt a spherical gradient illumination approach for generating a detailed normal map of the input face. Real-time computation [53] for this approach has been achieved using high-speed video cameras and a fast GPU implementation. Ma et al. [51] applied structured light scanning to capture low-frequency 3D structure, so their hardware system has to be well designed to allow capturing a large number of images in fast succession (13 images = 8 spherical illuminations + 5 structure light patterns per time instance). In contrast, our approach only requires 6 images per time instance for normal map calculation, thus can greatly increase the frame rate. Many extensions have appeared afterwards [29, 52], most related to our work is Wilson et al.'s approach [90], which made two improvements to Ma et al.'s work by firstly reducing the requirements of illumination condition, and secondly exploring dense temporal stereo correspondence for 3D structure reconstruction rather than structured light scanning. With the benefits of these improvements, their system can achieve higher frame rates and also more stable results. However, the aim of these approaches is to generate detailed 3D geometry for every captured frame rather than fitting a consistent template mesh.

3D morphable models 3D morphable model based approaches [6,20,60] can provide useful prior knowledge for marker-based or image-based facial performance capturing. General facial models [5, 79], which are trained on a large database, may miss fine details unique to a specific person, hence, recent developments also focus on subject specific models [89], or single patch representation in region based variants [37, 74]. To build a 3D face model, these approaches require semi-automatic fitting of templates to 3D scanner data with manually selected markers, while our automatic approach relies solely on normal data.

Chapter 3

Pattern Search in Scalar Fields

3.1 Overview

Pattern matching algorithms have proven useful for scalar [28, 34, 48] and vector fields [11, 25, 35, 80]. The general idea is to compute the similarity between a user-supplied pattern and every location in the data set. A pattern can be given in different forms such as a small subset of the domain or a selection of stream lines. A desired property for pattern matching is invariance to translation, rotation, and scaling, i.e., a translated, rotated and scaled copy of the pattern can be found in the data despite these transformations. To achieve this, some existing algorithms require substantial computation time. Furthermore, existing algorithms work for a single field only.

To the best of our knowledge, we present the first pattern matching method in the context of multi-field visualization. Our approach offers fast responses to the user by performing a large part of the pattern search using a sparse set of feature points. Features are computed using the 3D SIFT algorithm [17, 70]. A 3D SIFT feature is located at a point and describes the behavior of the scalar field in its neighborhood with invariance to translation, rotation, and scaling. We compute SIFT features for scalar fields. We deal with vector fields indirectly by means of computing SIFT features for derived scalar fields such as the vorticity magnitude of a flow. Scalar fields for which we compute SIFT features are called *trait fields.*

Pattern matching using our method works as follows: a user selects a search pattern as a region of interest in the domain (a small box). For each SIFT feature within this region, we find similar ones in the entire data set. This is a very fast procedure and yields candidate transformations of the search pattern. In other words, instead of testing the search pattern against *every* other location in the domain (as well as all possible rotations and scale factors), we test it only against a sparse and sufficient set of candidates. The actual similarity value is then computed using a weighted L2-norm over all considered trait fields. The final result is a scalar field that indicates regions in the multi-field data set where all trait fields show a similar behavior as in the user-selected region.

We give the following technical contributions:

- We introduce a novel pattern matching method for 3D multi-field data sets, which bundles the information from different fields into the description of a pattern.

- We achieve response times for pattern matching of less than a second even for large data sets by working with a sparse set of prominent features.
- In contrast to the two previous versions of the 3D SIFT algorithm [17, 70], we achieve full invariance to 3D rotation by finding a robust local coordinate system. This increases the accuracy of pattern matching.

The next section recapitulates the 3D SIFT algorithm and explains our improvement to it. Section 3.3 presents our pattern search algorithm for multi-fields. We evaluate and discuss our method in section 3.4. Results are shown in section 3.5.

3.2 Scale-Invariant Feature Transform

In the following, we recapitulate the basics of the 3D SIFT algorithm based on the work of Scovanner et al. [70] and Cheung et al. [17] as well as the original 2D work of Lowe [48]. Especially the original work comes with a lot of background information both on a theoretical and practical level, which serves as justification for the individual steps of the algorithm and for the parameter choices. In this paper, we follow these choices and refer the interested reader to [48] for more background information.

3.2.1 Background on 3D SIFT

There are two aspects about a SIFT feature: its *location* and its *description.* The computation of these aspects is related, as we shall see in the following.

Consider a 3D scalar field $S(x,y,z)$ on a 3D uniform grid. The scale space of S is constructed using a convolution with a Gaussian blur $G(x,y,z,\sigma)$

$$L(x,y,z,\sigma) = G(x,y,z,\sigma) * S(x,y,z) \tag{3.1}$$

where σ refers to the standard deviation of the Gaussian distribution. Note that σ spans the new scale-dimension and larger σ correspond to blurrier versions of the scalar field. The location of the SIFT features is computed from a derived space, namely the convolution of the scalar field S with the *Difference-of-Gaussian* function

$$D(x,y,z,\sigma) = (G(x,y,z,k\sigma) - G(x,y,z,\sigma)) * S(x,y,z), \tag{3.2}$$

where $k > 1$. This can be computed as the difference between two neighboring scales

$$D(x,y,z,\sigma) = L(x,y,z,k\sigma) - L(x,y,z,\sigma). \tag{3.3}$$

The locations of the SIFT features are the extrema of $D(x,y,z,\sigma)$.

In practice, the fields are constructed as shown in Figure 3.1. One starts with an initial blur; Lowe [48] suggests $\sigma_0 = 1.6$. The following scales are computed using a repeated convolution with $G(x,y,z,k)$. The value of k is set such that a doubling of σ is achieved after a certain number s of steps. Doubling σ is referred to as an *octave*.

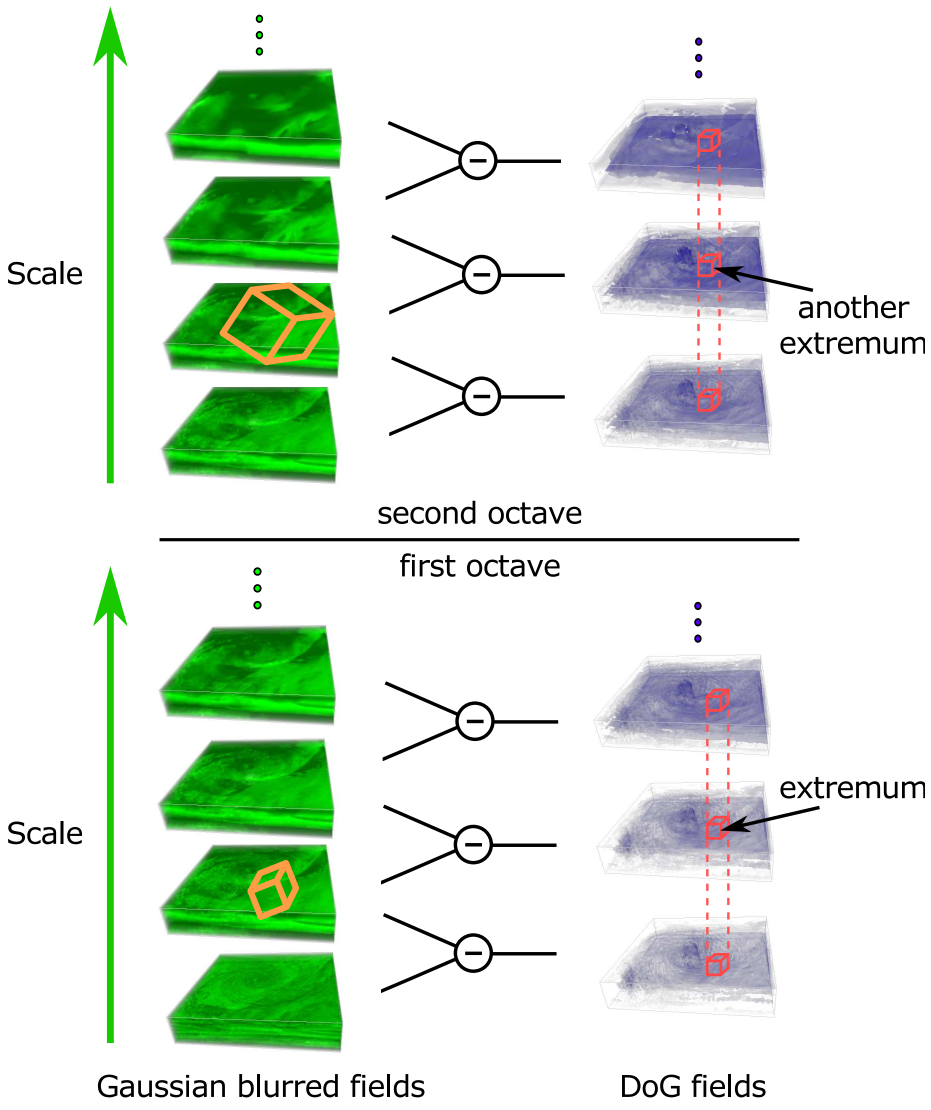


Figure 3.1: Localization of SIFT features in different scales. A sequence of Gaussian blurred fields with increasing σ is generated. DoG (Difference-of-Gaussian) fields are computed by subtracting neighboring blurred fields. The extrema in the DoG fields denote the SIFT feature locations. For a given location, the neighborhood in the blurred field of the corresponding scale (orange box) serves as a description of the SIFT feature (see also Figure 3.2).

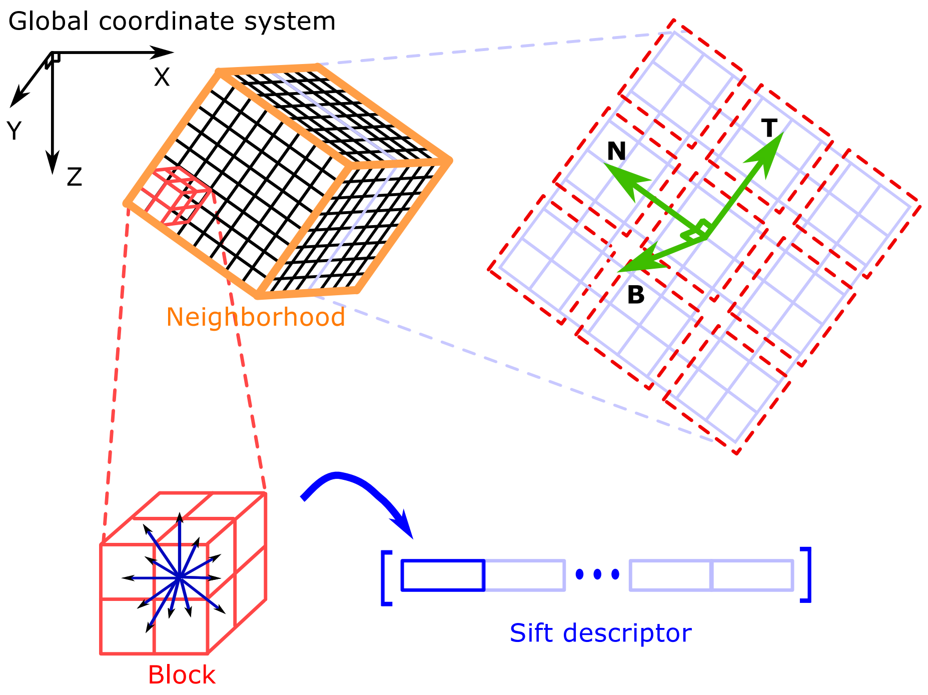


Figure 3.2: Computation of the SIFT descriptor. Based on a proper orientation (green arrows), a neighborhood of size $9 \times 9 \times 9$ is sampled in the blurred scalar field (see also Figure 3.1). It is split into 27 blocks. For each block, a 3D histogram of the gradient is computed and stored in the SIFT descriptor.

Lowe [48] suggests $k = 2^{1/s}$ with $s = 3$. The Difference-of-Gaussian (DoG) fields are computed from two neighboring scales as shown in Figure 3.1. Note that the total number of blurred scalar fields needs to be $s+3$ such that the SIFT feature locations can later be computed from a full octave. The next octave starts with $2\sigma_0$ and a halved resolution. We use a total of three octaves throughout the paper.

The *locations* of the SIFT features are found as extrema in the DoG fields. More precisely, a grid vertex in the DoG field is marked as an extremum if all its grid neighbors in the previous, current and next scale are smaller/larger, i.e., a $3 \times 3 \times 3 \times 3$ neighborhood is checked. See the right part of Figure 3.1 for an illustration. It does not matter whether an extremum is a minimum or a maximum.

The *description* of a SIFT feature is derived from a neighborhood in the blurred scalar fields. This is illustrated in Figure 3.1 as orange boxes over the blurred fields. Note that the effective physical size of these boxes is larger in higher octaves due to the changed resolution.

The computation of the SIFT descriptor is illustrated in Figure 3.2. The key here is to achieve rotation invariance by choosing a consistent orientation for the neighborhood box based on local properties of the data. For 2D SIFT features, Lowe [48] uses the local image gradient, i.e., a 2D vector, which uniquely defines the orientation of a 2D neighborhood. For the 3D case, the existing methods by Scovanner et al. [70] and Cheung et al. [17] use the 3D gradient vector as orientation. More precisely, they parameterize it to the spherical coordinate system and compute a 3D rotation matrix from that. We will show in the next section that this does *not* define the orientation of a 3D neighborhood in a unique manner.

However, after assigning an orientation, a neighborhood with size $9 \times 9 \times 9$ is considered around the extrema position. It is split into 27 small blocks with size $3 \times 3 \times 3$ each. For each block, the gradient of the blurred scalar field is sampled at the vertices and recorded in a 3D orientation histogram. It has 12 bins as defined by a Platonic icosahedron. The histogram records the magnitude of the gradient. Each histogram comprises a part of the SIFT descriptor. The complete SIFT descriptor has $12 \times 27 = 324$ elements. The descriptor is considered a vector and normalized to have unit length. The process of computing a SIFT descriptor is shown in Figure 3.2.

Euclidean distance is used to measure the cost (dissimilarity) between two SIFT descriptors **k** and **m**:

$$\text{cost}(\mathbf{k}, \mathbf{m}) = ||\mathbf{k} - \mathbf{m}||. \tag{3.4}$$

This direct and fast comparison is possible, since we transformed them already into a common space w.r.t. rotation and scale. Two SIFT descriptors are said to be *matching*, if their cost is below a certain threshold. This parameter is quite unproblematic in our setting since a larger set of matching features will just have a slight impact on running time. We fixed it to 0.2 in our implementation, where 0 means that the two descriptors are identical, and 2 is the largest possible distance between two diametrically opposed descriptors.[1]

[1] Since the descriptors are normalized to unit length, two descriptors **k** and **m** have a cost of 2, iff $\mathbf{k} = -\mathbf{m}$.

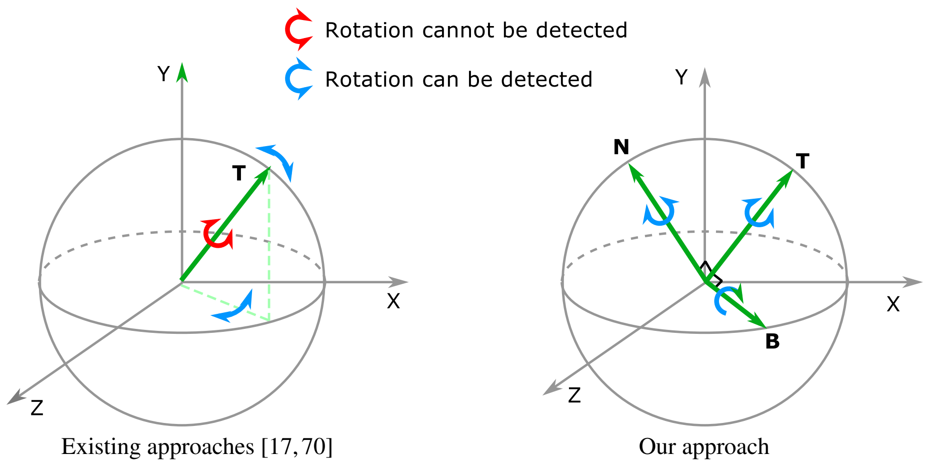


Figure 3.3: Rotation invariance of the SIFT descriptor can only be achieved, if the 3D orientation of the neighborhood is properly fixed. See also Figure 3.2. Previous approaches attempt this using a single vector encoded in spherical coordinates. The result is sensitive to rotation *around* this vector itself. We compute a stable local coordinate system using (3.10), which makes the 3D SIFT descriptor properly rotation invariant.

3.2.2 Obtaining Rotation Invariance for 3D SIFT

A single vector does not suffice to fix the orientation of a 3D neighborhood. Figure 3.3 shows how the existing methods by Scovanner et al. [70] and Cheung et al. [17] use the gradient direction **T** to fix two out of three possible rotations. Specifically, any rotation around **T** cannot be detected with their SIFT descriptors.

We fix this issue by computing the Frenet-Serret frame in the gradient field as the base coordinate system. This gives us a local, orthogonal coordinate system. We use it to assign a unique orientation to the neighborhood box before computing the SIFT descriptor.

The Frenet-Serret frame has three orthogonal axes, i.e., tangent **T**, normal **N**, and bi-normal **B**. They are computed from local derivatives, but conceptually they refer to the tangent, normal and bi-normal of the gradient curve **r** passing through a given point:

$$\mathbf{T} = \frac{d\mathbf{r}}{ds} \tag{3.5}$$

$$\mathbf{N} = \frac{d\mathbf{T}}{ds} \tag{3.6}$$

$$\mathbf{B} = \mathbf{T} \times \mathbf{N}. \tag{3.7}$$

To make the frame more reliable, we compute the averages of the tangent, normal and bi-normal of all the Frenet-Serret frames within a $3 \times 3 \times 3$ neighborhood. Let us denote them with $\overline{\mathbf{T}}$, $\overline{\mathbf{N}}$, and $\overline{\mathbf{B}}$. To obtain a orthogonal coordinate system, we apply the

Gram-Schmidt process, which can be written as

$$\mathbf{T}' = \overline{\mathbf{T}} \tag{3.8}$$

$$\mathbf{N}' = \overline{\mathbf{N}} - \frac{\overline{\mathbf{N}} \cdot \mathbf{T}'}{\mathbf{T}' \cdot \mathbf{T}'} \mathbf{T}' \tag{3.9}$$

$$\mathbf{B}' = \overline{\mathbf{B}} - \frac{\overline{\mathbf{B}} \cdot \mathbf{T}'}{\mathbf{T}' \cdot \mathbf{T}'} \mathbf{T}' - \frac{\overline{\mathbf{B}} \cdot \mathbf{N}'}{\mathbf{N}' \cdot \mathbf{N}'} \mathbf{N}', \tag{3.10}$$

where $\mathbf{T}'$, $\mathbf{N}'$, and $\mathbf{B}'$ are further normalized to have unit lengths.

This coordinate system gives us a reliable basis for a rotation-invariant SIFT descriptor. Figure 3.3 gives an illustration. We tested the rotation invariance in a practical setting, see Section 3.4.5.

3.3 Pattern Matching in Multi-Fields

We define a *pattern* as a user-defined 3D box in the data. It is characterized by its orientation, extent, location, and, most importantly, by the data values of the individual fields in the multi-field data set within this box. To match this pattern means to find locations in the domain where the individual fields of the multi-field data set attain similar values. Figure 3.4 depicts the algorithmic pipeline.

In the following, we show how we use the SIFT features from the previous section to effectively and efficiently match such patterns. Section 3.3.2 discusses the first stage of our algorithm, which is feature-based. Section 3.3.3 deals with the subsequent region-based part that leads to a dense output: a scalar field giving the similarity between the pattern and any other location. Before we come to that, we will first discuss how to incorporate vector fields into our setup in the next section.

3.3.1 Trait fields

SIFT features are defined for scalar fields. We are not aware of an extension to vector fields. For the purpose of this paper, we choose to incorporate vector fields indirectly by computing SIFT features for derived scalar fields. These fields can be directly incorporated into our multi-field approach. In general, all scalar fields for which we compute SIFT features will be called *trait fields* in the following, as they bear a characteristic trait of the underlying multi-field.

For a 3D vector field $\mathbf{v} = (u, v, w)^T$ with its Jacobian $\mathbf{J} = [\mathbf{v}_x \mathbf{v}_y \mathbf{v}_z]$, we consider the following trait fields, which reflect important characteristics of $\mathbf{v}$:

- magnitude $||\mathbf{v}||$
- norm of the Jacobian $||\frac{d\mathbf{v}}{dx} \frac{d\mathbf{v}}{dy} \frac{d\mathbf{v}}{dz}||$
- divergence $u_x + v_y + w_z$
- vorticity magnitude $||(w_y - v_z, u_z - w_x, v_x - u_y)^T||$
- helicity $u(w_y - v_z) + v(u_z - w_x) + w(v_x - u_y)$
- λ_2-criterion [41]
- Okubo-Weiss criterion [39]

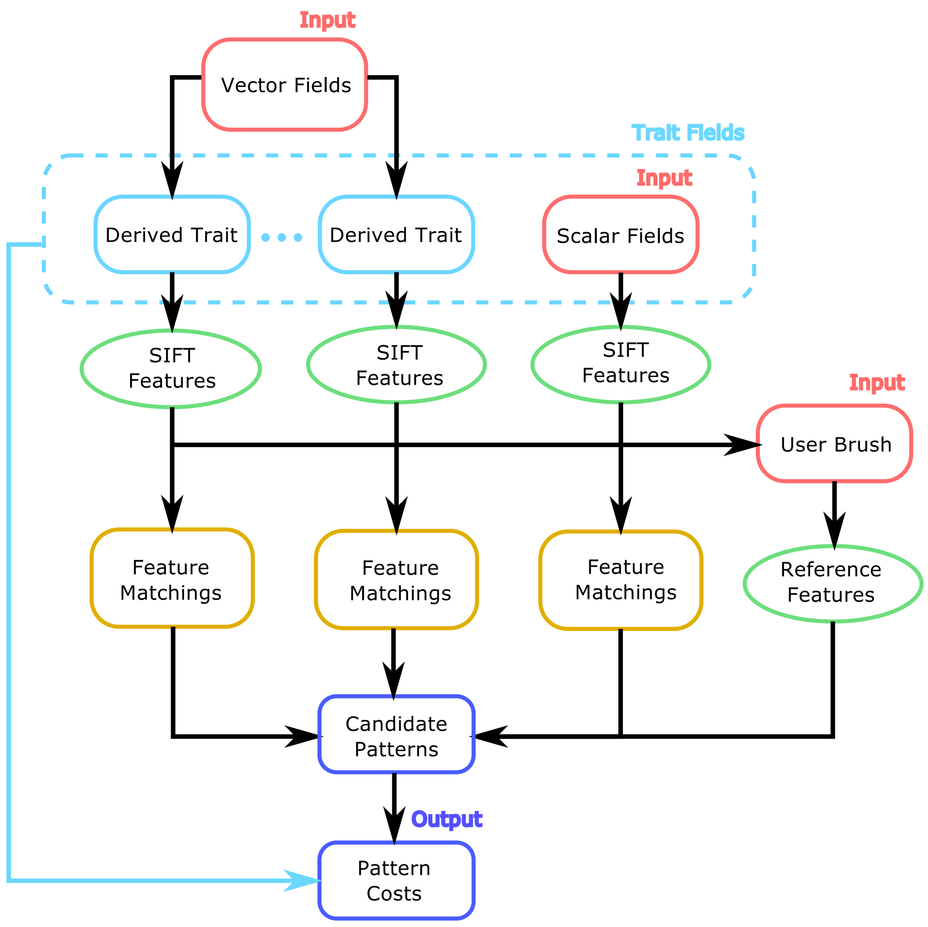
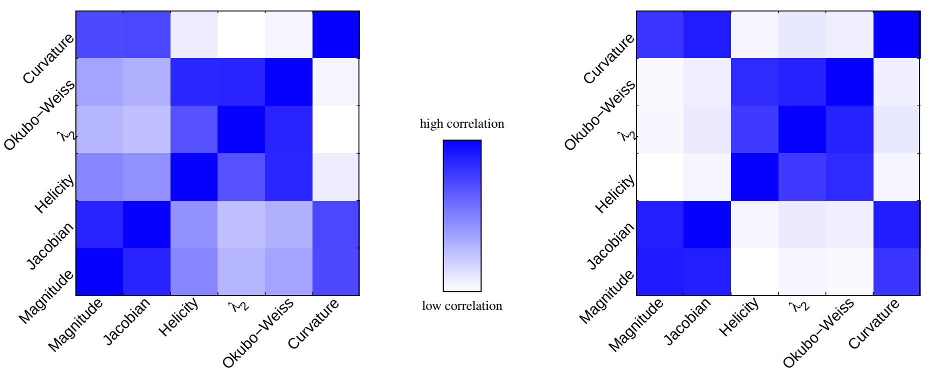


Figure 3.4: The pipeline of the proposed algorithm. Firstly, vector fields and scalar fields are all converted to trait fields. Later, SIFT features are extracted from each trait field independently. We select SIFT features which have intersection with the user brushed box as the reference features. For each reference feature, each of its matchers within a cost threshold determines a location of the candidate pattern regarding to the transformation between features. The actual pattern cost for each candidate pattern is the weighted sum of the costs in all trait fields.



(a) Borromean data set. (b) Rayleigh-Bénard flow.

Figure 3.5: Trait correlations. Some traits correlate strongly with each other (dark blue patches).

- curvature of the stream lines [85].

Some of these trait fields will only make sense in some applications. As an example, most fluid simulations are executed under the assumption of a divergence-free fluid, i.e., the divergence field is constant zero. Other trait fields may be highly correlated and therefore redundant. For example, both the λ_2-criterion and the Okubo-Weiss criterion describe vortex structures in flows.

In essence, each application has to decide on the specific set of trait fields. To aid such a decision, we provide some guidance in Figure 3.5. Here, we computed the pairwise correlation between trait fields. We employ the *gradient similarity* measure (GSIM) with the correlation characteristic ac_N as discussed in Sauber et al. [67]. The correlation patterns are rather similar for the two examined vector fields in Figure 3.5. Helicity, λ_2, and Okubo-Weiss are highly correlated. A high correlation can also be observed between magnitude, the norm of the Jacobian, and the stream line curvature. In our experiments, we often choose just one member of such a correlation group for the actual pattern matching.

A similar approach can also be taken to incorporate tensor fields. We leave this for future work.

3.3.2 Feature-Based Search for Candidate Patterns

Consider a user-brushed 3D box as the reference pattern $\mathbf{P}$. We define its center $\mathbf{c_P}$ as the geometric center of the box, its orientation $\mathbf{O_P}$ is given by the world coordinate system, and the scale is given as $s_\mathbf{P} = 1$.

The following procedure is carried out individually for each trait field. We define the set of *reference SIFT features* $\mathbf{K}$ as those SIFT features whose supporting neighborhood fully or partly overlaps with the box of the reference pattern $\mathbf{P}$. A SIFT feature $\mathbf{k} \in \mathbf{K}$ comes with a position $\mathbf{c_k}$, an orientation of its neighborhood $\mathbf{O_k} = (\mathbf{T}', \mathbf{N}', \mathbf{B}')$, and a scale $s_\mathbf{k}$.

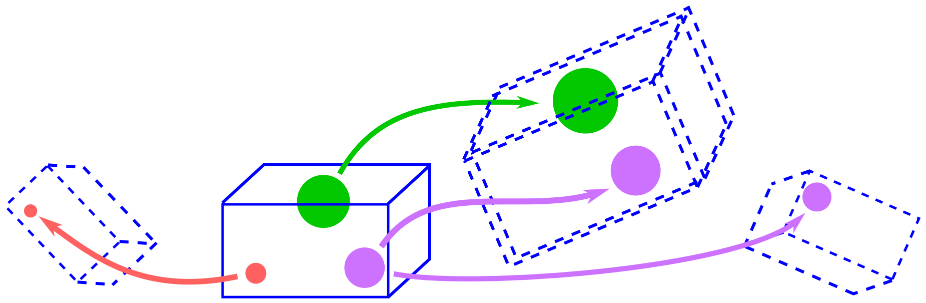


Figure 3.6: Finding candidate patterns from matching SIFT features. The solid blue box denotes the user-defined reference pattern **P**. The colored circles denote different types and scales of SIFT features. The SIFT features overlapping with **P** are called reference features. Matching them to other SIFT features in the data set leads to transforming the reference pattern **P** to several candidate patterns **P**′, shown as dashed boxes. A subsequent region-based cost computation yields the final result.

For each $\mathbf{k} \in \mathbf{K}$, we find the set of matching SIFT features $\mathbf{M_k}$ (cf. Section 3.2). Only SIFT features from the same trait field are compared with each other. Again, each SIFT feature $\mathbf{m} \in \mathbf{M_k}$ comes with a position $\mathbf{c_m}$, an orientation $\mathbf{O_m}$, and a scale $s_\mathbf{m}$.

Given $(\mathbf{c_k}, \mathbf{O_k}, s_\mathbf{k})$ and $(\mathbf{c_m}, \mathbf{O_m}, s_\mathbf{m})$, we can compute a linear transformation that maps the neighborhood box of **k** to the neighborhood box of **m**. We apply this transformation to the user-defined reference pattern **P** and obtain a *candidate pattern* **P**′ by computing its center $\mathbf{c_{P'}}$, orientation $\mathbf{O_{P'}}$, and scale $s_\mathbf{P'}$ as follows:

$$\mathbf{c_{P'}} = \mathbf{c_m} + \mathbf{O_m}(\mathbf{Iv}) \tag{3.11}$$

$$\mathbf{O_{P'}} = \mathbf{O_m}\mathbf{O_k}^{-1} \tag{3.12}$$

$$s_\mathbf{P'} = \frac{s_\mathbf{m}}{s_\mathbf{k}} \tag{3.13}$$

$$\text{with} \quad \mathbf{v} = s_\mathbf{P'}\,\mathbf{O_k}^{-1} \cdot (\mathbf{c_P} - \mathbf{c_k}), \tag{3.14}$$

where **I** is the identity matrix. Figure 3.6 depicts how we find candidate patterns from matching SIFT features.

3.3.3 Region-Based Cost Computation

After matching the SIFT features, we have a number of candidate patterns. They are transformed versions of the reference pattern, i.e., translated, rotated, and scaled 3D boxes. Each candidate pattern has been computed using the SIFT features of a specific trait field. We disregard this information now. We are only interested in the boxes themselves. More precisely, we want to know whether the trait fields attain similar values in a candidate pattern versus the reference pattern.

We compute the cost between the reference pattern and a candidate pattern in each trait field using the L2-norm within these boxes. The final cost is then a weighted sum of the individual costs:

$$\text{cost}(\mathbf{P}') = \sum_t \sum_i w_t ||\mathbf{P}_{it} - \mathbf{P}'_{it}||^2, \tag{3.15}$$

where t denotes the trait field, i is the index of the grid vertices of $\mathbf{P}$, and w_t is the importance weight for trait field t.

The costs of all candidate patterns are combined in one global scalar field, which we use to visualize the matching result.

3.4 Evaluation and Discussion

3.4.1 Rationale Behind the Feature-Based Approach

A straightforward way of computing the cost scalar field without the help of SIFT features would be the following: We could sample the space of all possible rotations, translations, and scales that can be applied to the reference pattern. After transforming the reference pattern, we would evaluate (3.15) for each transformed box and combine all these costs in the global cost scalar field.

A simple example shows that this approach is more time-consuming than our matching using SIFT features. Consider a single scalar field such as the electrostatic potential of the Benzene molecule shown in Figure 3.22. It is sampled on a 257^3 grid. In order to shift the reference pattern to every grid vertex, we require just as many translations. Sampling 3D rotations is not as straightforward as many sampling schemes typically oversample the polar regions. Such matters are discussed in the Robotics community; Kuffner [43] shows how unit quaternions help in uniformly sampling rotations. Let us assume that 100 rotation samples allow for enough accuracy. Finally, let us use the same 18 scale samples we use in our approach (see Section 3.2). We end up with a total of over 30 billion transformations.

On the other hand, our approach uses sparse feature sampling to drastically decrease the number of considered pattern transformations. The numbers are shown in Table 3.1. For the Benzene data set, we have 38 SIFT features and the pattern overlaps with 8 of those. We have to compare those 8 features with the others, and create a transformed candidate pattern if they match. The entire pipeline including the cost computation is done in under 0.2 seconds. The detection of the SIFT features themselves takes about 80 seconds, but they can be reused for any pattern.

Our computation time depends on the total number of SIFT features n, and the number of selected SIFT features m, where $m \leq n$ but typically $m \ll n$. Every selected feature is compared against all others. Our current implementation does this straightforwardly using linear search with a computational complexity of $O(mn)$. For very large numbers of SIFT features, it may be beneficial to use a hashing function for a faster comparison. We refer the interested reader to the extensive discussions of this topic by Paulevé et al. [58] as well as Muja and Lowe [56]. The latter comes with a public domain software library for this purpose.

For very small numbers of SIFT features, one may have the issue that parts of the domain are not covered and therefore unavailable for defining patterns. We did not encounter this issue in our experiments, and deem it rather negligible for two reasons:

- Consider a region in a scalar field with uniform behavior, i.e., constant scalar value or no monotony breaks. Such a region is typically of low interest to a user and will also not have SIFT features. On the contrary, SIFT features indicate regions with non-uniform behavior in a scalar field, which are typically the regions of highest interest in an analysis.

- Our method works with multi-fields. A lack or sparsity of SIFT features in one trait field is compensated by the SIFT features of the other trait fields. In fact, our method needs only a single SIFT feature in the user-defined reference pattern **P**.

Table 3.1 shows that we find a high number of SIFT features in all our experiments. Figure 3.23 shows a number of trait fields with their SIFT features, some of which are covered densely and some sparsely.

Ultimately, it is a data- and application-dependent question whether the patterns of interest are covered by SIFT features. We will discuss this in the next section for the special case of pattern matching in vector fields.

3.4.2 Comparison to Vector-Based Matching Methods

In the following, we compare our method to the pattern matching methods for vector fields due to Ebling et al. [25] and Heiberg et al. [35]. We implemented both approaches in our system.

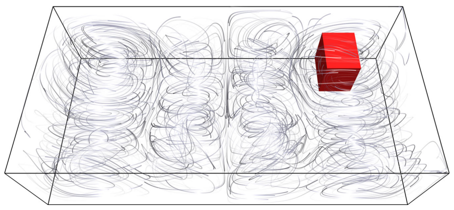
The major difference is that the above approaches work directly and only on vector-valued data, whereas our method deals with vector fields indirectly by considering multiple derived scalar fields. This has consequences when analyzing numerical flow simulations:

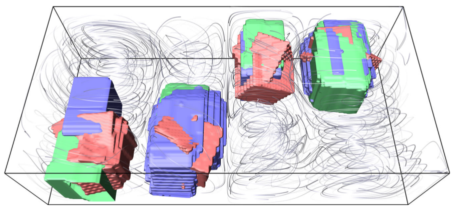
- Flow data sets are multi-fields. They contain the vector-valued flow velocity as well as scalar-valued traits such as viscosity, pressure, and density. Our method is able to incorporate these scalar fields, at the expense of treating the velocity field indirectly.

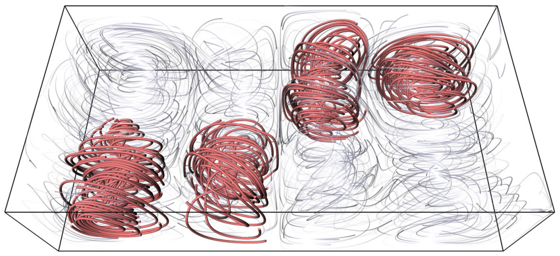
- The matching results of Ebling et al. [25] and Heiberg et al. [35] are not Galilean invariant, since they are directly obtained from the velocity vectors. Many interesting flow features such as the von Kármán vortex street (cf. Figure 3.28) can only be observed in the velocity field when choosing a particular frame of reference. This is often not trivial. On the other hand, our method incorporates Galilean invariant scalar fields and is therefore able to detect many flow features directly.

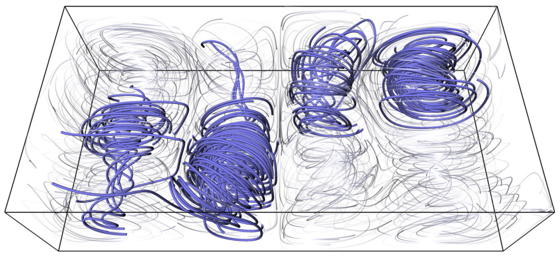
We devised a fair comparison to Ebling et al. [25] and Heiberg et al. [35] by concentrating on a single vector field and selecting a feature that can be observed in the original frame of reference. We chose the Rayleigh-Bénard flow as shown in Figure 3.7. It is a flow due to thermal convection of heated and cooled boundaries, obtained using the software NaSt3DGP [31][2]. The flow has 8 vortex structures. Half of them have a left-handed

[2]NaSt3DGP was developed by the research group in the Division of Scientific Computing and Numerical Simulation at the University of Bonn.


(a) Selection of a vortex with a left-handed sense of rotation.


(b) Detections of all three methods overlaid.


(c) Stream lines seeded in the voxels detected by our method. Computation time for the pattern matching is 8 s, and requires the reusable SIFT features (49 s).


(d) Stream lines seeded in the voxels detected by Ebling et al. [25]. Computation time: 284 s.

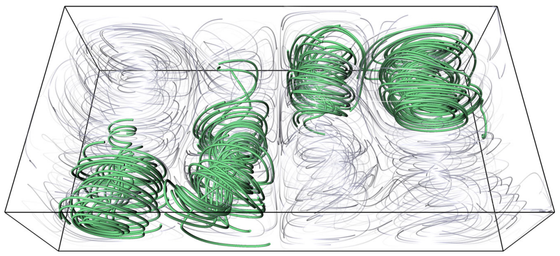

(e) Stream lines seeded in the voxels detected by Heiberg et al. [35]. Computation time: 158 s.

Figure 3.7: Comparison of the results using our multi-field method and the vector-based methods from Ebling et al. [25] and Heiberg et al. [35]. Shown is the Rayleigh-Bénard convection flow. All three methods correctly identify the four vortices with left-handed sense of rotation, but require significantly different computation times.

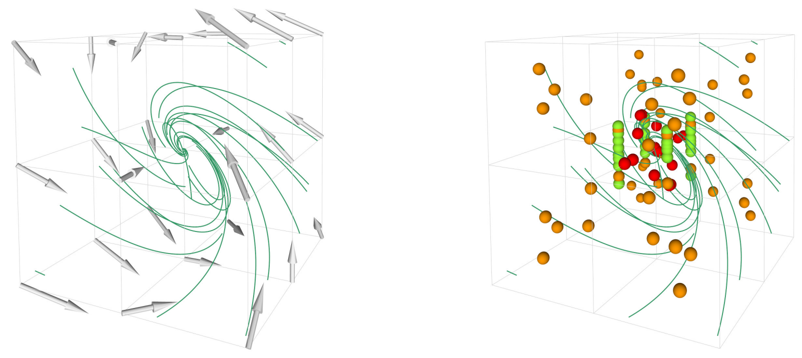

(a) Pattern. (b) SIFT features.

Figure 3.8: Attracting Focus.

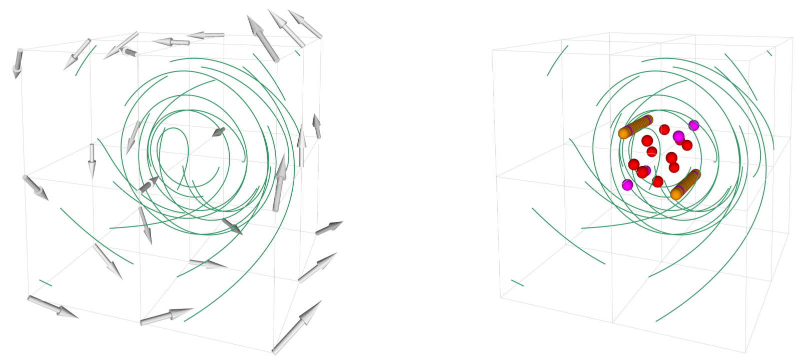

(a) Pattern. (b) SIFT features.

Figure 3.9: Attracting Focus Saddle.

sense of rotation, and the other half a right-handed one. We selected a vortex with a left-handed sense of rotation as the search pattern (Figure 3.7a).

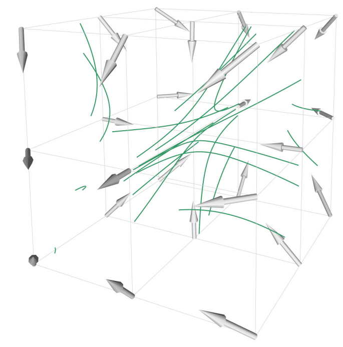
All three methods correctly identify the 4 left-handed vortices.[3] Figures 3.7b–e show the matched regions as well as stream line renderings highlighting these vortices.[4]

The computation times are rather different. Our method needs 8 seconds for the pattern matching itself, and 49 seconds for the pre-computation of the SIFT features in seven trait fields. Note that these SIFT features can be reused for further pattern matching in this flow, i.e., to identify the right-handed vortices. The vector-based pattern matching methods need significantly more computation time. The method of Ebling et al. [25] requires 284 seconds, the method of Heiberg et al. [35] requires 158 seconds.

As a second experiment, we designed 12 vector field patterns in the spirit of Ebling

[3]As a side note, our method picks up on the rotation sense due to the *Helicity* trait field (the sign gives the chirality), whereas the other methods detect the rotation sense from the orientation of the velocity vectors.

[4]Note that we seeded the stream lines in the matched regions, but their integration was unrestricted.


(a) Pattern.

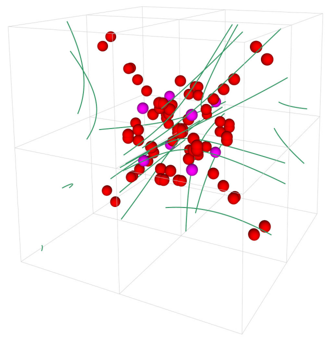
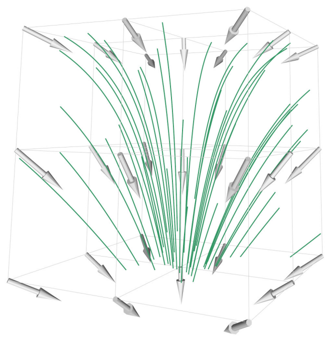

(b) SIFT features.

Figure 3.10: Attracting Saddle.


(a) Pattern.

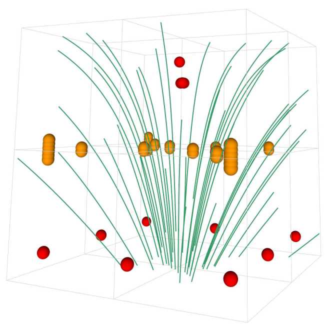
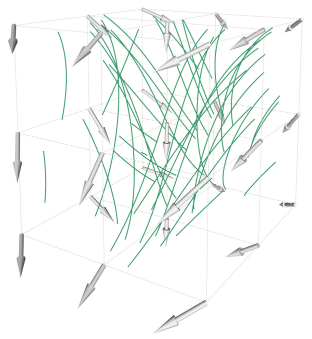

(b) SIFT features.

Figure 3.11: Convergence.


(a) Pattern.

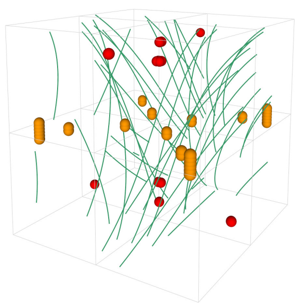
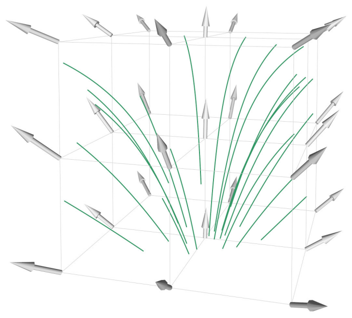

(b) SIFT features.

Figure 3.12: Convergence Divergence.


(a) Pattern.

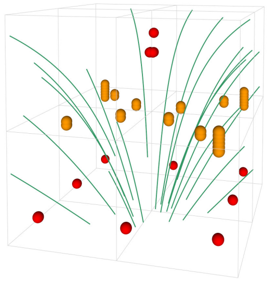
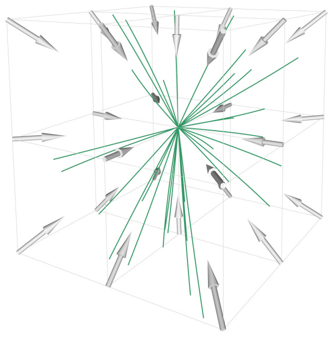

(b) SIFT features.

Figure 3.13: Divergence.


(a) Pattern.

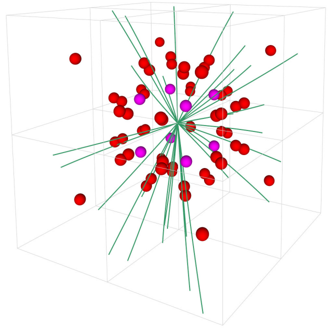
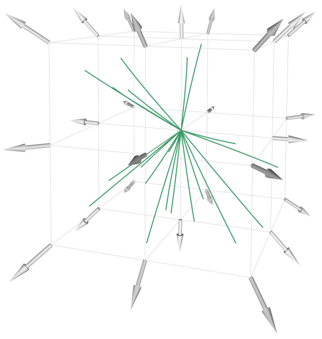

(b) SIFT features.

Figure 3.14: Node Sink.


(a) Pattern.

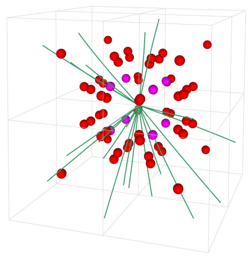
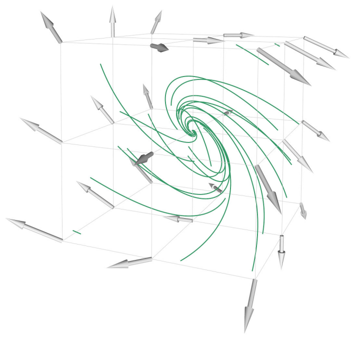

(b) SIFT features.

Figure 3.15: Node Source.


(a) Pattern.

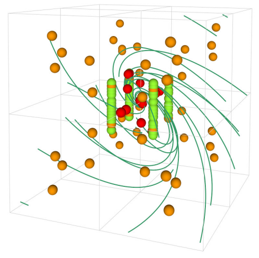
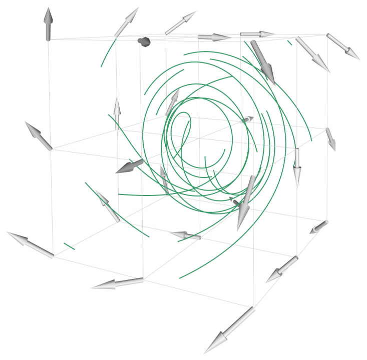

(b) SIFT features.

Figure 3.16: Repelling Focus.


(a) Pattern.

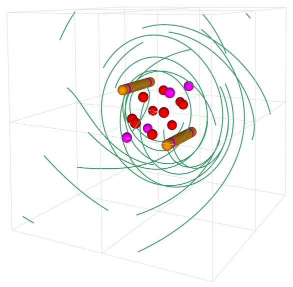
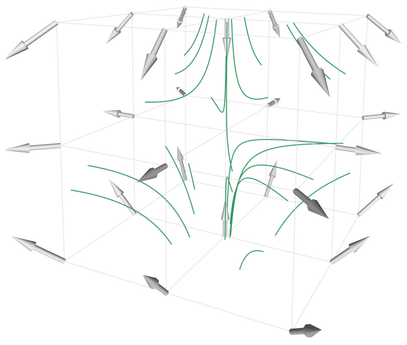

(b) SIFT features.

Figure 3.17: Repelling Focus Saddle.


(a) Pattern.

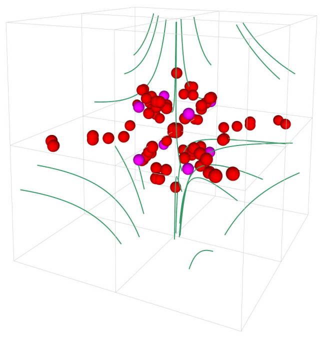
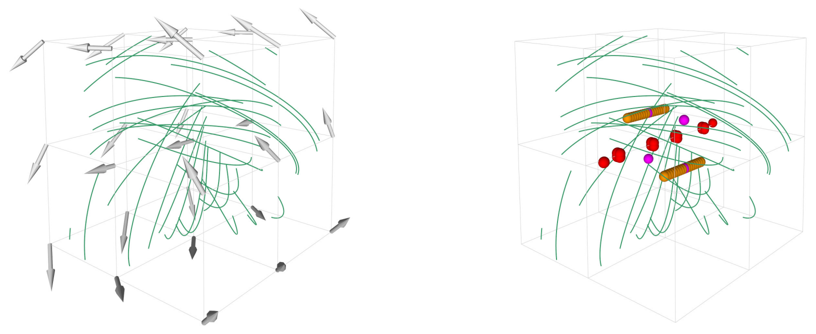

(b) SIFT features.

Figure 3.18: Repelling Saddle.



(a) Pattern. (b) SIFT features.

Figure 3.19: Vortex.

et al. [25] and Heiberg et al. [35], including vortices, convergent/divergent flow, and all first-order critical points. We made sure to include all patterns mentioned in these papers. As we show in Figure 3.8-3.19, we find SIFT features for all these patterns. Hence, our method can handle the same vector field patterns as these previous methods.

There is one exception. One pattern cannot be observed with our method: parallel flow as in $\mathbf{v}(\mathbf{x}) = (1,0,0)^T$. Such a flow does not contain any flow features and all conceivable derived scalar fields do not contain SIFT features.

3.4.3 Discussion of False Negatives and False Positives

A false negative is a pattern $\mathbf{P}'$ that has not been found despite it being a translated, rotated, and scaled copy of the reference pattern $\mathbf{P}$. It is easy to see that our algorithm *cannot* have false negatives: Each SIFT feature $\mathbf{k}$ in the reference pattern $\mathbf{P}$ has a corresponding SIFT feature $\mathbf{k}'$ in $\mathbf{P}'$, because SIFT features are exceptionally invariant to translation, rotation, and scaling as we show in Section 3.4.5. The SIFT features $\mathbf{k}$ and $\mathbf{k}'$ are practically identical, which makes it easy to find $\mathbf{P}'$.

It becomes more interesting when noise or other deformations cause a difference between $\mathbf{P}$ and $\mathbf{P}'$. SIFT features react gradually to such changes (see the noise experiment in Section 3.4.5), i.e., the difference between $\mathbf{k}$ and $\mathbf{k}'$ is comparable to the difference between $\mathbf{P}$ and $\mathbf{P}'$. Considering an increasing difference between $\mathbf{k}$ and $\mathbf{k}'$, we will stop computing the cost between $\mathbf{P}$ and $\mathbf{P}'$ after exceeding a certain threshold. As already discussed in Section 3.2.1, we can afford a larger threshold, since this will just have a slight impact on running time.

A false positive is a pattern $\mathbf{B}$ that has been found despite it being rather different from the reference pattern $\mathbf{P}$. It is easy to see that our algorithm *cannot* have false positives: Consider that $\mathbf{B}$ and $\mathbf{P}$ have no matching SIFT features. Then $\mathbf{B}$ will also not be considered a matching pattern of $\mathbf{P}$. Alternatively, consider that $\mathbf{B}$ and $\mathbf{P}$ have matching SIFT features. Then the region-based cost computation (3.15) will return the actual cost between these two patterns.

3.4.4 Discussion of Parameters

The following parameters pertain to the computation of the SIFT features. We list their values here and refer to the literature [48] for background information. See also Section 3.2.

- Number of octaves: 3
- Number of Gaussian blurred fields: 6
- Initial blur $\sigma_0 = 1.6$
- Factor for subsequent blurring $k = 2^{1/3}$
- DoG extrema neighborhood: $3 \times 3 \times 3 \times 3$
- SIFT descriptor neighborhood: $27 \times 27 \times 27$
- Number of SIFT histograms: $3 \times 3 \times 3$
- Number of bins per histogram: 12

Two parameters pertain to the core of our method:

SIFT descriptor matching cost threshold Two SIFT descriptors match, if their cost from Equation (3.4) is below this threshold. Increasing this threshold gives more matching SIFT features, which leads to more region-based cost computations using Equation (3.15), i.e., it has a slight impact on performance, but not on the quality of the matching. Decreasing this threshold gives less matching SIFT features, which imposes less flexibility for deformation of patterns other than translation, rotation and scaling. We suggest to err on the side of increasing this threshold.

Pattern matching cost threshold After computing the region-based costs using Equation (3.15), this threshold is defined by the user for the isosurface or volume rendering visualizations showing the matching patterns. Examples are shown throughout the paper, e.g., in Figures 3.7b and 3.22b. Increasing this threshold shows more matching patterns, decreasing it shows less.

3.4.5 Evaluation of the Invariance of the SIFT Features

We made the following experiment to evaluate how invariant the 3D SIFT features are under rotation, translation, scaling, and noise. Figure 3.20 shows the setup: a single scalar field with values in the range $[0,1]$ and the isosurface at 0.5 is a round, axis-aligned box. This field has 8 SIFT features corresponding to the corners of the cube. Let us denote this set with $\mathbf{A}$.

After transforming or adding noise to the scalar field, we compute the set of new SIFT features $\mathbf{B}$. We compare the sets $\mathbf{A}$ and $\mathbf{B}$ using Hausdorff distance $H(\mathbf{A},\mathbf{B})$:

$$D(\mathbf{x},\mathbf{Y}) = \min\{\mathrm{cost}(\mathbf{x},\mathbf{y}) \mid \mathbf{y} \in \mathbf{Y}\} \tag{3.16}$$

$$H(\mathbf{A},\mathbf{B}) = \max\{\max\{D(\mathbf{a},\mathbf{B}) \mid \mathbf{a} \in \mathbf{A}\}, \max\{D(\mathbf{b},\mathbf{A}) \mid \mathbf{b} \in \mathbf{B}\}\} \tag{3.17}$$

Figure 3.21 shows the results. For interpretation, remember that the SIFT feature descriptors have unit length, i.e., the largest possible cost is 2. We will detail the experiments in the following.

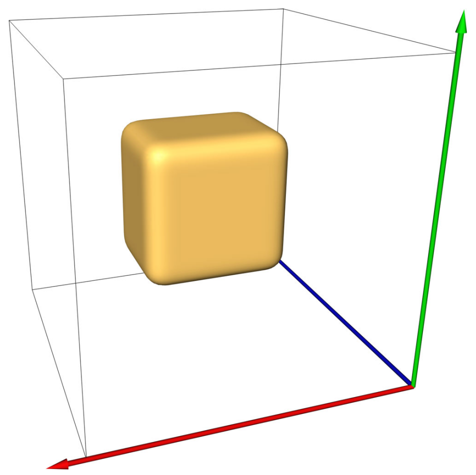


Figure 3.20: Setup for the evaluation of the invariance of the SIFT features. The scalar field has values in the range $[0,1]$ and the isosurface at 0.5 is a round, axis-aligned box. For the evaluation, we rotate, translate and scale the domain as well as adding noise to the data. Results are shown in Figure 3.21.

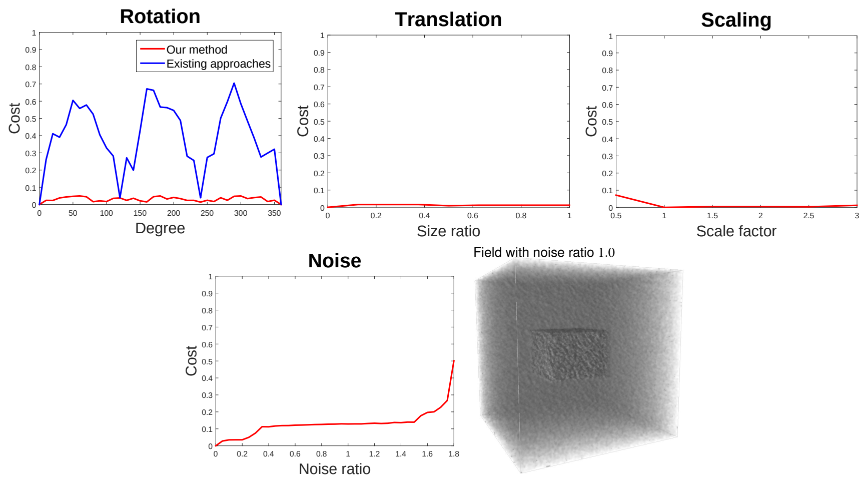


Figure 3.21: Evaluation of the invariance of the SIFT features against rotation, translation, scaling, and adding noise. The setup is shown in Figure 3.20. For the rotation evaluation, we included the results of existing approaches [17, 70]. In these plots, lower values are better.

Rotation We rotate the domain in steps of 10 degrees around the axis $(1,1,1)^T$. We also made this experiment with the existing approaches [17, 70]. As discussed earlier, they are not fully invariant against rotation as the blue curve shows. Our results are shown by the red curve and show a high rotation invariance.

Translation We translate the domain along the x-axis until the shifted distance reaches the size of the box. The result shows the expected high invariance against translation.

Scaling We scale the domain uniformly with the factors $[0.5, 1.0, 1.5, 2.0, 2.5, 3.0]$. The cost between 1.0 and 3.0 is constant zero, which shows full scale invariance. The cost is slightly higher for the factor 0.5, because when the box becomes small enough, the largest scale in the scale space sees the corners as one feature.

Noise We add white noise with increasing amplitude to the data. The shown *noise ratio* refers to the amplitude. A noise ratio of 1 means that the value range of the noise and the data are equal. The cost remains quite low until a noise ratio of 1.5. After that, the data is corrupted and the cost increases rapidly. As it can be seen in Figure 3.21, a noise ratio of 1 creates already a highly distorted field, yet the cost is still within an acceptable range. This shows how stable the SIFT features are against noise.

3.4.6 Invariance against Intensity Scaling or Shifting

The invariances discussed above relate to transforming the domain. What about transformations of the *data values* such as a scaling or shifting? SIFT features are naturally invariant against it, since the SIFT descriptor contains only gradient information and is normalized.

In detail: A multiplication of the scalar field with a constant factor (value scaling) changes only the magnitude of the gradient and not its direction. The normalization of the SIFT descriptor makes it invariant against this. An addition of a constant value (value shifting) does not change the gradient of the scalar field at all. Hence, the SIFT descriptor is invariant against it. Finally, the locations of SIFT features are computed as extrema of the DoG fields, which are unaffected by these transformations.

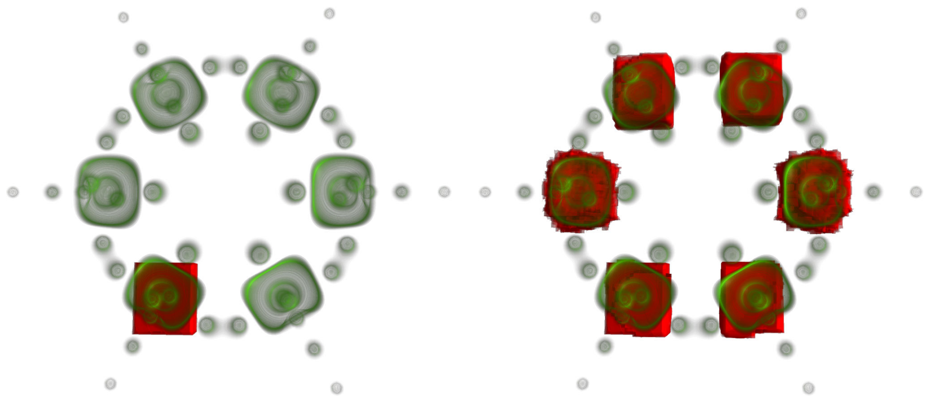
3.5 Results

All results have been computed in a single thread on a 3.1 GHz Intel Xeon E31225 with 16 GB main memory. Computation times as well as the number of SIFT features are shown in Table 3.1.

Benzene We start with a single scalar field to showcase the matching qualities of our algorithm in a setting that is easy to understand. The electrostatic potential of the Benzene molecule in Figure 3.22 exhibits a 6-fold symmetry. We selected the area around one of the six carbon atoms that make up the inner ring of this molecule. The

Data set	Dims	# Traits	# SIFT features		Timings in Seconds			
			total	selected	SIFT localization	SIFT Descriptor	Feature Matching	Cost Computation
Benzene	$257 \times 257 \times 257$	1	38	8	82	0.002	0.001	0.154
Borromean	$256 \times 256 \times 256$	6	5773	92	6×120	0.1 - 0.8	0.008	8.5
				45			0.006	2.5
Climate	$480 \times 241 \times 27$	3	737	20	3×5	0.07 - 0.13	0.005	0.1
Isabel	$500 \times 500 \times 100$	11	23057	2047	11×120	1	1.6	3.5
Rayleigh-Bénard	$127 \times 127 \times 127$	7	1054	56	7×7	0.1 - 0.2	0.01	7.7
Square Cylinder	$115 \times 64 \times 48$	7	1207	670	7×0.5	7×0.03	0.03	2

Table 3.1: Running times and number of SIFT features for the data sets used in this paper.



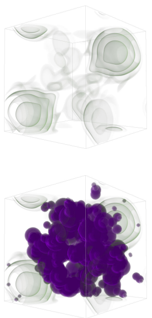
(a) Selection. (b) Matching patterns.

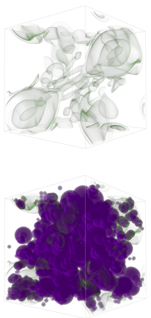
Figure 3.22: Selection of an area around a carbon atom in the electrostatic potential of the Benzene molecule. This pattern can be found six times in this scalar field, namely around all six carbon atoms.

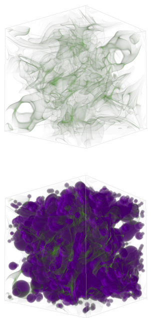
result of the pattern matching highlights all six carbon atoms. This real-world example shows once more that our 3D SIFT descriptors are rotation invariant.

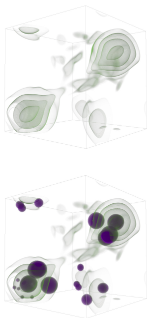
Borromean Magnetic Flux Vector Field Figures 3.23-3.25 show the Borromean magnetic flux – a vector field from an experiment studying magnetic energy decay [14]. In its initial state, it features interlocked magnetic rings. The shown state exhibits already a large amount of decay. We include this vector field to show how our multi-field pattern matching can help in understanding vector fields despite working only with derived scalar fields.

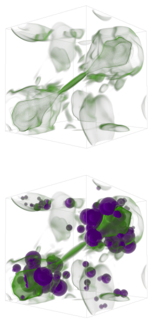
Figure 3.23 shows the six trait fields we computed from the original vector field: magnitude, norm of the Jacobian, stream line curvature, helicity, λ_2, and Okubo-Weiss. Besides a volume rendering of each trait field, Figure 3.23 also shows their SIFT features as spheres. The size of the spheres denotes the scale of the SIFT feature, i.e., the size of the supporting neighborhood. Some of the trait fields are densely covered with SIFT features, while others exhibit them only in distinguished regions. This is not much of an issue, since (i) all regions have coverage by at least one of the trait fields, and (ii) SIFT features are only used to generate candidate patterns and the subsequent cost


(a) Flow magnitude and its SIFT features.


(b) Norm of the Jacobian and its SIFT features.


(c) Curvature and its SIFT features.


(d) Helicity and its SIFT features.


(e) λ_2 and its SIFT features.

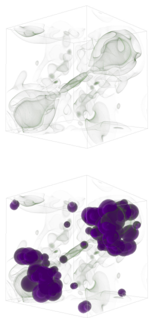

(f) Okubo Weiss and its SIFT features.

Figure 3.23: Trait fields of the Borromean magnetic flux vector field. Their SIFT features are shown as spheres.

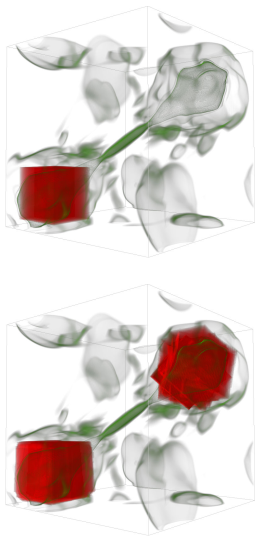
computation involves again all traits (see Section 3.3.3).

We made two pattern selections in this data set. In Figure 3.24 we selected one of the outer rings. The pattern matching yields the other outer ring on the opposite side of the volume. The stream line rendering highlights these structures. Note how our multi-field approach is able to address structures that are inherent to the vector field.

For Figure 3.25 we selected a region in the middle of the domain. The matching result shows a ring-like structure. The stream line rendering reveals that these are the remains of the interlocked rings from the beginning of the magnetic energy decay experiment.

Climate Multi-Field Data Set Figure 3.26 shows a multi-field climate data set. This is a time step of a large re-analysis of the world's climate in the years 1979–2013. The data set is courtesy of Dim Coumou and Thomas Nocke from the Potsdam Institute for Climate Impact Research (PIK). This 3D multi-field data set spans the entire planet and several kilometers of the atmosphere. Figures 3.26c–e show volume renderings of the considered trait fields iso-pressure height, temperature, and wind speed. We selected a region at the North-West coast of the USA. Interestingly, the iso-pressure height did not produce any matching patterns, since this particular location does not contain SIFT features. This is not much of a surprise, since this data set shows only structures near the ground, but the isosurfaces are almost planar in higher regions. Anyway, we got plenty of matches in the other two trait fields. In Figure 3.26b we show the final pattern matching result for all traits combined.

Hurricane Isabel Multi-Field Data Set In Figure 3.27 we applied our method to the Hurricane Isabel data set from the IEEE Visualization 2004 contest. This is a complex 3D time-dependent data set produced by the Weather Research and Forecast (WRF)


(a) Selection (top) and matching patterns.

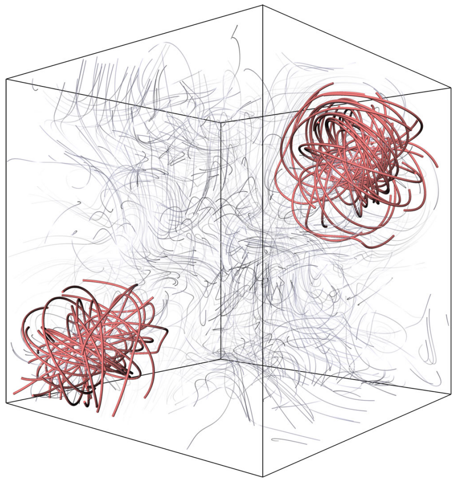
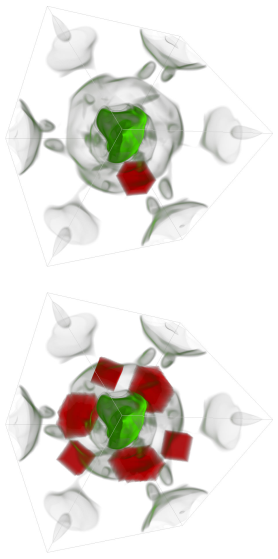

(b) Stream lines shown for the entire vector field and highlighted within the matching patterns.

Figure 3.24: Borromean data set with matching outer rings.


(a) Selection (top) and matching patterns.

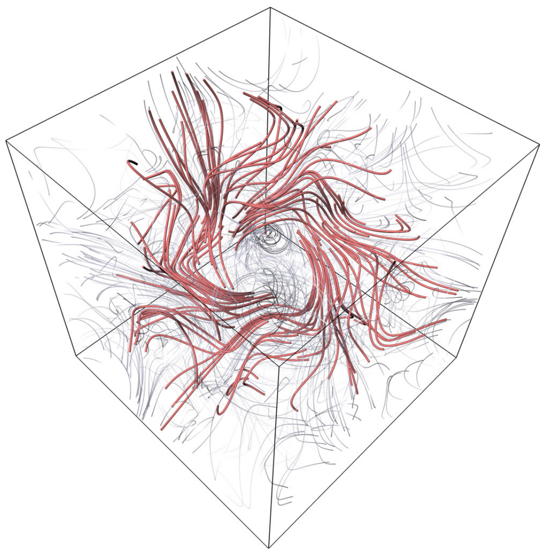
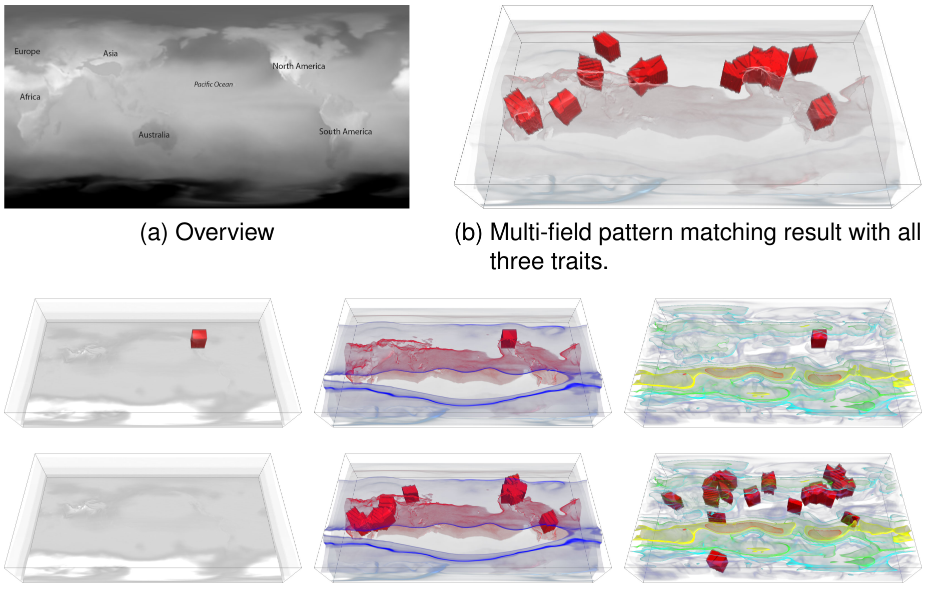

(b) Stream lines shown for the entire vector field and highlighted within the matching patterns.

Figure 3.25: Borromean data set with the inner ring revealed by our method.



(a) Overview (b) Multi-field pattern matching result with all three traits.

(c) Iso-Pressure Height: selection (top) and matching patterns (none). (d) Temperature: selection (top) and matching patterns. (e) Wind speed: selection (top) and matching patterns.

Figure 3.26: Climate multi-field data set with three traits.

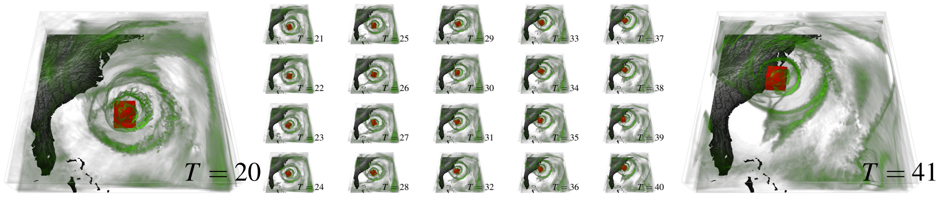
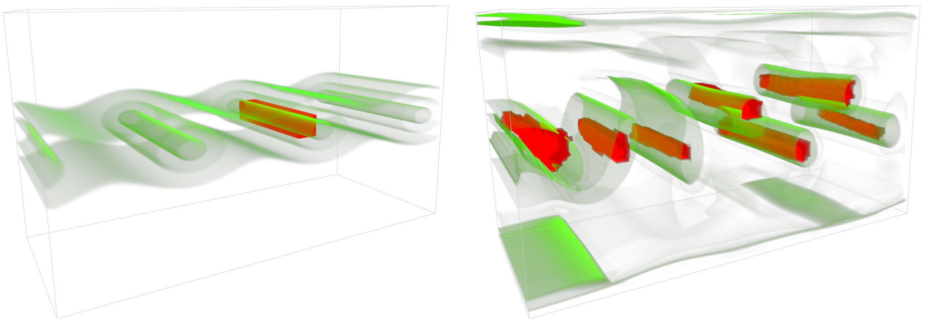


Figure 3.27: The *Hurricane Isabel* data set. The user selects the eye of the hurricane at $T = 20$ using a red box. Our algorithm uses the 3D SIFT features of 11 scalar fields simultaneously to find matching patterns in the following time steps. This amounts to a tracking of the eye of the hurricane.



(a) Analytic vortex description (Stuart Vortex) used to define a reference pattern.

(b) Matching patterns in the numerical simulation of a flow behind a square cylinder.

Figure 3.28: A pattern has been analytically “designed” and then applied to a real-world flow in order to find all vortex structures in the von Kármán vortex street. Both images show a volume rendering of the *vorticity magnitude* trait field of the respective flow.

model courtesy of NCAR and the U.S. National Science Foundation (NSF). It contains 10 scalar fields and 1 vector field. For our purposes, we considered all 10 scalar fields as well as the magnitude of the flow (wind speed).

We made a more advanced experiment with this data set: we select the eye of the hurricane in the time step $T = 20$ and make this our reference pattern. However, we apply the pattern matching to the following time steps $T \in [21, \ldots, 41]$. All 11 trait fields are considered for this. As Figure 3.27 as well as the accompanying video show, this leads to a stable tracking of the eye of the hurricane.

Square Cylinder Flow In Figure 3.28 we take this approach even one step further. Instead of applying a pattern from one time step to another, we apply a pattern from a different data set to the square cylinder flow [13, 84].

The interesting part in this flow is the von Kármán vortex street. It is characterized by alternating vortices created by periodic vortex shedding directly behind the cylinder. In our experiment, we attempt to capture these vortices using pattern matching. However, unlike our other experiments, the pattern is not a selection from the same data set, but an analytic vortex description often referred to as the *Stuart Vortex*:

$$\mathbf{v} = \left(\sinh(2y)\ ,\ \frac{1}{4}\sin(2x)\ ,\ z\,(\cosh(2y) - \frac{1}{4}\cos(2x))\right)^T .$$

We sampled this field and computed the same trait fields that we also have for the square cylinder flow, namely: magnitude, stream line curvature, helicity, divergence, vorticity magnitude, λ_2, and Okubo-Weiss.

Figure 3.28a shows the selection in the Stuart Vortex. This selected vortex pattern has been applied to the unrelated square cylinder flow. Figure 3.28b shows the matching result, which nicely covers the vortices in the von Kármán vortex street. This example shows that our algorithm can also be applied in scenarios, where a reference pattern is "designed" beforehand as a way to describe features of interest.

3.6 Summary

We introduced a novel pattern matching approach for multi-field data sets. It bundles the information from different fields into the description of a pattern. The method is very efficient, since we work with a sparse set of features to drastically reduce the search space for the pattern matching. We discussed how to achieve full rotation invariance for the SIFT features. For future work, tensor fields should be taken into consideration.

Chapter 4

Hierarchical Hashing for Pattern Search in 3D Vector Fields

4.1 Overview

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 chapter, 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

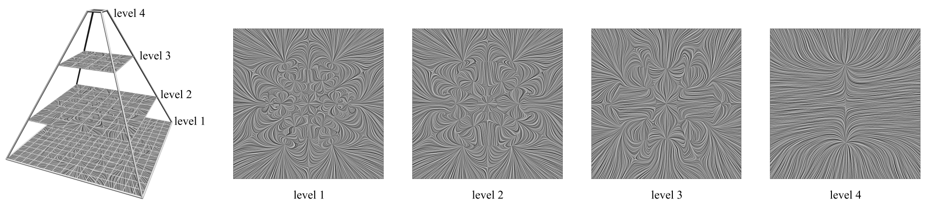


Figure 4.1: 2D illustration of the scale space of a vector field $\mathbf{v}$. The scale space consists of a number of derived vector fields, where every level is a smoothed version of the previous level at half the resolution. Our algorithm works with 3D vector fields.

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.

In the following, In section 4.2 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 4.3 uses this for the actual pattern search in 3D flows, where the patterns are defined as arrangements of spheres. In section 4.4 we evaluate the accuracy and robustness of our method. Section 4.5 showcases results using different flow data sets. We summarize with a discussion in section 4.6.

4.2 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.

4.2.1 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 the basis for our pattern search algorithm. It is an ordered set $\mathcal{V} = [\mathbf{v}_0, \dots, \mathbf{v}_n]$ of vector fields derived from $\mathbf{v}$ with decreasing complexity and resolution. It allows us to describe the features of the flow at different scales and is the key to making the entire algorithm scale-independent.

The original vector field $\mathbf{v}$ is the lowest level of the scale space, i.e., $\mathbf{v}_0 = \mathbf{v}$. Each subsequent level $\mathbf{v}_{\ell+1}$ is generated by filtering $\mathbf{v}_\ell$ using a Gaussian kernel and halfing the resolution. Figure 4.1 gives an illustration.

4.2.2 Base Descriptors

A base descriptor encodes the local flow behavior at a certain level of the scale space. We will sample the domain with base descriptors. Later, we show how to combine them to find flow patterns.

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 $\mathbf{b}(\mathbf{p}_\ell) = \mathbf{t}(\mathbf{p}_\ell) \times \mathbf{n}(\mathbf{p}_\ell)$ (cf. [23]). 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 $\mathbf{p}_\ell$

$$(\mathbf{p}_\ell \pm u_\ell \mathbf{t}) \qquad (\mathbf{p}_\ell \pm u_\ell \mathbf{n}) \qquad (\mathbf{p}_\ell \pm u_\ell \mathbf{b}), \tag{4.1}$$

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

Rotation invariance is achieved by relating the local coordinate system to the global one. We use Singular Value Decomposition (SVD) to compute a rotation matrix $\mathcal{R}(\mathbf{p}_\ell)$ between the local coordinate system and the world coordinate system (cf. Arun et al. [3]):

$$(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathcal{R}(\mathbf{p}_\ell) \cdot (\mathbf{t}(\mathbf{p}_\ell), \mathbf{n}(\mathbf{p}_\ell), \mathbf{b}(\mathbf{p}_\ell)) \,. \tag{4.2}$$

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

$$\mathcal{B}(\mathbf{p}_\ell) = (\mathcal{R}\tilde{\mathbf{v}}_1, \ldots, \mathcal{R}\tilde{\mathbf{v}}_6). \tag{4.3}$$

4.2.3 Base Descriptor Hashing

Hashing is used to speed up the search for similar descriptors. The general idea is to quantize similar descriptors into bins, akin to a histogram. 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 4.2 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.

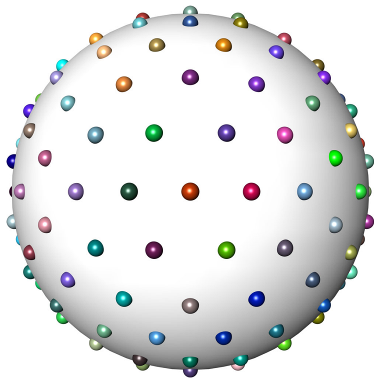


Figure 4.2: Equidistant point sampling on the unit sphere. The Voronoi cells of the shown points are the bins for hashing orientations.

Classic hashing fails to discriminate similar items at bin boundaries. We employ two strategies to deal with this: *Locality Sensitive Hashing* (LSH) [40] and its extension *multi-probe Locality Sensitive Hashing* [50].

LSH [40] 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 [50] 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.

4.2.4 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 (this follows from the multi-probe LSH algorithm). 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.

To simplify the explanation of the following parts, we introduce a binary cost for both querying and comparison of base descriptors. Two base descriptors $\mathcal{B}_i$ and $\mathcal{B}_j$ are

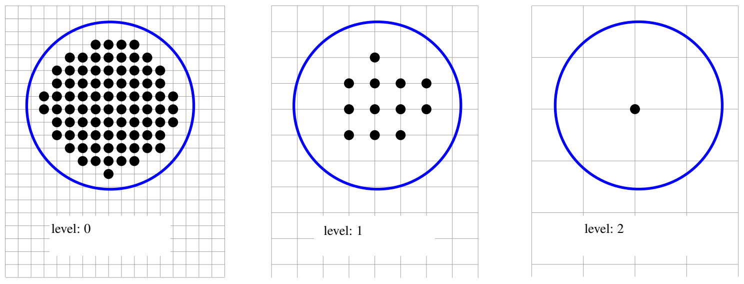


Figure 4.3: 2D illustration of a sphere descriptor (blue circle) consisting of a number of base descriptors (black dots) at different levels of the scale space.

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

$$\mathcal{E}_B(\mathcal{B}_i, \mathcal{B}_j) = \begin{cases} 0 & \mathcal{B}_i = \mathcal{B}_j \\ 1 & \mathcal{B}_i \neq \mathcal{B}_j \end{cases} . \tag{4.4}$$

4.3 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.

4.3.1 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

$$\mathcal{D}(\mathbf{o}, r) = \{\mathcal{B}(\mathbf{p}_\ell) : \ ||\mathbf{p} - \mathbf{o}|| < r, \ \mathbf{p} \in \mathcal{N}_d(\mathbf{p}_\ell)\}\,, \tag{4.5}$$

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 4.3. 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

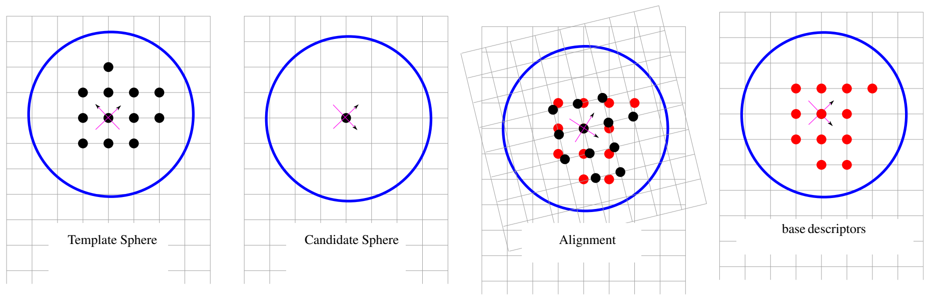


Figure 4.4: 2D illustration of finding a candidate sphere and its corresponding base descriptors at a specific scale level. both first base descriptors $\mathcal{B}_0$ marked with a coordinate system are aligned. The red dots represent the closest integer grid point for the rotated base descriptors of the template sphere.

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. (4.2)). 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 4.4 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:

$$w(0) = 1 \tag{4.6}$$
$$w(\ell+1) = 8\,w(\ell)\,. \tag{4.7}$$

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

$$\mathcal{E}_D(\mathcal{D},\mathcal{D}') = \sum_i w_i \mathcal{E}_B(\mathcal{B}_i,\mathcal{B}'_i)\,. \tag{4.8}$$

Since the cost $\mathcal{E}_B$ of two base descriptors is a binary value (see (4.4)), 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

$$\mathcal{S}_D(\mathcal{D},\mathcal{D}') = \frac{\sum_i w_i - \mathcal{E}_D(\mathcal{D},\mathcal{D}')}{\sum_i w_i}\,, \tag{4.9}$$

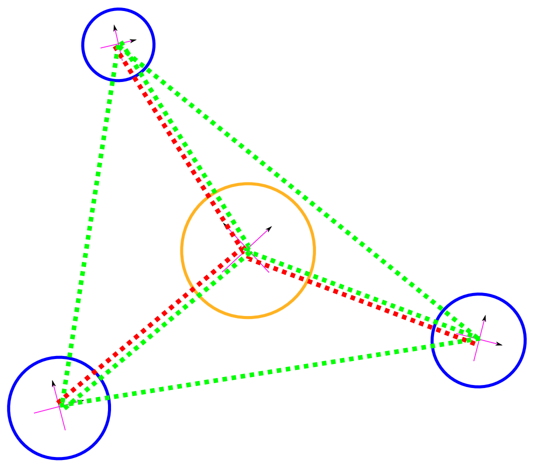


Figure 4.5: Layout verification illustrated in 2D. Four circles define a template pattern in 2D. The yellow circle in the middle is the central circle as it is closest to the center of the pattern. Red dotted lines indicate the pairwise verification of scaling and rotation. The green dotted lines indicate the pairwise verification of translation.

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.

4.3.2 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 $\mathcal{B}_0^i$. 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 $\mathcal{B}_0^i$. Furthermore, we define the center of a template pattern $\mathcal{P}$ by weighting the origins of all its spheres

$$\mathcal{O} = \frac{\sum_{i=1}^{\mathbf{m}} r^i \mathbf{o}^i}{\sum_{i=1}^{\mathbf{m}} r^i} \quad . \tag{4.10}$$

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 4.5):

- **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

  $$\Lambda^i - \Lambda^c = \Lambda^{i'} - \Lambda^{c'} \quad . \tag{4.11}$$

- **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

  $$\measuredangle(\mathcal{R}_c \Gamma^i_j, \mathcal{R}_c{}' \Gamma^{i'}_j) < \theta, j \in \{1,2,3\} \; , \tag{4.12}$$

  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 4.2.3.

- **Translation** A distance ratio threshold λ is introduced to constrain the distance deviation for any pair of spheres as

  $$\forall i, j \; \left| \frac{||\mathbf{o}^i - \mathbf{o}^j|| - \left|\left|\mathbf{o}^{i'} - \mathbf{o}^{j'}\right|\right|}{||\mathbf{o}^i - \mathbf{o}^j||} \right| < \lambda \; , \tag{4.13}$$

  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

$$\mathcal{S}_P(\mathcal{P}, \mathcal{P}') = \frac{1}{m} \sum_{i=1}^{m} \mathcal{S}_D(\mathcal{D}^i, \mathcal{D}^{i'}) \quad . \tag{4.14}$$

4.4 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.

4.4.1 Parameter Overview

Our pattern matching approach contains two parameters that can be adjusted by the user: the distance ratio threshold λ from (4.13), and a threshold on the pattern similarity

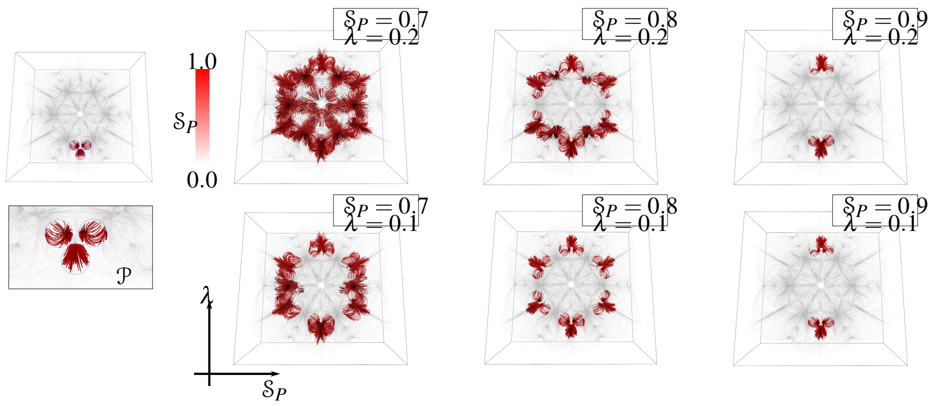

$\mathcal{S}_P = 0.7$ $\lambda = 0.2$
$\mathcal{S}_P = 0.8$ $\lambda = 0.2$
$\mathcal{S}_P = 0.9$ $\lambda = 0.2$
1.0
$\mathcal{S}_P$
0.0
$\mathcal{S}_P = 0.7$ $\lambda = 0.1$
$\mathcal{S}_P = 0.8$ $\lambda = 0.1$
$\mathcal{S}_P = 0.9$ $\lambda = 0.1$
$\mathcal{P}$
λ
$\mathcal{S}_P$

Figure 4.6: 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 $\lambda \in \{0.1,\ 0.2\}$.

value $\mathcal{S}_P$ from (4.14). 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 4.6. 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 4.6. 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 (subsection 4.2.3), and the number of such (randomly rotated) hashing spheres. We fixed the former to 162, and the latter to 20.

4.4.2 Evaluation of Retrieval Accuracy

We conduct an experiment to perform this evaluation. Given two unit vectors **a** and **b** with angle α, we generate 20000 random vector pairs by transforming **a** and **b** in combination with random rotation matrices. In any LSH algorithm, If **a** and **b** are considered similar, idealy all the transformed vector pairs with angle α should be considered similar. In this experiment, we compute the dissimilarity (cost) of two vectors with angle α as the percentage of all 20000 pairs which are considered as

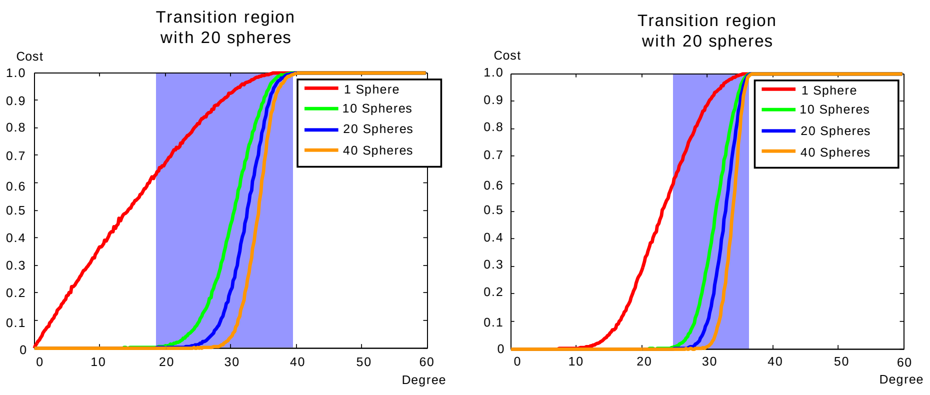


Figure 4.7: Accuracy curves of different LSH strategies and different number of hashing functions (Spheres). Left: classic LSH algorithm. Right: Multi-probe LSH algorithm.

different. We repeat this computation for the angles from 0.0 to 60.0. The resulting curve describes the accuracy of the retrieval algorithm. In perfection, the accuracy curve should be a step function, i.e., starting from 0, jumping to 1 at middle and keeping the value to the end, while in practice, a high accuracy means the transition region where the curve jumps from 0 to 1 should be narrow. In Figure 4.7, we plot the curves under different number of hashing functions (Spheres). In both algorithms, we observe that the more hashing functions used, the better accuracy obtained. To ballance the accuracy and efficiency, we choose 20 hashing functions in our algorithm.

4.4.3 Comparison of LSH strategies

We compare the classic LSH algorithm and multi-probe LSH algorithm also using the accuracy curves as mentioned above. In Figure 4.7, we observe that when both algorithms use 20 hashing functions, the transition region of multi-probe LSH is much smaller than that of classic LSH algorithm. Based on this observation, we use multi-probe LSH algorithm in this paper.

4.4.4 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 4.12 shows this at the von Kármán vortex street in the flow behind a square cylinder.

4.4.5 Robust to Translation, Rotation, and Scaling

In Figure 4.8, we evaluate our approach using the pattern defined in an analytical flow field. In this 3D flow field, each vector (u, v, w) is given by

$$\begin{aligned} u(x,y,z) &= y \\ v(x,y,z) &= (x-0.5)\cdot(x+0.5) \\ w(x,y,z) &= (z-0.5)\cdot(z+0.5+2y), \end{aligned} \tag{4.15}$$

where (x, y, z) indicates the location. We select four spheres as the reference pattern which is shown in Figure 4.8; We evaluate our approach under different translations, rotations, scaling factors, white noises, and deformations. In the evaluation against translation and rotation, we conduct two test cases. First, as solid lines marked in Figure 4.8, we translate or rotate the pattern along a number of randomly selected axes, then evaluate their average similarity. Second, as dashed lines marked in Figure 4.8, we translate or rotate the pattern strictly along single axis, i.e., x-axis. We observe that, in any case, the similarities increase with the augment of the volume resolution. The reason is that when the volume resolution increases, the error caused by pattern alignment is thereby reduced. In the second test case of translation and rotation, as well as the scaling validation, their similarities reach maximum when the grid point arrangement is perfect, i.e., in 90 degree and 180 degree of rotation, in integer voxel size of translation, and in 2^n where $n \geq 1$ of scaling. When the scaling factor is smaller than 1.0, details of the pattern become fewer and fewer. It causes the drop of the similarity curve. In the noise validation, we pollute our pattern by adding different level of white noises to each component in the data set. The level of noise is based on the range of component-wise magnitude inside the reference pattern. In resolution of 65, our algorithm achieves similarity of 0.7 with 20% noise. It also increases when a higher resolution volume is given. In the deformation validation, we scale the pattern in one of three dimensions, and record the average similarity. In resolution of 65, our approach obtains similarity of 0.7 with the deformation ranged from 0.7 to 1.8. Again, it increases when a higher resolution is given.

4.4.6 Timings

Table 4.1 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 4.4.1, a small threshold on the pattern similarity 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.

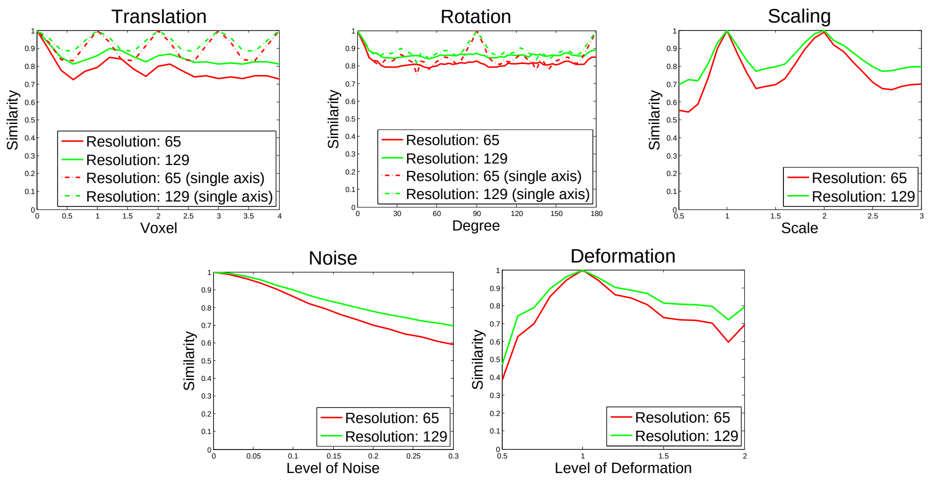


Figure 4.8: Robustness validation. All the validations is conducted in the volume with resolution of 65 and 129. We conduct the validation in 5 aspects, i.e., validation for translation, rotation, scaling, noise, and deformation. The dashed curves in translation and rotation figures indicates the transformation is performed along single axis, i.e. x-axis.

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 4.1: 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.

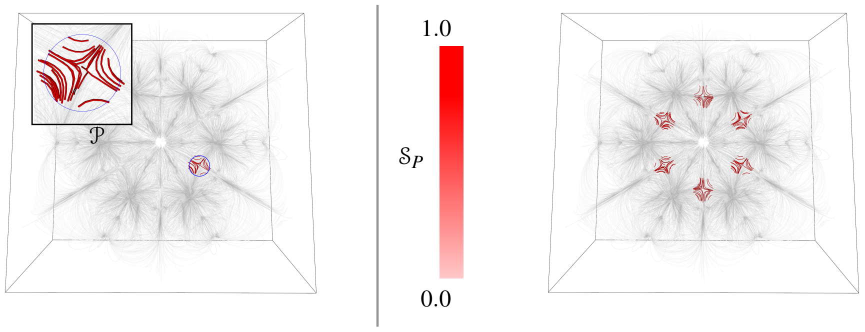


Figure 4.9: Detection of the rotational symmetry in the BENZENE data set using a saddle-like template pattern.

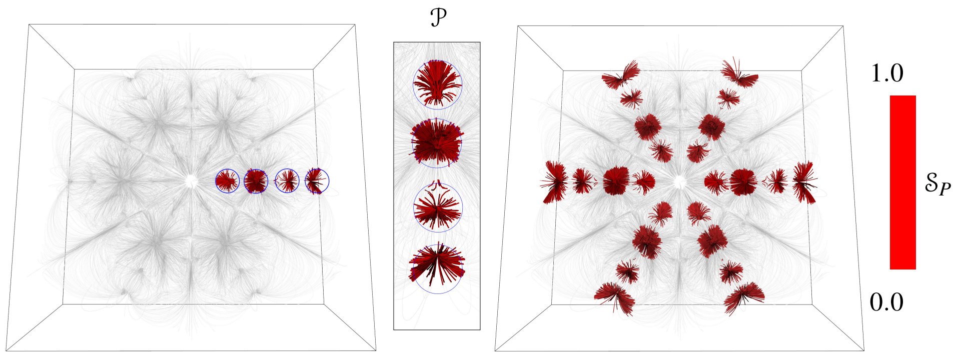


Figure 4.10: Pattern search in the BENZENE data set. The template pattern $\mathcal{P}$ links four spheres. Each of them consists a singularity inside.

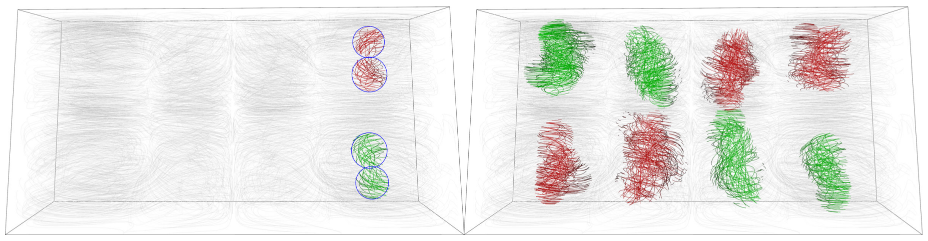


Figure 4.11: Pattern search in the RAYLEIGH-BÉNARD flow. The template pattern $\mathcal{P}$ uses two spheres to describe a narrowing spiral.

4.5 Results

Figures 4.9 and 4.10 show further results from the BENZENE data set. Note how a pattern consisting of a single sphere in Figure 4.9 is able to capture the 6-fold rotational symmetry of this data set. The pattern in Figure 4.10 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 4.11, 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 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 4.4.4.

In Figure 4.12, 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 4.13 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 use two tightly placed spheres in one of them to describe a short segment of the vortex. 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.

4.6 Summary

In this chapter, 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 chapter 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

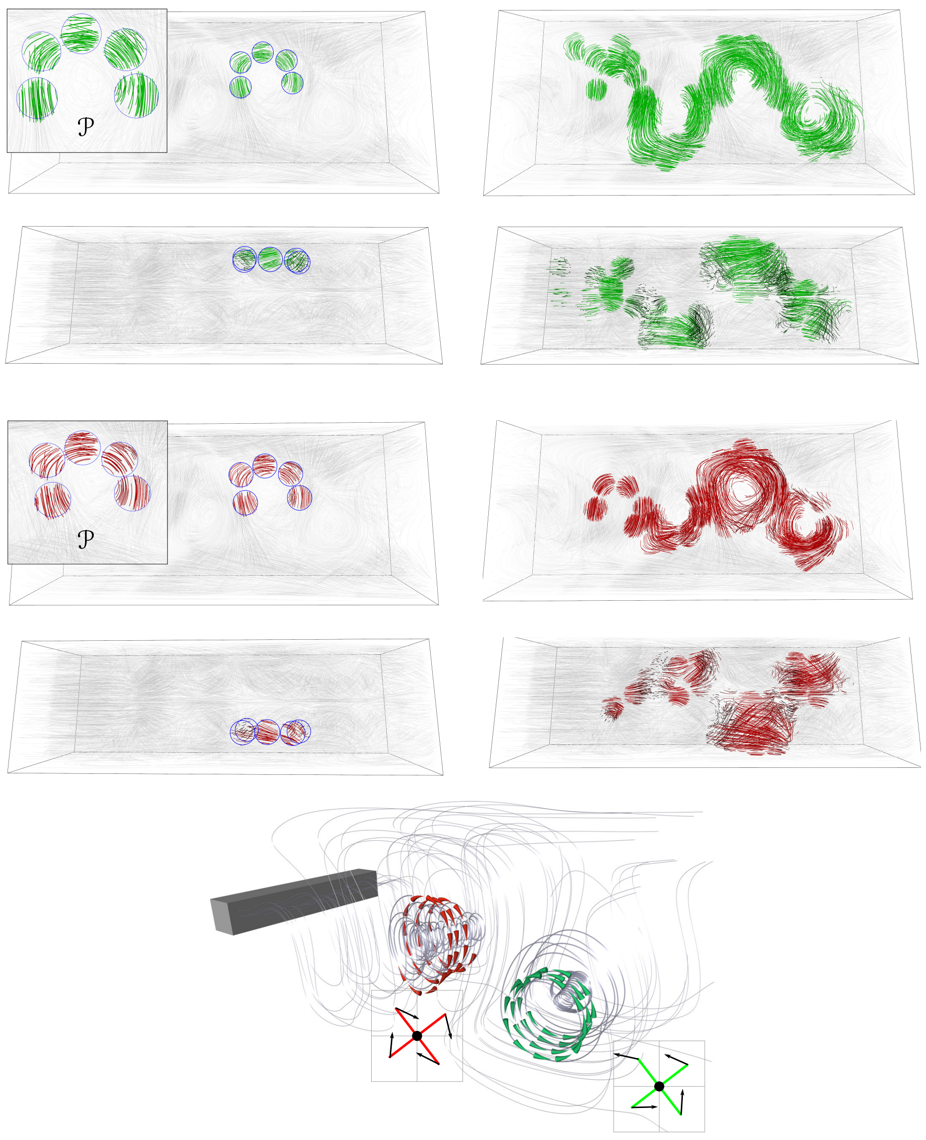


Figure 4.12: Pattern search in the CYLINDER flow. The template pattern $\mathcal{P}$ consists of five spheres describing an extended arc. Since base descriptors are sensitive to the orientation of swirling flow. This enables us to distinguish between clockwise and counterclockwise rotating vortices.

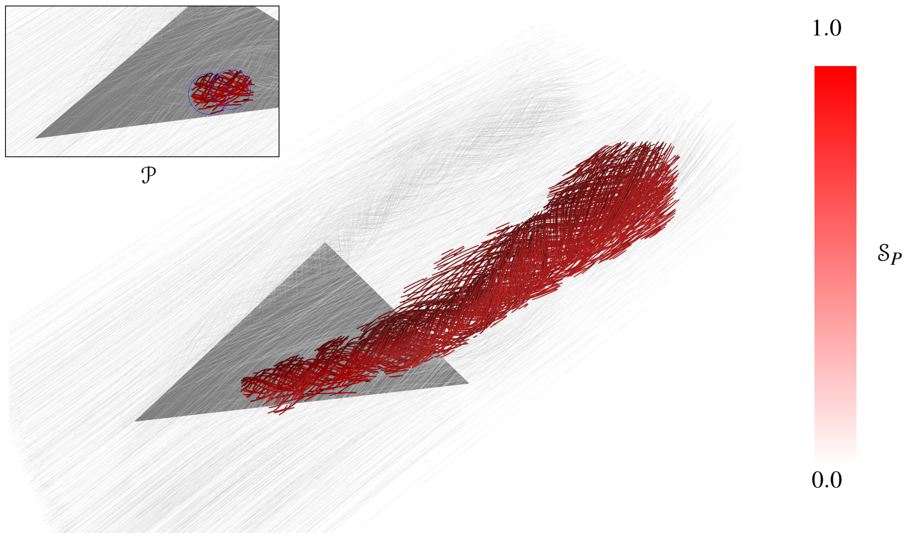


Figure 4.13: Pattern search in the DELTAWING flow. We select two small nearby spheres to describe a small segment of a vortex core. The result shows that we detect occurrences at different scales.

good results most of the time, it still has some limitations. For example, it works best on vector fields with cubic voxels, i.e., the length of a voxel is the same in all three dimensions. This is because of the rotational alignment when comparing sphere descriptors, as illustrated in Figure 4.4. 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.

Chapter 5

Pattern Search in Flows based on Similarity of Stream Line Segments

5.1 Overview

The visualization and analysis of vector fields is of major importance for various scientific disciplines. Among the different classes of vector field visualization techniques, geometry-based techniques are well-established [55]. They rely on integral curves such as stream lines representing integrated flow behavior.

However, line-based flow visualizations face certain challenges: for instance, if applied to very complex data sets, they can quickly lead to cluttered visualizations. In particular, this is a problem for 3D flows. Figure 5.11 shows such an example, where the gray stream lines of the 3D vector field occlude each other to an extent that renders the entire visualization almost illegible. The possibly existing structures within this field are lost due to the visual clutter.

Furthermore, a stream line is a domain-wide integrated entity, but very often not all of its parts are equally important: for some applications, the part of a stream line in the vicinity of a vortex or critical point is more important than the part running through a laminar region of the flow. However, ultimately the definition of what constitutes a "flow feature" depends on the specific application and the visualization target of the domain expert.

The method presented in this paper empowers the user to define complex flow features. We propose an example-based pattern retrieval approach: users are able to specify interesting flow features as *patterns* that are constructed of stream line *segments*, i.e., parts of stream lines. In contrast to previous work, our method supports patterns represented by *multiple* line segments, which increases the flexibility and expressiveness of the specified patterns. This way, it is possible to specify even complex flow patterns such as the one shown in Figure 5.11. Patterns are matched with the vector field and successful matches of the example pattern are presented to the user. We formulate the matching to be invariant to similarity transformations, such that matched patterns are found independently of their location, orientation, or scale. In addition, pattern occurrences can be found in the same data set, in a different time step of the same data set, or in a different data set.

At its core, our formulation of flow pattern retrieval requires suitable steam line segmentations as well as measures for segment similarity, for which we propose possible solutions. Suitable stream line segmentations should allow the convenient selection of flow feature regions. We relate flow features to flow curvature, such that segment boundaries are naturally found at minima of curvatures. In addition, segmentations need to be consistent along multiple scales to allow scale-invariant pattern retrieval. We propose a new stream line segmentation scheme that is based only on intrinsic curve properties and fulfills all of these requirements. In addition, our stream line segmentation scheme and similarity estimates are valuable on their own rights and can also be use, e.g., for flow clustering applications.

In the following two sections, we present details of our two main contributions: the *consistent intrinsic stream line segmentation* and *segment similarity estimation* (Section 5.2) and the *segmentation-based flow feature pattern retrieval* (Section 5.3). Section 5.5 examines the efficiency of the proposed method by conducting several experiments and we discuss properties of our approach in section 5.6.

5.2 Intrinsic Stream Line Segmentation

Our approach is motivated by the observation that long stream lines often pass through several mutually distinct flow features of the underlying vector field, e.g., through multiple vortical regions. An independent analysis and retrieval of these distinct features is not possible by considering whole stream lines. Instead, our method considers spatially restricted parts of stream lines in form of shorter *stream line segments*, which can be combined to form sets of segments that represent *flow feature patterns*. Searching for these pattern requires stream line segmentations that are consistent both among all possible stream lines and among similar flow features. A segmentation has to be designed in such a way that it facilitates the estimation of individual segment similarities. Given a consistent stream line segmentation, our method for pattern retrieval evaluates pairwise similarities of segments. Both operations are formulated to be invariant to similarity transformations by exploiting intrinsic curve properties. This enables our method to find patterns at different locations and at different scales.

Notation. We make use of the following formal concepts: let $\mathbf{v}(\mathbf{x})$ denote steady differentiable vector fields with associated over two ($d = 2$) and three-dimensional ($d = 3$) flow domains $\mathcal{D} \subset \mathbb{R}^d$ with $\mathbf{x} \in \mathcal{D}$. Parametric *stream lines* $\mathbf{c}(t) = \mathbf{x}_0 + \int_0^t \mathbf{v}(\mathbf{c}(u))\,du$ are curves defined through integration along $\mathbf{v}$ starting from a seed point $\mathbf{x}_0$ for an integration time t. We partition stream lines $\mathbf{c}$ into disjoint *stream line segments* $\mathbf{s}_i(t)$ by splitting $\mathbf{c}$ at integration times t_i such that the points of $\mathbf{c}$ and $\mathbf{s}_i$ coincide for $t \in [t_i, t_{i+1}]$. For a stream line integrated from t_0 to t_n, a segmentation is defined by the sequence $[t_0, \ldots t_i, \ldots t_n]$ of segment boundaries t_i. We denote the length of $\mathbf{s}_i$ by l_i.

5.2.1 Globally Consistent Segmentation of Stream Lines

We identify three requirements a globally consistent segmentation has to satisfy for our application: first, a segmentation should be *feature preserving* in that all segments shall preserve the important features of the given set of curves. In general, long and sharp arcs of stream lines are considered to be important and significant curve features, while short and straight curves are less important: for instance, Günther et.al. [33] define local stream line importance by local curvature. Figure 5.1 (a) exemplifies this property: segments of the upper curve do not preserve the important curly shapes of high curvature in the original curve, whereas the bottom segmentation maintains this feature in the collection of segments.

Comparison of individual segments is also facilitated by this requirement, as segment boundaries will always represent boundaries of feature regions and give suitable reference points for similarity computations. This is related to the second requirement: a segment should be *distinct* enough to describe a complete feature. For instance, a circle should not be separated into two semi-circles. Figure 5.1 (b) shows an example in which the upper spiral curve is over-segmented into several semi-circles, which do not represent the more dominant spiral feature anymore. In contrast, this distinct feature is well-represented in the segmentation underneath. The third property requires a segmentation to be *consistent* in that segments, which describe similar flow features, should have similar shapes. This implies that two congruent curves at different scales shall always be segmented in a compatible way. In other words, the segmentation should be invariant to translation, rotation, scaling, and reflection, i.e., invariant to similarity transformations. This property is illustrated in Figure 5.1 (c), in which the bottom curve is a translated, rotated, and scaled version of the upper one. Since both curves have the same segmentation structures, their segmentations are consistent. Our segmentation only relies on stream line curvatures $\kappa_{2/3}(t)$. Although 2D and 3D curve curvatures are defined differently, i.e., κ_2 are signed, while κ_3 are always positive, our segmentation scheme supports 2D and 3D curves in a unified way. Stream line segmentation proceeds in two phases, curvature-based splitting and subsequent segment merging, and we continue to describe both in more detail.

Segment Splitting. Both curvature estimations κ_2 and κ_3 differ in their signedness. Hence, we consider absolute local curvatures $\widehat{\kappa}(t) = |\kappa_{2/3}|$ for a unified stream line segmentation scheme that is applicable for both two and three dimensional stream lines. Vector field features are usually coupled to high absolute stream line curvatures (see, e.g., [54]). Therefore, to obtain feature-preserving and distinct segmentations, points of absolute local curvature minima that bound these high curvature regions are candidates for possible segment boundaries. Along a stream line, let t_i denote the integration times corresponding to absolute local curvature minima, i.e., $\widehat{\kappa}(t) > \widehat{\kappa}(t_i)$ for $t \in [t_i - \varepsilon, t_i + \varepsilon]$. We call these segments bounded by consecutive absolute local curvature minima *minimal segments*. Minimal segments are the initial building blocks of the final segmentation and will not be split further to preserve the features of higher curvature they represent. Still, as the total curvature of the features that minimal segments represent can vary considerably, we merge minimal segments into segments of higher significance.

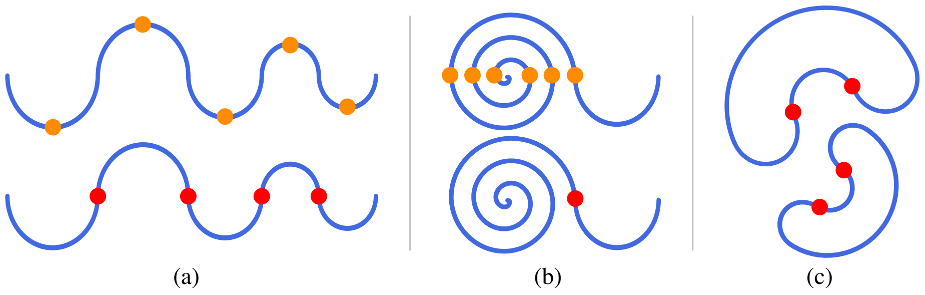


Figure 5.1: Curve Segmentation. Our segmentation scheme splits stream lines in a globally consistent way at (●) (alternative, less suited split locations are colored as (●)). Shown examples illustrate different properties of our segmentation, i.e., *feature preservation* (a), *feature distinction* (b), and segmentation *consistency* w.r.t. location, orientation, and scale (c).

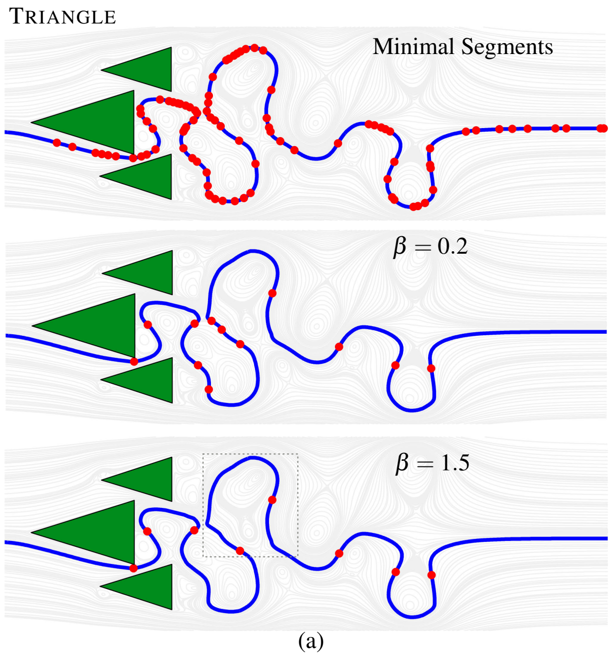
Segment Merging. We merge neighboring segments based on two segment properties: total segment curvature and average segment orientation. Both properties are scale-invariant. The *total segment curvature* $\widehat{\kappa}_i$ is given by

$$\widehat{\kappa}_i = \int_{t_i}^{t_{i+1}} \widehat{\kappa}\, ||\dot{\mathbf{c}}||\, dt \; . \tag{5.1}$$

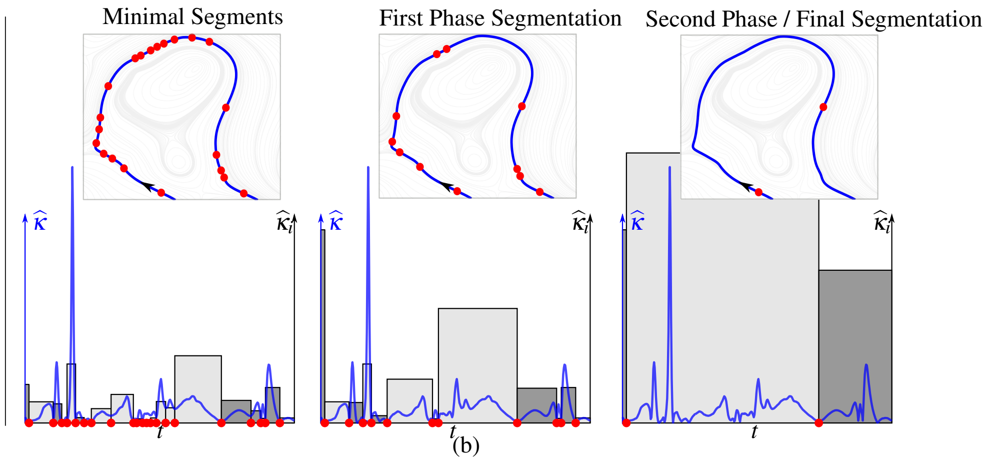
Along each segment, the orthonormal Frenet-Serret frames $(\mathbf{t}(t), \mathbf{n}(t), \mathbf{b}(t))$ are given by the tangent, normal, and bi-normal directions, respectively. We observe that along a minimal segment the variation of bi-normal directions is usually small. Therefore, we assign each segment an *average orientation* $\bar{\mathbf{b}}_i$ based on its average bi-normal direction.

Our algorithm for merging of segments consists of growing segments of low total curvature with neighboring segments, if they are merge-compatible. Compatibility is tested in two phases based on two criteria: first, two neighboring segments are mergeable if they have similar average orientations, i.e., if the angle $\alpha_i = \measuredangle(\bar{\mathbf{b}}_i, \bar{\mathbf{b}}_{i+1})$ is smaller than a user-specified upper bound α. Second, if one segment has a low total curvature, i.e., $\widehat{\kappa}_i < \beta$ for a user-specified upper bound β, it is mergeable with both of its neighboring segments if these two segments have similar average orientations w.r.t. α. Figure 5.3 illustrates two examples of the criteria. The merging algorithm iteratively processes segments based on a priority queue that is ordered by the total segment curvature such that segments of lowest curvature are processed first.

We illustrate the different steps of our segmentation scheme in Figure 5.2. The initial minimal segments of a single stream line are shown in Figure 5.2 (a) together with two segmentation results for different β values. This parameter steers the coarseness of the segmentation, and segmentations are usually not sensitive w.r.t. small β variations. Note that curve orientation is either positive or negative for all 2D curve segmentations. Hence, it is sufficient to select $\alpha = \pi/2$ for this case. For a closeup region, Figure 5.2 (b) shows segmentations and relation to local absolute curvatures after each segmentation phase. The graphs show that neighboring segments of similar orientation are merged into segments of higher total curvature. In the second phase, triplets of compatible segments are merged. Figure 5.4 depicts the consistency of segmentations along multiple stream



(a)



(b)

Figure 5.2: Segmentation Scheme. (a) Starting from minimal segments (top) our segmentation scheme applies two phases of iterative segment merging. Shown are two results for different β parameters (bottom, $\alpha = \pi/2$ in both cases). (b) For the three segmentations of the $\beta = 1.5$ computation, the absolute curvatures $\widehat{\kappa}$ (●) and total discrete segment curvatures $\widehat{\kappa}_i$ (box height) together with average 2D segment orientation (positive ●, negative ●) of a cutout region are shown. In the first phase, neighboring segments of close average orientation are merged. In the second phase, compatible segments of low curvature are combined if its two neighbors are compatible w.r.t. average orientation.

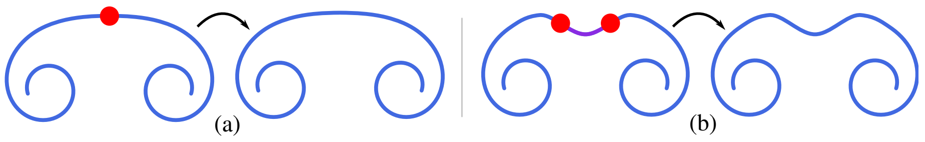


Figure 5.3: Segment Merge Criteria. Pre-merge segment boundaries are colored (●), and two different average segment orientations are colored (●) and (●). (a) A pair of segments is mergeable if they both have similar average orientations. (b) A triplet of segments is mergeable if the center segment (●) has a low average total curvature compared to its neighboring segments, which have similar average orientations.

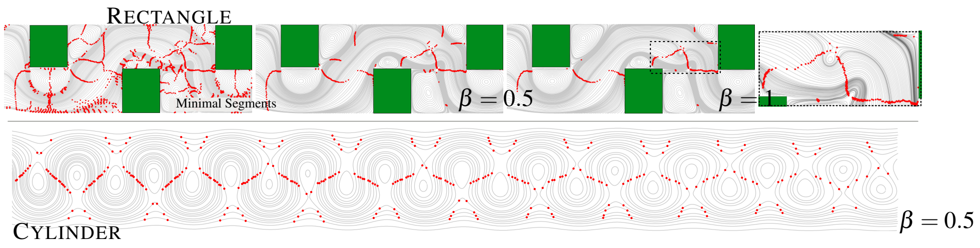


Figure 5.4: Consistent Stream Line Segmentation. Starting from the minimal segments (top left) our segmentation scheme extracts segmentations for which intrinsically similar stream line segments are segmented in a compatible way. The closeup shows that segmentations form orthogonal patterns to laminar flow regions. Removal of low curvature segments is steered by the β parameter. The slowly varying CYLINDER flow on the bottom illustrates the consistency of the segmentation.

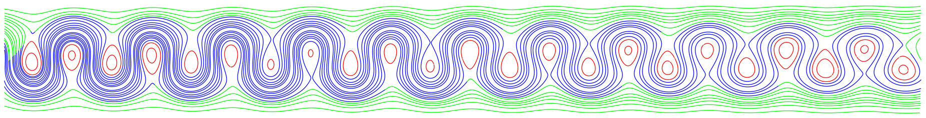


Figure 5.5: Similarity-based Clustering. Using the consistent segmentation of the CYLINDER flow shown in Figure 5.4, a clustering of segments based on pairwise intrinsic segment similarities is computed. The shown three clusters consist of approximately laminar flow segments (●), highly curves segments (●), and circular flow segments (●).

lines for two different flows. The results illustrate that our segmentation consistently computes similar segments for similar flow patterns.

5.2.2 Intrinsic Similarity of Stream Line Segments

Based on our consistent curve segmentation scheme, we propose a general scale-invariant method for intrinsic curve segment comparison.

First, we discretize the continuous intrinsic curve properties like curvatures along each segment into $n > 0$ uniformly sized bins. The parameter n steers the profile resolution and accuracy. In all our experiments, we observe that a value of $n = 40$ is usually sufficient to enable accurate segment comparison, e.g., for pattern retrieval. For comparability, we scale-normalize each profile by the curve lengths l_i.

To measure the intrinsic similarity of a pair of curve segments, we employ hEMD [59], which is a generalization of the *Earth Movers Distance* (EMD), for the comparison two scale-normalized profiles. hEMD is a cross-bin measure that is more robust w.r.t. local deformations and also a well-defined metric for our setting of unequal total profile sums.

In order to obtain similarity transformation invariance for 2D curves, we need to compute the minimum of four distance measures, i.e., two for inverting the curve traversal order, and two for flipping the sign of the curvature. For 3D segment similarity estimation, we combine differences in unsigned curvature κ_3 and torsion τ as $d_{\kappa_3} + w_\tau d_\tau$ of individual hEMD profile distances in curvature d_{κ_3} and torsion d_τ. A weight parameter $w_\tau < 1$ is chosen to reduce the influence of torsion to the final similarity estimation. Similar to the 2D case, to evaluate the similarity of two 3D segments, four hEMD evaluations are required, i.e., two for inverting the traversal order, and two for flipping the sign of the torsion.

We demonstrate the effectiveness of the intrinsic segment similarity estimation in Figure 5.5 by computing a clustering of intrinsically similar flow regions. Clustering is based on pairwise segment similarities as the metric for hierarchical clustering using Ward's minimum variance algorithm [81]. The clustering result consists of three intrinsically different clusters of segments: linear, curved, and circular segments. Note that intrinsically similar segments are grouped to clusters at different scales, which shows the scale-invariance of our similarity estimation.

Relying on the proposed consistent stream line segmentation and segment similarity estimation, we are now able to present our flow pattern search method.

5.3 Pattern Search

In this section, we propose an example-based flow pattern search approach for the detection of similar flow feature patterns given a query pattern. Patterns are defined by a collection of stream line segments, which are obtained using our consistent segmentation such that similar patterns are segmented similarly. Intrinsically similar occurrences of these pattern can be retrieved from other parts of the domain, from other points in time, or even from other data sets. Our approach starts with a dense curvature-based domain sampling by stream line integration, followed by stream line segmentation, pattern definition, and pattern retrieval. We formulate it independently of the curve dimension and the algorithm is applicable for both 2D and 3D curves.

5.3.1 Stream Lines Placement

We represent flow features by segments of stream lines in their vicinity. Hence, a sufficiently dense coverage of all important flow features by stream lines is required. Uniform stream line placement is a well-known visualization problem and we refer to the survey by McLoughlin et al. [55] for an overview on recent standard solutions. Note that standard placement approaches generally strive for uniform domain coverage by imposing both upper and lower bounds on *local* pointwise curve distances. In contrast, our setting requires curves that only respect an upper distance bound, as preemptively terminating curve integration by a lower bound on curve distances would bias the segmentation by the order of integration. Also, curve segments in converging flow regions often provide distinctive flow pattern candidates and should not be discarded. This considerably simplifies global stream line placement.

As we relate flow features to regions of high flow curvature, we employ a simple curvature-based importance sampling for stream line placement (see [85, 93]): candidate seed points are distributed uniformly in the domain using dart throwing such that they respect a prescribed upper distance bound, i.e., a candidate seed point is rejected if its nearest neighbor is closer than a prescribed value (we use the domain diagonal scaled by 10^{-5} for this upper bound). From this candidate set, we randomly draw seed points using local absolute curvature $\widehat{\kappa}$ as the importance distribution. A prescribed number of stream lines is integrated from these seed points. This scheme yields a denser distribution of stream lines in regions of higher curvature. Integration is performed in both forward and backward directions using standard fourth-order Runge Kutta integration with adaptive stepsize and equidistant curve sampling [61]. For closed stream lines, integration is stopped at the junction point.

Note that the number of total stream line segments governs the retrieval performance. To increase the efficiency of the pattern retrieval, we subsample the integrated stream lines to remove geometrically redundant, i.e., *globally* close curves, as redundant curves would yield equal and therefore redundant segments. Similar to Rössl and Theisel [62], we use two-sided Hausdorff curve distances to detect and discard extrinsically close and redundant curves while preferring longer stream lines: a set of selected stream lines is maintained and all stream lines are compared against the set members using two-sided Hausdorff distance in order of their length. If the Hausdorff distance to all set members is sufficiently high, i.e., the current curve is geometrically not redundant, it is added to the set of selected curves. This procedure is efficient as only a fraction of all possible pairwise Hausdorff distances need to be computed due to the length-based ordering. Using this filter, the cardinality of the selected set of stream lines can usually be reduced by up to 90% while retaining stream lines close to feature regions. This set of stream lines is segmented using the consistent segmentation described in Section 5.2 to give the set of segments $\mathcal{S} = \{\mathbf{s}_i\}$.

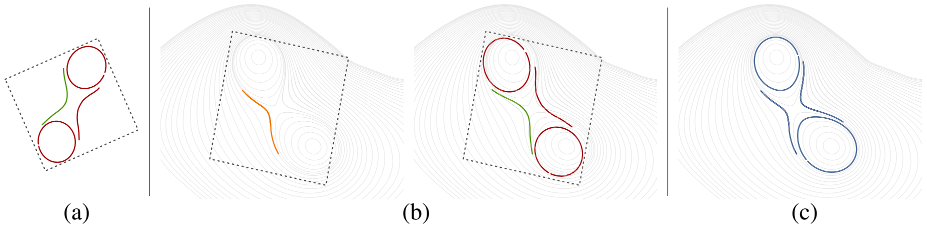


Figure 5.6: Pattern Retrieval Overview. (a) A flow feature pattern is a set of user-selected segments (●) with one distinguished root segment (●). (b) For *global alignment*, segments similar to the root segment are found using scale-invariant intrinsic similarity (● left), then all pattern segments are transformed to their vicinity by a fitted similarity transformation (right). (c) For *local alignment*, all transformed pattern segments are matched with the local data segments to detect matched patterns (●).

5.3.2 Pattern Definition

Example-based pattern search requires a suitable definition of the query pattern. In this work, we rely on the consistent stream line segmentation for pattern selection and define flow feature *patterns* $\mathcal{P} \subset \mathcal{S}$ as subsets of all segments consisting of $|\mathcal{P}|$ segments. Example query patterns are selected by the user to define flow features for pattern retrieval. Usually, a flow feature pattern consists of distinctive elements of different types of flow features, e.g., a central saddle bounded by two vortical center-like closed stream lines. For pattern search, all segments of the pattern are weighted equally and independent of their spatial extend, i.e., no segment is preferred to evaluate matching quality. Given a query pattern, our method returns geometrically matching occurrences of similar patterns from a variety of different search domains, e.g., the same data set, a different time step, or a different data set. Similar patterns are close to shape-preserving similarity transformations of the query pattern, i.e., they are geometrically close to translated, rotated, and uniformly scaled versions of the query pattern. Hence, for retrieval, the relative pairwise position, orientation, and scale of segments in a pattern is respected by our approach.

5.3.3 Pattern Retrieval

Given a query pattern, we search for differently scaled and geometrically compatible pattern locations in the search domain. Pattern retrieval consists of two consecutive *global* and *local* phases. In the global phase, we fit similarity transformations that align the query pattern to possible occurrences in the search space. In the local phase, each segment of the transformed query pattern is matched with geometrically compatible segments in its vicinity and a match is found if all segments have matching segments in the data. This pattern retrieval procedure is illustrated in Figure 5.6 and we continue to present its details.

Global Pattern Alignment. Global pattern alignment requires the localization of candidate patterns and the computation of the similarity transformations that align the query pattern with candidate matches.

We use the segment similarities presented in Section 5.2.2 for candidate match localization: by comparing the intrinsic similarity of individual pattern segments with segments in the search space it is straightforward to identify possible candidate pattern occurrences for which at least a single segment is matched. Due to the invariance of our segment similarity measures, candidate pattern occurrences are found independently of their location, orientation, and scale.

We observe that it is sufficient to perform similarity computations with a single segment of the query pattern, i.e., using the most distinguishable segment from the pattern that we call the *root segment* $\mathbf{r}(s) \in \mathcal{P}$. We define it to be the segment of highest end point distance. This is because the estimation of similarity of closed segments with coinciding end points depends on the particular location of the junction point and selecting the segment with highest end point distance converses this property. For pattern search in 3D, we additionally require the root segment to have non-vanishing curvature.

Using the root segment, we find a set $\mathcal{Q}$ of similar segments $\mathbf{q}(s) \in \mathcal{Q} \subset \mathcal{S}$ using the scale and orientation-invariant segment similarity For pattern retrieval, we found that it is usually sufficient to use $p = 10\%$ of the most similar segments in $\mathcal{S}$ to $\mathbf{r}$ to define the set of candidate root curve matches $\mathcal{Q}$. Note that the correct pairwise orientation of $\mathbf{r}$ and $\mathbf{q}$ can be determined from the orientation of the minimal oriented difference. Global similarity estimation to the root curve turns out to be the most expensive part of our approach. However, note that pairwise similarities can be precomputed, which enables more responsive pattern retrieval.

Given two similar segments $\mathbf{r}$ and $\mathbf{q}$, we continue to compute the closest similarity transformation that aligns $\mathbf{r}$ to $\mathbf{q}$. The transformation is computed for discretized segments: let $\mathbf{r}_k$ and $\mathbf{q}_k$ denote the vertices along each curve and $|\mathbf{s}|$ the number of vertices of the segment. We resample the shorter curve to have the same number $|\mathbf{r}| = |\mathbf{q}| = m$ of vertices as the longer one. Similarity transformations consists of a translational part $\mathbf{t}$, a rotational part $\mathbf{R}$, and a scaling factor c, such that for an optimal alignment $\mathbf{q}_k = c\,\mathbf{R}\,\mathbf{r}_k + \mathbf{t}$ holds. Optimal alignment of both segments is only possible for intrinsically identical segments. Still, it is possible to fit the closest similarity transformation in least-squares-sense, which aligns both segments sufficiently well as they are known to be similar: given the centers of mass $\bar{\mathbf{r}} = \frac{1}{m}\sum_k \mathbf{r}_k$ and $\bar{\mathbf{q}} = \frac{1}{m}\sum_k \mathbf{q}_k$, the optimal scaling factor relating both segments is given by

$$c = \left(\frac{\sum_k ||\mathbf{r}_k - \bar{\mathbf{r}}||^2}{\sum_k ||\mathbf{q}_k - \bar{\mathbf{q}}||^2} \right)^{\frac{1}{2}} \tag{5.2}$$

(see, e.g., Horn et al. [36]). Independent of this scale the covariance matrix of mass-centered samples

$$\mathbf{K} = \sum_k (\mathbf{r}_k - \bar{\mathbf{r}}) \top (\mathbf{q}_k - \bar{\mathbf{q}}) \tag{5.3}$$

can be used to compute the optimal rotation using a spectral decomposition of $\mathbf{K}$ based on quadratic (2D) or cubic (3D) polynomials (see [36]). Instead, we use the numerically more stable technique proposed by Arun et al. [3]: based on the singular value decomposition $\mathbf{K} = \mathbf{U}\Sigma\mathbf{V}\top$, the optimal rotation is obtained as the polar decomposition $\mathbf{R} = \mathbf{U}\mathbf{V}\top$. Note that different scales of the segments are encoded in Σ and do not contribute to the optimal rotation, which is scale-independent. From this the optimal translational part is given by $\mathbf{t} = \bar{\mathbf{q}} - c\,\mathbf{R}\,\bar{\mathbf{r}}$. This transformation is propagated to all segments of $\mathcal{P}$ to align the query pattern to a candidate match location suitable for local pattern matching.

Local Pattern Matching Global pattern alignment yields approximate locations, orientations, and scales of potential pattern matches by pairing the root segment with intrinsically similar segments globally in the whole search space. Consecutively, in the local pattern alignment step, the occurrence of a proper pattern match is checked by incorporating all pattern segments: for each transformed segment geometrically similar segments in the search space are sought and a pattern match is found if every segment can be matched to a compatible segment.

Note that, for local pattern matching, we can not measure intrinsic segment similarity anymore but instead perform an extrinsic similarity estimation, as for pattern search extrinsically matching segment configurations are required: for two segments $\mathbf{p}$ and $\mathbf{q}$ with vertices $\mathbf{p}_i$ and $\mathbf{q}_j$ (not necessarily of equal count), we measure *extrinsic* shape similarity using the symmetric *Chamfer distance* $e_c(\mathbf{p},\mathbf{q})$ given by

$$e_c(\mathbf{p} \to \mathbf{q}) = \frac{1}{|\mathbf{p}|} \sum_i \min_j ||\mathbf{p}_i - \mathbf{q}_j|| \tag{5.4}$$

$$e_c(\mathbf{p},\mathbf{q}) = \max(e_c(\mathbf{p} \to \mathbf{q}), e_c(\mathbf{q} \to \mathbf{p}))\ . \tag{5.5}$$

The Chamfer distance is successfully applied for other shape matching problems [30] and measures the average sum of closest point distances. It is therefore less sensitive to single distance variations and more robust as, e.g., Hausdorff distances, because all pointwise minimal distances contribute to the final distance. Usually, patterns in vector fields will not match exactly and small variations in the pattern's shape should be admissible. To allow slight variations in segment shape and position, for pattern matching we use the *segment matching cost* function

$$e(\mathbf{p},\mathbf{q}) = e_c(\mathbf{p} - \bar{\mathbf{p}}, \mathbf{q} - \bar{\mathbf{q}}) + w_e||\bar{\mathbf{p}} - \bar{\mathbf{q}}|| \tag{5.6}$$

given as a combination of mass centered extrinsic shape similarity and segment distance expressed by the distance in center of mass. Note that both, the Chamfer distance and center of mass distance, are compatible estimations, as both measure forms of Euclidean distances in the same scale of the current candidate match. The weight w_e allows to balance between the required shape similarity and the allowed segment distance and it can be choose to be $w_e = 1$ for equal weighting.

To locally match a pattern to a candidate match position, let $\mathcal{P}'$ denote the set of segments transformed by the fitted similarity transformation. We obtain the set of candidate match segments $\mathcal{C} \subset \mathcal{S}$ of the underlying flow by also transforming the pattern bounding box $\mathcal{B}$ to $\mathcal{B}'$. Then all segments are included into $\mathcal{C}$ for which at least a single vertex is inside the transformed box $\mathcal{B}'$. The set of matched segments is computed as

$$\mathcal{M} = \left\{ \hat{\mathbf{q}}_i \mid \forall \mathbf{p}'_i \in \mathcal{P}' : \hat{\mathbf{q}}_i = \operatorname*{arg\,min}_{\mathbf{q}_i \in \mathcal{C}} e(\mathbf{p}'_i, \mathbf{q}_i) \right\} , \tag{5.7}$$

i.e., it is given by the segments $\hat{\mathbf{q}}_i \in \mathcal{C}$ that minimize the segment matching costs to each transformed pattern segment $\mathbf{p}'_i$. The total *pattern matching cost* $e(\mathcal{M}) = \sum_i e(\mathbf{p}'_i, \hat{\mathbf{q}}_i)$ is then given by the sum of individual minimal segment matching costs. It represents the quality of the geometric correspondence of the pattern with the currently matched assignment of data segments.

By applying this local matching procedure to all occurrences of segments that are intrinsically similar to the pattern root segment, we compute both all assignments of matched segments and the quality of these matchings. However, as flow features usually vary smoothly in the domain due to the continuity of the flow, it is likely that in the vicinity of a high quality matching several very similar matchings of slightly inferior quality are found. As a single flow feature should only be represented by a single matching pattern, we cluster multiple matchings in a straightforward way by their location: let $\bar{\mathbf{b}}'$ denote the center of mass of the transformed pattern bounding box $\mathcal{B}'$. Then we cluster matched segments based on their location $\bar{\mathbf{b}}'$ by simply discarding a new matching if it found to close to a previous matching and has higher pattern matching costs compared to the previous matching. This way distinct matches of high quality are obtained. Users are then able to browse through the pattern matches in order of their matching costs, or all matches up to a predefined cost value are visualized.

5.4 Validation

In this section, we use the well-known 2D flow behind a CYLINDER that develops a von Kármán vortex street behind an obstacle to validate our algorithm.

In Figure 5.7, we apply our pattern retrieval approach to a single time step of the CYLINDER flow. For a given user-selected flow feature pattern $\mathcal{P}$, we show the resulting pattern matching costs $e(\mathcal{M})$ in the upper part. Pattern matching costs in the vicinity of the original pattern are high due to the similarity of the surrounding repetitive flow features at similar scale. Matching costs slightly increase away from the obstacle due to decreased extrinsic similarity to $\mathcal{P}$. The bottom figure shows the distinctively clustered pattern representatives after aggregating close matchings. The repetitive flow feature is well represented in the matching results.

In another validation example, we apply a pattern from one time step to all other time steps. We demonstrate this in Figure 5.8: given the pattern $\mathcal{P}$ selected in one time step of the previous CYLINDER example (Figure 5.7), we match it to all occurrences at *every* time step of the time-dependent CYLINDER flow. Shown are the matched patterns in all following time steps, which we visualize in space-time domain. The result is consistent with the single time step result and the individual pattern evolution is well-represented. Note that this example does not focus on *tracking* of individual segments over time. Rather, it demonstrates the ability of our approach to match the extrinsic configuration of a consistently segmented pattern from one time step to any other time step.

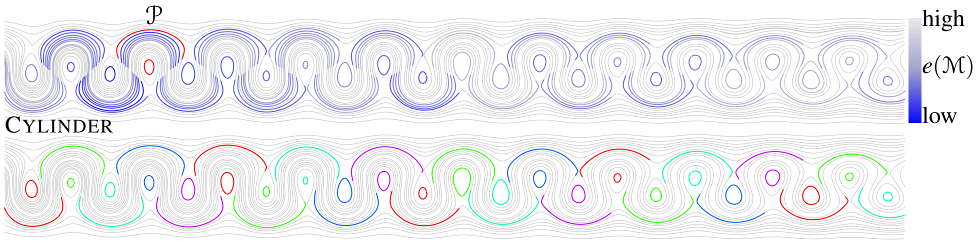


Figure 5.7: Pattern Retrieval in the CYLINDER Flow. In a single time step of the flow behind a circular CYLINDER obstacle (not shown) a flow pattern $\mathcal{P}$ (●) is selected. Our pattern retrieval evaluates the geometric matching costs $e(\mathcal{M})$ of local candidate occurrences of the pattern at different locations, orientations, and scales (top). Clustering of locally similar candidate occurrences yields representative and distinctive pattern matches (bottom, differently colored). The consistent segmentation of this flow is shown in Figure 5.4.

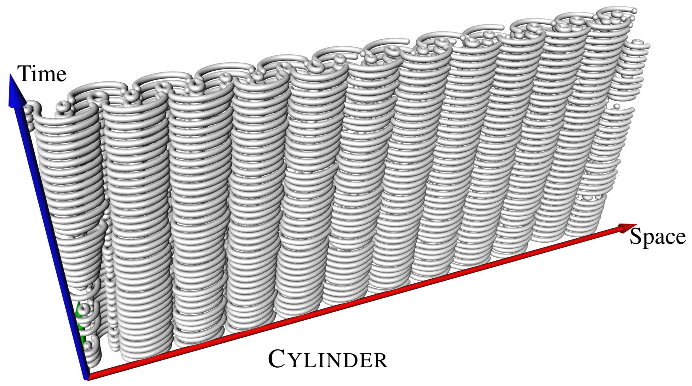


Figure 5.8: Time-dependent Pattern Search. For the time-dependent 2D CYLINDER flow, we perform pattern retrieval for the pattern $\mathcal{P}$ selected in the time-step shown in Figure 5.7. The consistent matching results are visualized in space-time domain.

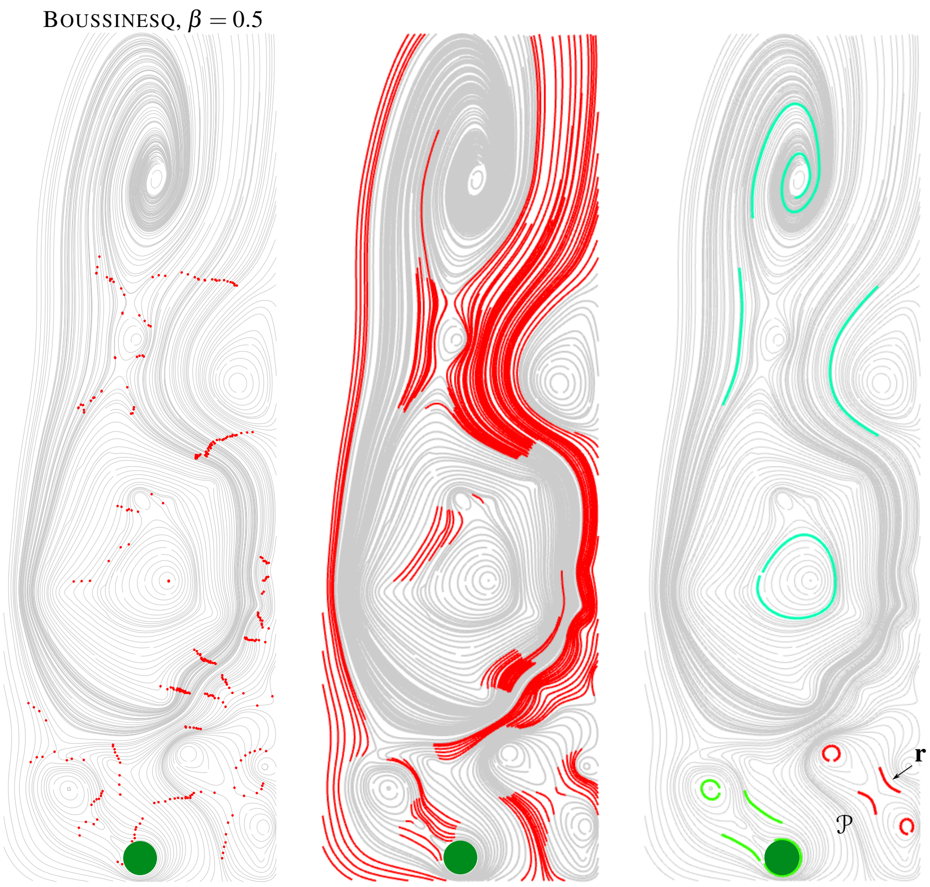


Figure 5.9: BOUSSINESQ Pattern Search. The BOUSSINESQ flow represents the advective mass transport induced by a circular heat source (●). In the segmented flow (left), we search for occurrences of the selected pattern $\mathcal{P}$ (●, bottom right). For the root segment **r** (bottom right), the middle image shows the most similar segments of the data set that are used for matching. The pattern is matched to two different occurrences (right, differently colored) at different locations, orientations, and scales.

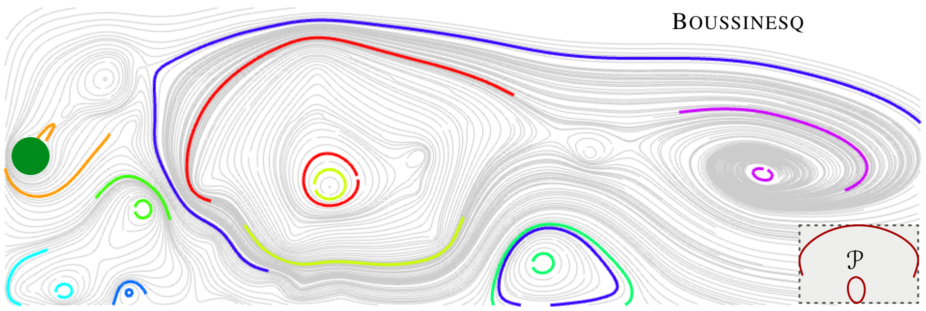


Figure 5.10: Pattern Search with External Pattern. In the BOUSSINESQ flow, we search for the pattern $\mathcal{P}$ (bottom right) that is given by two external segments from the CYLINDER flow shown in Figure 5.7. The pattern is matched to nine occurrences in the BOUSSINESQ flow (differently colored) at various different *nested* locations, orientations, and scales.

5.5 Results

We continue to present results of our pattern retrieval approach in this section. In Figure 5.9, we show pattern search results for the more complex simulated 2D BOUSSINESQ flow representing the Boussinesq approximation applied to solve for the flow generated by a heated cylinder. We show both the segmentation result and the most similar segments to a given root curve of the user selected pattern in the same data set. The flow pattern consists of two vortical regions that are separated by a saddle-like structure. In the same data set, two occurrences are detected. They are matched at very different scales, illustrating the scale-invariance of our approach.

In the same date set illustrated in Figure 5.10, we exemplify the use case of matching with *externally* defined patterns: using the 2D flow pattern defined by consistently segmented segments in the CYLINDER flow of Figure 5.7, we detect nine matching occurrences in the BOUSSINESQ flow. Note that matchings are found at various different locations, orientation, and scales. As long as a query pattern consists of consistently segmented segments it can be used as an *external* pattern in our method. In particular, external patterns need not conform to the scale of the data set due to the invariance to similarity transformations of our method. In addition, this result demonstrates that our approach also retrieves *nested* matches, i.e., pattern occurrences at different scales but at same locations. Nested patterns can be interpreted to give a multi-scale representation of a particular flow pattern. They are supported by our method due to the sparseness of our pattern definition. Note that matching of nested patterns is usually not supported by pattern matching approaches that are defined by dense stencils [10, 27, 35, 68], as this would require self-similar stencils.

In Figure 5.11, we illustrate the application of our approach to 3D pattern retrieval in the electrostatic field around a BENZENE molecule [72]. The specified complex pattern is matched at six accumulation points corresponding to the sixfold molecule symmetry. This example demonstrates that for 3D flows matched patterns are also detected in a rotational-invariant way by our method.

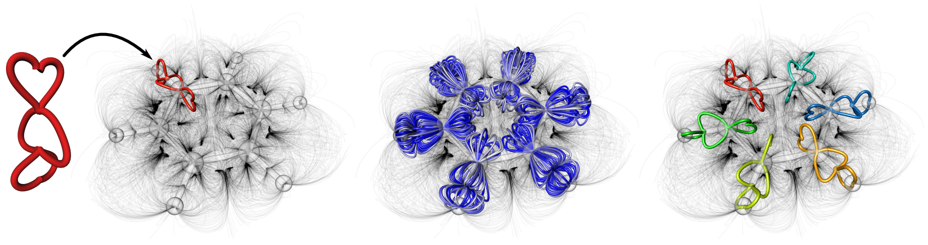


Figure 5.11: The input to our method is a vector field and a user-defined set of stream line segments as a query pattern (● left). We find all "similar" occurrences in a location, translation, and scale-invariant way (● middle). A representative of each cluster is shown on the right.

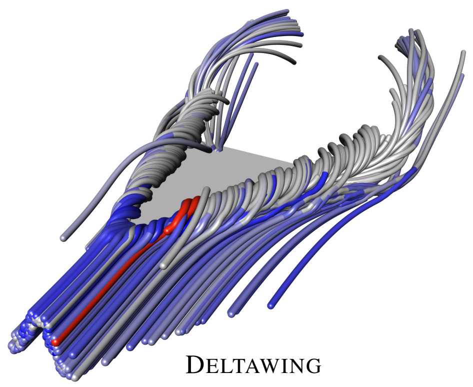


Figure 5.12: 3D Pattern Search in the DELTAWING Flow. The selected pattern $\mathcal{P}$ (●) consists of a straight segment combined with spiraling segment at the tip of one vortex. Segments with small matching costs (●) are detected close to similar regions entering both vortices.

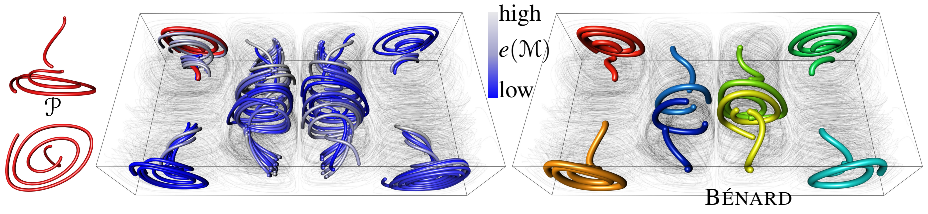


Figure 5.13: 3D Pattern Search in the BÉNARD Flow. For the user-selected pattern $\mathcal{P}$ (● left) in the Rayleigh-BÉNARD convection flow, the pattern matching costs (middle, low costs matches are rendered with thicker lines) indicate eight locations of increased pattern occurrences. All eight distinctive pattern matches are found by clustering these matches for the retrieval result (right).

Data set	$\vert\mathcal{L}\vert$	$\vert\mathcal{S}\vert$	$\vert\mathcal{Q}\vert$	$\vert\mathcal{P}\vert$	SIMI (s)	PSEARCH (s)
CYLINDER	55	400	400	2	0.6	0.3
BOUSSINESQ	172	473	154	4	0.9	1.7
BENZENE	7832	9116	4019	6	162	137
BÉNARD	266	8747	547	2	57	119
DELTAWING	302	1700	1646	2	12	1.6

Table 5.1: Timings. For each data set, we list the number integrated stream lines $|\mathcal{L}|$, number of total segments $|\mathcal{S}|$, the number of considered match candidates $|\mathcal{Q}|$, the number of pattern segments $|\mathcal{P}|$, root curve intrinsic similarity computation (SIMI) as well as the local and global phases of the pattern search with match clustering (PSEARCH).

The DELTAWING flow in Figure 5.12 is a simulated field around a triangle-shaped airplane, courtesy of Markus Rütten (DLR). By selecting a straight stream line segment that has a spiraling segment at the tip of one of the two vortices as the flow feature pattern, our method detects similar stream lines that enter one of both vortices. We only show segments having this characteristic and color code their similarity to the pattern.

The Rayleigh-BÉNARD flow in Figure 5.13 is a simulated data set of fluid motion as the result of thermal convection of a heated and cooled boundaries, obtained using the software NaSt3DGP (University of Bonn). The selected pattern consists of a vortical part with an orthogonally aligned segment. For each segment of the search space the middle image shows the individual pattern matching costs, which results in eight locations of increased pattern occurrences. Segment clustering results in eight representative and distinctive pattern matches.

Table 5.1 summarizes the used number of segments and processing time of all examples of this section. Processing times were measured with a parallel implementation on an Intel Core i7-4770K 3.5GHz quad core system. Our consistent stream line segmentation is a very efficient operation even for a high number of stream lines: for all tested data sets, segmentation time is less than 0.03 seconds. Among all operations, the similarity computation of the root curve to the search space segments is one of the most expensive steps of our method. In fact, the costs are not unexpected, as this operation effectively corresponds to a *global* and *scale-invariant* segment matching. Note that other methods evaluating distances between discretized intrinsic property profiles [49, 54] have similar overall complexity. Slightly higher runtimes are caused by our usage of the more general and accurate, but also more expensive hEMD distance estimation, if compared to standard $\mathcal{X}^2$ or EMD distances. Similar to similarity estimations, pattern search performance depends on the number of tested segments and their vertex count, which we found to be highly data set-dependent. Still, all operations can be parallelized in a straightforward way for increased performance.

5.6 Discussion

In our approach, we design the retrieval of flow feature patterns based on segmentations of stream lines. Compared to existing methods, this has a number of implications w.r.t. stream line similarity estimation and flow pattern matching.

For stream line similarity estimation, a number of previous methods either rely on extrinsic [9, 62], intrinsic [54], or edit distances [82] of *whole* stream lines. As emphasized by Lu et al. [49], the comparison of long curves based on intrinsic properties quickly leads to ill-posed similarity estimations, and they propose a segmentation-based approach instead. Similar EMD-based segmentation differences are estimated in their approach. As their method is not scale-invariant and normalizes intrinsic profiles, a more efficient EMD variant can be used. However, whole stream lines are still the entities for which similarities are estimated in their approach, albeit based on individual segmentations that can vary in number. Again, this leads to ill-posed similarity estimations if curves of different segment number are compared, e.g., a single-arc curve with a multi-arc curve. In contrast, in our work we emphasize the importance of only comparing compatible curve segments that usually represent a single dominant region of maximal curvature. On the one hand, this restricts the spatial extend of the comparable entities, on the other hand, similarity estimations become more reliable. In a similar way, Tao et al. [73] perform stream line segmentation and define patterns as a series of partially matching segments *along* individual stream lines, enabling partial stream line matching. This differs from our segmentation-based approach in that our patterns are not restricted to belong to single stream lines. In fact, compared to related methods for dense [10, 27, 35, 68] and sketch-based [82] flow pattern matching approaches, our method explicitly supports the definition of sparse sets of stream line segments as query patterns. This increases the flexibility of pattern definition and enables a greater range of possible pattern retrieval applications, e.g., the transfer of patterns to external data sets or the matching of nested flow patterns.

Limitations and Outlook. Although stream line segmentation enhances the reliability of segment similarity computations, we identify a number of related drawbacks. First, as we define flow patterns as sets of stream line segments, users must stick to these segments when defining patterns. Artificially designed patterns will often not match properly. Usually this is unproblematic, since segments correspond to intuitive and distinctive flow regions due to the consistent curvature-based segmentation. However, certain types of flow patterns are harder to describe this way: an example is the combination of a vortical region with a *straight* curve of limited extend in its vicinity, because straight curves will not be segmented into individual segments due to the absence of curvature. It is an interesting direction for further research to identify alternative curve segmentations, e.g., hierarchical multi-resolution schemes for segmentation and similarity computation, which could alleviate this limitation.

We see a second limitation in the usage of the extrinsic Chamfer distance for local segment matching: it represents a purely geometric curve distance measure and is not related to the potential pattern deformation due to underlying flow characteristics. Therefore, it has a limited problem-dependent range in distinguishing purely geometric dissimilarities from segment deformations due to local vector field advection. We believe it is an interesting direction for further studies to formulate local distance measures that differentiate between purely geometric distances and distances induced by the local vector field. On the other hand, our purely geometric formulation makes our approach also directly applicable to different field types, e.g., time-dependent flows and tractography data. Hence, in the future, we would like to generalize our approach to more general time-dependent line fields such as streak line fields [84, 86, 88] and fiber bundles [16, 42], which also potentially benefit from segment-based pattern definition and retrieval.

5.7 Summary

In this chapter, we presented a novel approach to pattern retrieval in flows that is based on a consistent stream line segmentation. Flow patterns are defined by sparse sets of stream line segments. This provides flexibility for their definition. They are matched independently of position, location, and scale to the same data set, a different time step, or even a different data set. We demonstrate the pattern retrieval effectiveness on a number of 2D and 3D data sets. Efficient pattern matching is enabled by a new stream line segmentation scheme that is solely based on intrinsic curve properties and segments intrinsically similar stream lines in a scale-invariant and feature-consistent way. Based on this segmentation, intrinsic segment similarity estimates are proposed that are invariant w.r.t. rigid or similarity transformations.

Chapter 6

3D Face Template Registration Using Normal Maps

6.1 Overview

3D face models are widely used for computer graphics and computer vision applications. Of particular interest are morphable 3D face models that are based on a single deforming 3D template mesh, which can represent different individuals or different facial expressions. The template deformation is typically controlled via a small set of parameters. Examples are hand-crafted blend shapes [22] or learned morphable face models [6], which are based on the analysis of a large database of 3D laser scans. In order to generate a morphable face model, a template mesh has to be registered to all 3D laser scans in the database. After registration a template vertex with a particular index in the template mesh is located at the same semantically corresponding point in all 3D scans (e.g., vertex no. 101 is always the tip of the nose, vertex no. 512 is the corner of the right eye, and so on). The registration is a crucial step in the generation of a morphable face model because once corresponding vertex positions are established in all the database exemplars, it is already possible to perform a linear blend between the exemplars to generate new individuals or interpolated facial expressions. To reduce the number of blending weights, typically a Principle Component Analysis (PCA) is performed, which generates a low dimensional parameter space that can still represent the observed differences in the vertex positions of the exemplars [6].

Using a so called *light stage* (a sphere with a large number of individually controllable light sources) a very detailed 3D scan of a human face can be captured [1]. Here, the low resolution 3D geometry is typically acquired using a structured light approach, and the fine details are captured via normal map generation. The normal maps are created by taking images under 4 to 7 different illumination conditions that are generated with the light stage. In addition, a projector is used to generate a series of stripe patterns (typically 5 to 15 patterns of increasing frequency) for the structured light reconstruction. Though projector and light stage patterns could be in theory displayed at fast succession, high frame rates are difficult to achieve in practice due to frame rate limitations of the camera as well as switching time limitations of the light stage and the projector. Consequently, the captured subject should not move during the acquisition, which is quite challenging, especially for less relaxed facial expressions.

In order to reduce the capturing time and effort, we propose in this paper a method to register a 3D face template only to the normal maps. The omission of structured light scanning reduces the capturing time by almost 50 percent. We claim that it is possible to skip structured light scanning because the low resolution 3D geometry is already approximately given by the initial 3D face template. However, in our experiments we found that a normal map from a single camera view can not resolve the depth ambiguities. Consequently, our approach uses multiple normal maps that are generated simultaneously by observing the face with multiple cameras, which does not increase the capturing time.

State-of-the-art approaches [1,2,6] use non-rigid ICP algorithms to fit a 3D template mesh to point cloud data (that is obtained via laser or structured light scanning). In our set-up this non-rigid ICP algorithm is replaced by an algorithm that registers a 3D face template to several normal maps. Thereby, the proposed registration approach performs three steps. First, some manually selected feature points and their projections on the normal maps are registered to roughly align the template. Second, a normal registration method is applied to align the template semantically to the normal maps. This step aims to find the correlation between the template geometry and the geometry information encoded in the normal maps. The result of this step is a deformed template mesh that better resembles the geometry of the real subject, but still maintains its basic structure. Third, to further refine the shape of the template, a shape refinement is executed. In this step, we employ the constraint that for a given 3D position, its projections in neighbouring views should have the same normals.

The proposed algorithm resides on the following core contributions:

- A novel method to semi-automatically fit a 3D face template to normal maps. This includes three main steps: feature point registration, normal registration, and shape refinement.
- In normal registration, a novel optimization strategy to minimize a highly non-linear function is proposed. It splits the problem to several constrained optimization steps which can be linearised and solved efficiently.

As structured-light scanning is omitted, the acquisition time can be reduced by over 50%, while the fitting result is still accurate.

The rest of this chapter is organized as follows. In Section 6.2, the employed hardware set-up for normal map generation is introduced. The proposed template registration algorithm is described in Section 6.3. In Section 6.4, several experiments are presented to evaluate our algorithm. This chapter ends with concluding remarks and future works in Section 6.5.

6.2 Hardware Set-up and Normal Map Generation

The employed data capturing system is shown in Fig. 6.1. It comprises of a light stage consisting of 156 LEDs arranged on a spherical metal frame and six digital cameras. The

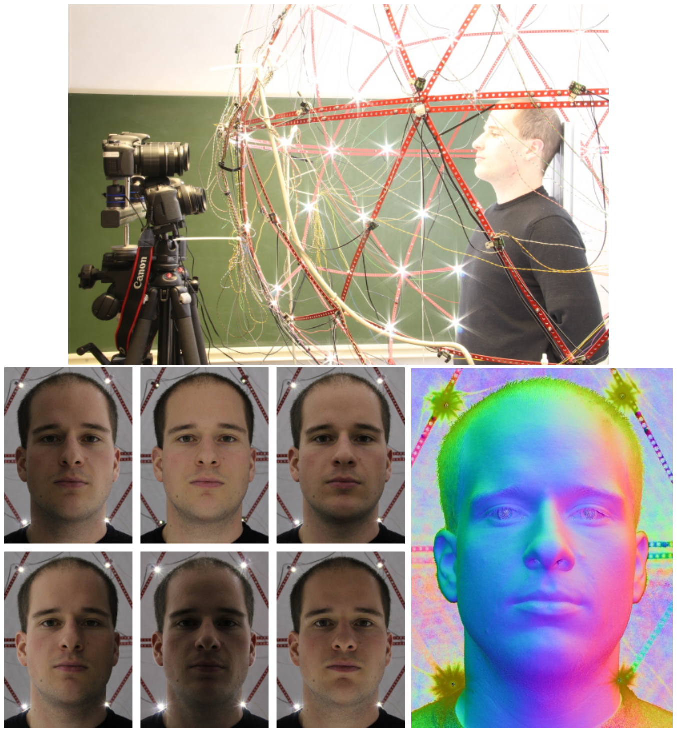


Figure 6.1: (from left to right) The employed data capturing system comprising a light stage, which can produce different illumination patterns, and several digital SLR cameras; six spherical gradient illumination patterns; normal map computed from the images of the six gradient illumination patterns.

light stage is used to produce six axis parallel spherical gradient illuminations. The set of images captured by the c-th camera is denoted as $\mathcal{L}_c = \left\{\mathbf{L}_c^x, \mathbf{L}_c^{-x}, \mathbf{L}_c^y, \mathbf{L}_c^{-y}, \mathbf{L}_c^z, \mathbf{L}_c^{-z}\right\}$. For the spherical gradient illumination the intensity values of the LEDs are translated and shifted to the range $[0,1]$, since negative light cannot be emitted. The normal map of the c-th camera can be calculated using the spherical gradient illuminations in a pixel-wise manner (as proposed in [90]):

$$\mathbf{N}_c = \frac{\left(\mathbf{L}_c^x - \mathbf{L}_c^{-x}, \mathbf{L}_c^y - \mathbf{L}_c^{-y}, \mathbf{L}_c^z - \mathbf{L}_c^{-z}\right)^\top}{\left|\left|\left(\mathbf{L}_c^x - \mathbf{L}_c^{-x}, \mathbf{L}_c^y - \mathbf{L}_c^{-y}, \mathbf{L}_c^z - \mathbf{L}_c^{-z}\right)^\top\right|\right|} \quad . \tag{6.1}$$

The digital cameras are calibrated with a calibration pattern. The calibration pattern is placed inside the light stage in an axis-aligned way such that its center coincides with the center of the light stage. This assures that the cameras are calibrated with respect to the light stage coordinate system. During camera calibration we employ the focal length given by the EXIF data provided by the camera and estimate the extrinsic camera parameters with Tsai's approach [78]. Then a bundle adjustment is performed to further refine the extrinsic camera parameters. The size of the normal maps used in our experiments is 2592×1728.

6.3 Algorithm

The proposed algorithm aims to register a mesh template to multi-view normal maps. The input consists of a face template, given as a polygonal mesh $\mathcal{S} = \{\mathcal{V}, \mathcal{E}\}$ which is defined by a set $\mathcal{V}$ of vertices $\mathbf{V}_i$ and a set of edges $\mathcal{E}$. Also, the set-up described in Section 6.2 provides the camera perspective projection matrices $\mathbf{P}_c$, and normal maps $\mathbf{N}_c$ for each camera (with index c) of the camera set $\mathcal{C}$. The output is a deformed template face which is fitted to the observed face.

The algorithm has three steps. Firstly, to roughly align the template and the normal maps, we manually select eight 3D feature points for the template and their projections for all the views. We register these eight points to the normal maps, and the rest of the template deforms smoothly. Secondly, ignoring the neighbour-view consistency, a normal registration method is executed to align the vertices of the template to their semantically correct positions in all the normal maps. Thirdly, through a multi-view refinement, the shape of the face is further improved by enforcing neighbouring-view consistency.

These three steps of the algorithm are described in the following three subsections in detail.

6.3.1 Feature-based registration

This step aims to match eight 3D feature points of the template to a set of user-defined 2D locations, while the rest of the vertices should deform smoothly. In our experiments we used the eight feature points visualized in Fig. 6.2. The problem is solved as a non-rigid registration problem.

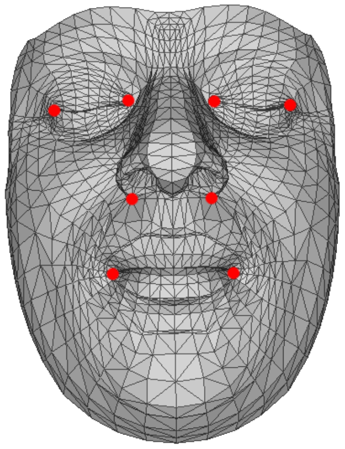


Figure 6.2: The template used in our experiments. It has 1250 vertices in total. The red points indicate the 3D feature points.

We assign each vertex of the template $\mathbf{V}_i = (v_x, v_y, v_z)^\top$ a translation vector $\mathbf{T}_i = (t_x, t_y, t_z)^\top$. To enforce that the back-projection of a translated vertex is located at a user-defined 2D location $\mathbf{u} = (u_x, u_y)^\top$ in the normal maps, an energy term is defined by

$$\mathbf{E}_{corner} = \sum_{c \in \mathcal{C}} \sum_{\mathbf{u}_{ic} \in \mathcal{U}} ||\mathbf{u}_{ic} - \mathbf{P}_c (\mathbf{V}_i + \mathbf{T}_i)||_2^2 \quad , \tag{6.2}$$

where $\mathbf{P}_c\{\cdot\}$ is a projection function which projects a 3D point to a 2D location in the image plane of the c-th camera; a set $\mathcal{U}$ includes all the user-defined 2D locations $\mathbf{u}$. To make the rest of the vertices deform smoothly, we constrain the translations of two connected vertices in the template mesh to be similar. The smoothness term is given by

$$\mathbf{E}_{smooth} = \sum_{(i,j) \in \mathcal{E}} ||\mathbf{T}_i - \mathbf{T}_j||_2^2 \quad . \tag{6.3}$$

Combining Eq. (6.2) with Eq. (6.3), the total cost can be written as

$$\mathbf{E} = \mathbf{E}_{corner} + \lambda \mathbf{E}_{smooth} \tag{6.4}$$

where λ is a weighting factor. The result is obtained by minimizing Eq. (6.4) using the non-linear optimizer.

Non-linear optimization Since projection from 3D space to 2D space is non-linear, Eq. (6.4) becomes a non-linear optimization problem. We solve it as a non-linear least squares problem iteratively. The projection process is given as

$$m \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} \bar{u} \\ \bar{v} \\ m \end{pmatrix} = \mathbf{P}_c (\mathbf{V}_i + \mathbf{T}_i) \quad , \tag{6.5}$$

where u and v are the coordinate elements of a 2D location. To linearise this process, we can calculate the derivatives of a 2D coordinate, i.e. $\frac{\partial u}{\partial \mathbf{T}_i}$ and $\frac{\partial v}{\partial \mathbf{T}_i}$, analytically. Then the Jacobian $\mathbf{J}$ of Eq. (6.2) is the concatenation of

$$\mathbf{J}_i = \left(\cdots - \frac{\partial u}{\partial \mathbf{T}_i} \quad - \frac{\partial v}{\partial \mathbf{T}_i} \cdots \right) \tag{6.6}$$

for each vertex. The Jacobian of Eq. (6.3) can be written as

$$\mathbf{H} = \mathbf{D} \otimes \mathbf{I}_{3\times 3} \quad , \tag{6.7}$$

where $\mathbf{D}$ is the node-arc incidence matrix [21], and $\otimes$ is the Kronecker product operator. By combining Eq. (6.6) and Eq. (6.7), the final over-determined linearised equation system can be written as

$$\underbrace{\begin{pmatrix} \mathbf{J} \\ \lambda\,\mathbf{H} \end{pmatrix}}_{\mathbf{B}} \Delta\mathbf{T} = -\underbrace{\begin{pmatrix} \mathbf{r} \\ \mathbf{0} \end{pmatrix}}_{\mathbf{b}} \quad , \tag{6.8}$$

where the residual vector $\mathbf{r}$ is the concatenation of $\mathbf{r}_i = \mathbf{u}_{ic} - \mathbf{P}_c(\mathbf{V}_i)$ for each vertex. The resulting normal equation is given as

$$\mathbf{B}^\top \mathbf{B} \Delta\mathbf{T} = \mathbf{B}^\top \mathbf{b} \quad . \tag{6.9}$$

Since the coefficient matrix in Eq. (6.9) is large and sparse, it can be solved efficiently using the iterative conjugate gradient method.

6.3.2 Normal registration

After feature-based registration, the template is roughly aligned. The purpose of normal registration is to semantically register the template to the normal maps. The constraint is that the projections of the normals of the template should be equal to the normals in the normal maps.

The main difference between color images and normal maps is that normal maps encode the geometric information, e.g. convex and concave positions, as a normal distribution. This means a surface can be recovered using normal integration from one normal map only, while for surface reconstruction multiple color images are needed (approximation techniques such as shape from shading can recover a surface from only one color image, but it also needs to approximate a normal map from the image). In order to overcome the influence of occlusions and also the errors in normal maps, an approach based on multi-view normal registration is proposed.

Since the three elements of a normal $\mathbf{n} = (n_x, n_y, n_z)^\top$ are correlated, e.g. $||\mathbf{n}||_2 = 1$, first we re-parametrize a normal to spherical coordinates, so that we can use two independent angular components θ and ϕ to represent a normal. Then the cost for the normal similarity is defined as:

$$\mathbf{E}_{normal} = \sum_{c\in\mathcal{C}} \sum_{i\in\bar{\mathcal{V}}_c} \sum_{p\in\{\theta,\phi\}} \sum_{\mathbf{w}\in\mathcal{W}} (r_{cip\mathbf{w}})^2 \quad , \tag{6.10}$$

where $\bar{\mathcal{V}}_c$ is the set of vertices which are not occluded in the view of camera c, p is the element index of a normal in spherical coordinates, $\mathcal{W}$ is a square window. Here, $r_{cip\mathbf{w}}$ is defined as

$$r_{cip\mathbf{w}} = \mathbf{G}(\mathbf{w}) \cdot \mathbf{N}_c \left(\mathbf{P}_c \left(\mathbf{V}_i + \mathbf{T}_i\right) + \mathbf{w}\right)_p - \tilde{\mathbf{N}} \left(\mathbf{V}_i + \mathbf{T}_i\right)_p \quad , \tag{6.11}$$

where $\mathbf{G}$ is a Gaussian kernel function defined on the window $\mathcal{W}$, and $\mathbf{N}_c(\cdot)$ is a function which takes a pixel position as the input and returns the normal at the given pixel position of the c-th normal map.

Considering that the semantics of the template changes when some non-smooth deformations are applied, a smoothness term which not only allows a big range of deformation but also maintains the basic structure of the template is needed. In this paper, instead of using absolute translations, i.e. Eq. (6.3), to ensure the mesh rigidity, we employ mesh differentials as used in [71] to define the smoothness term. It can be written as

$$\mathbf{E}_{smooth} = \sum_{\mathbf{V}_i \in \bar{\mathcal{V}}} ||\mathbf{V}_i + \mathbf{T}_i - \frac{1}{k} \sum_{\mathbf{V}_j \in \mathcal{N}\{\mathbf{V}_i\}} (\mathbf{V}_j + \mathbf{T}_j)||_2^2 \quad , \tag{6.12}$$

where $\mathcal{N}\{\cdot\}$ represents a set of neighbouring vertices.

Combining with the cost of the feature points in Eq. (6.2), the result can be obtained by minimizing the cost function

$$\mathbf{E} = \mathbf{E}_{normal} + \alpha \mathbf{E}_{smooth} + \beta \mathbf{E}_{corner} \quad , \tag{6.13}$$

where α and β are weighting factors.

Non-linear optimization In Eq. (6.11), $\mathbf{N}_c(\cdot)$ given a 2D image location returns the normal from the normal map of view c. Since the vertex' normal can be changed by deforming its neighbouring vertices, the representation of a vertex normal by using the positions of its neighbouring vertices, i.e. $\tilde{\mathbf{N}}(\cdot)$, is a non-linear function. Concluded from that, Eq. (6.13) is a highly non-linear function. In particular, we find that the minimization of Eq. (6.13) cannot be solved by direct linearisation. In this paper, we address this problem by separating the whole optimization to several constrained optimization steps. These steps are:

1. Update the normals of the template using the current structure of the template, and fixate the normals.

2. Optimize Eq. (6.13) iteratively until the largest translation of all the vertices is smaller than a threshold.

3. If no further optimization of Eq. (6.13) is achieved, then finish, otherwise go back to step 1.

In step 2, when we fixate the normals of the template, Eq. (6.11) becomes

$$r_{cip\mathbf{w}} = \mathbf{G}(\mathbf{w}) \cdot \mathbf{N}_c \left(\mathbf{P}_c \left(\mathbf{V}_i + \mathbf{T}_i\right) + \mathbf{w}\right)_p - \mathbf{m}_{ip} \tag{6.14}$$

where $\mathbf{m}_i$ represents the current fixated normal of the i-th vertex. We solve Eq. (6.14) by linearization. The Jacobian of Eq. (6.14) can be calculated as

$$\begin{aligned}\mathbf{R}_{cip\mathbf{w}} &= \mathbf{G}(\mathbf{w})\frac{\partial \mathbf{N}_c\left(\mathbf{P}_c\left(\mathbf{V}_i+\mathbf{T}_i\right)+\mathbf{w}\right)_p}{\partial \mathbf{T}_i} \\ &= \mathbf{G}(\mathbf{w})\left(\frac{\partial \mathbf{N}_{cp}}{\partial \mathbf{n}_x}\frac{\partial \mathbf{n}_x}{\partial \mathbf{T}_i}+\frac{\partial \mathbf{N}_{cp}}{\partial \mathbf{n}_y}\frac{\partial \mathbf{n}_y}{\partial \mathbf{T}_i}\right) \quad ,\end{aligned} \tag{6.15}$$

where $\mathbf{N}_{cp}$ is an abbreviation of $\mathbf{N}_c\left(\mathbf{P}_c\left(\mathbf{V}_i+\mathbf{T}_i\right)+\mathbf{w}\right)_p$, and $\mathbf{n} = \{\mathbf{n}_x, \mathbf{n}_y\}^\top$ is a 2D image location.

In Eq. (6.15), $\frac{\partial \mathbf{n}_x}{\partial \mathbf{T}_i}$ and $\frac{\partial \mathbf{n}_y}{\partial \mathbf{T}_i}$ can be calculated analytically, and $\frac{\partial \mathbf{N}_{cp}}{\partial \mathbf{n}_x}$ and $\frac{\partial \mathbf{N}_{cp}}{\partial \mathbf{n}_y}$ are the normal gradients in image domain. Since we re-parametrize the normal representation to spherical coordinates, the two angles θ and ϕ are independent, so that we can interpolate a normal by performing a finite difference operation.

The Jacobian of Eq. (6.12) can be written as a constant matrix $\mathbf{Q}$ defined as follows:

$$\mathbf{Q}_{ij} = \begin{cases} 1, & i=j \\ -\frac{1}{k_i}, & \mathbf{V}_j \in \mathcal{N}(\mathbf{V}_i) \\ 0, & \text{else} \end{cases} \quad , \tag{6.16}$$

where k_i is the number of the neighbouring vertices of $\mathbf{V}_i$.

The final linear equation system of Eq. (6.13) can be written as

$$\underbrace{\begin{pmatrix} \mathbf{R} \\ \alpha\,\mathbf{Q} \\ \beta\,\mathbf{J} \end{pmatrix}}_{\mathbf{B}} \Delta\mathbf{T} = -\underbrace{\begin{pmatrix} \mathbf{h} \\ \alpha\,\mathbf{s} \\ \beta\,\mathbf{r} \end{pmatrix}}_{\mathbf{b}} \quad , \tag{6.17}$$

where $\mathbf{R}$ is the concatenations of Eq. (6.15), $\mathbf{Q}$ is defined as in Eq. (6.16), $\mathbf{h}$ is the normal residual defined as the concatenation of

$$\mathbf{h}_{cip} = \sum_{\mathbf{w}\in\mathcal{W}} \mathbf{G}_\mathbf{w}\cdot \mathbf{N}_c\left(\mathbf{P}_c\left(\mathbf{V}_i\right)+\mathbf{w}\right)_p - \mathbf{m}_{ip} \quad ,$$

$\mathbf{s}$ is the smoothness term residual defined as the concatenation of

$$\mathbf{s}_i = \mathbf{V}_i - \frac{1}{k}\sum_{\mathbf{V}_j\in\mathcal{N}\{\mathbf{V}_i\}}(\mathbf{V}_j) \quad ,$$

$\mathbf{J}$ and $\mathbf{r}$ are the correlated feature point terms defined in Eq. (6.8). As in Eq. (6.9), the resulting normal equation can be solved efficiently by the conjugate gradient method.

6.3.3 Multi-view Refinement

After normal registration, the position of the vertices of the template are much closer to their semantically correct positions in the normal maps. However, a shape refinement is needed to further correct the resulting surface for two reasons: First, the smoothness term that was employed in normal registration maintains the basic structure of the template, and second, in order to make the optimization solvable, we use several constrained optimization steps to approximate the original problem formulation. In this refinement step, we make use of the neighboring view consistency information to enforce that the projections of a vertex to a pair of neighboring camera views should have the same normal in both normal maps. Since the shape refinement only refines the 3D shape but does not have any concept of semantics, a good initial result of normal registration from the previous step is required. A good result means that, after normal registration, vertices are moved very close to their semantically correct positions and the back-projections are consistent in all views.

We formulate the optimization problem as

$$\begin{aligned} \operatorname*{argmin}_{\mathbf{t}_i} & \sum_{(s,t)\in\mathcal{M}} \sum_{i\in\bar{\mathcal{V}}_s\cap\bar{\mathcal{V}}_t} \sum_{p\in\{\theta,\phi\}} \sum_{\mathbf{w}\in\mathcal{W}} (a_{stip\mathbf{w}})^2 \\ & + \sigma \sum_{(i,j)\in\mathcal{E}} \|\mathbf{T}_i - \mathbf{T}_j\|_2^2 \\ & + \tau \sum_{c\in\mathcal{C}} \sum_{\mathbf{u}_{ic}\in\mathcal{U}} ||\mathbf{u}_{ic} - \mathbf{P}_c(\mathbf{V}_i + \mathbf{T}_i)||_2^2 \quad , \end{aligned} \tag{6.18}$$

where

$$\begin{aligned} a_{stip\mathbf{w}} = & \mathbf{N}_s(\mathbf{P}_s(\mathbf{v}_i + \mathbf{t}_i) + \mathbf{w})_p \\ & - \mathbf{N}_t(\mathbf{P}_t(\mathbf{v}_i + \mathbf{t}_i) + \mathbf{w})_p \end{aligned} \tag{6.19}$$

is the difference of the two normals in normal map $\mathbf{N}_s$ and $\mathbf{N}_t$, $\mathcal{M}$ is a set which consists of all the available camera pairings, σ and τ are weighting factors. The smoothness term and feature data term in Eq. (6.18) are the same as the ones in the feature-based registration in Eq. (6.3) and Eq. (6.2). This optimization problem can also be solved by linearisation. The Jacobian of the two parts of Eq. (6.19) are calculated as in Eq. (6.15), and the final normal equations are similar to Eq. (6.9).

Multi-resolution Both in the normal registration and multi-view refinement, a multi-resolution approach is employed to improve efficiency and accuracy. In the experiment, we use three layers with different resolutions. Since the three elements of a normal in Cartesian coordinates are correlated, we first convert the normal representation from Cartesian coordinates to spherical coordinates, and employ a Gaussian kernel in this domain to blur the images. While processing each layer, the weighting parameters of the cost function are fixated.

6.4 Results

Our template fitting approach is evaluated with synthetic data as well as real data. Both types of experiments are executed with the same camera setup. We use only 6 cameras to cover the frontal face, and the face template shown in Fig. 6.2 in all experiments.

In the synthetic data experiment, we use a 3D head model to represent the human head, and render it in 3D Max to generate normal maps. The size of the model is similar

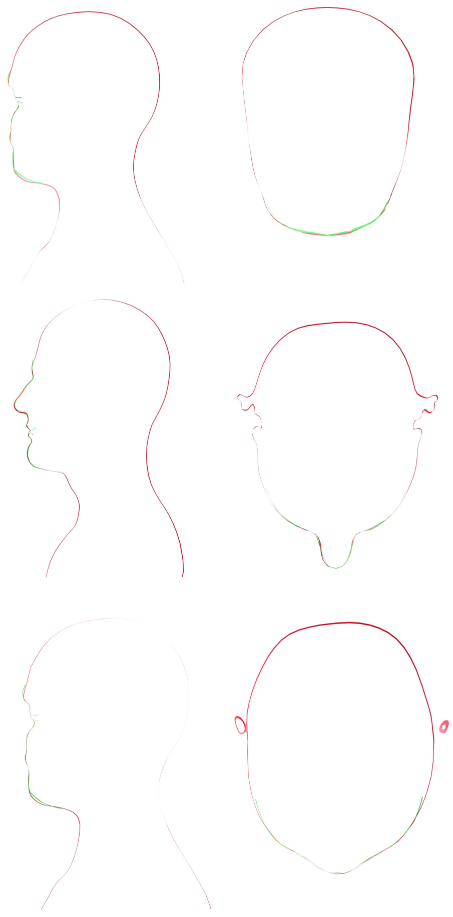


Figure 6.3: Some slices of the synthesis ground-truth model and our result. The red contour indicates the ground-truth model, and the green contour indicates our result. The left column includes three vertical slices, and the right column includes three horizontal slices. (To evaluate the overlap clearly, an interested reader can open a digital version of this paper and can zoom into the figure)

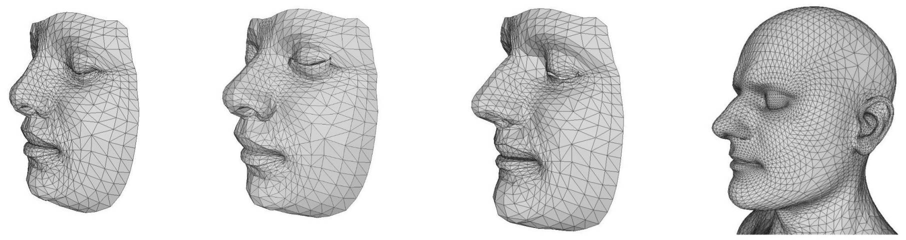


Figure 6.4: Template fitting progress of our algorithm. From left to right: feature-based registration result, normal registration result, final result, and the model used to generate the synthetic ground-truth input data.

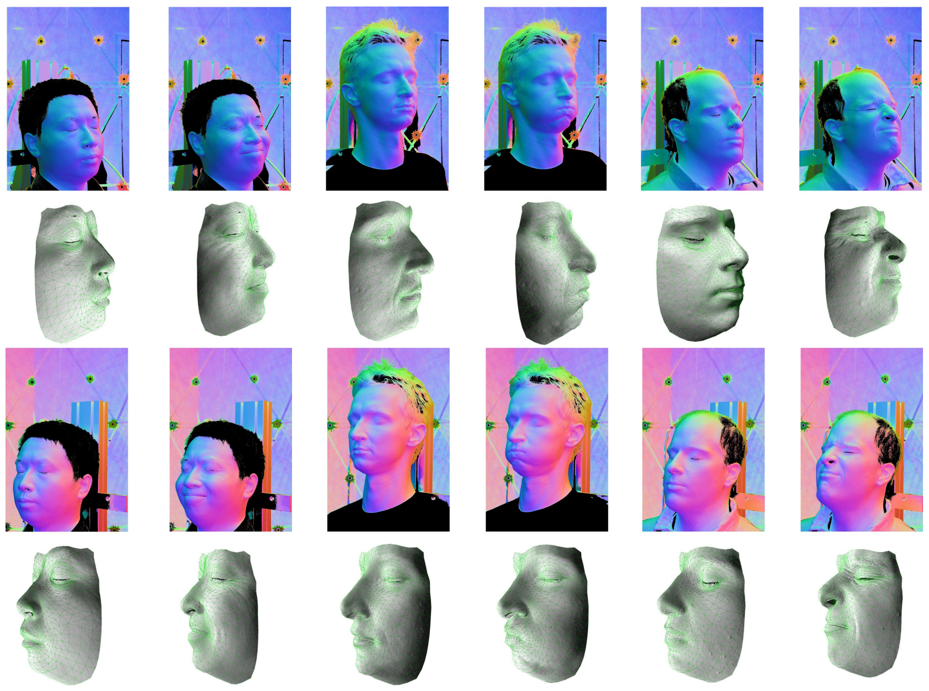


Figure 6.5: Quantitative results. Each column represents a test. For any column, the first and the third figures are two of the normal maps used in the test; The second and the forth figures are two views of the result with normal map textured and template mesh overlapped.

to the size of a real human head. Fig. 6.4 shows the result after each optimization step. We can see that after feature-based registration, the shape is still dissimilar to the input head model. After normal registration, the face is deformed closer to the head model but still keeping the basic shape of the face template. This step is performed to ensure that each semantic position, i.e. concave and convex positions, is moved closer to the corresponding position of the head model. The next step refines the shape of the face. To compare the final result with the synthetic model, we show the comparison with several horizontal and vertical slices in Fig. 6.3. Since the mesh of the face template is sparser than the synthetic model, the two contours cannot be perfectly matched. We can see that larger errors appear only at the boundary of the face. That is because the camera matrix only covers the frontal area of the face. When computing the Hausdorff distance between the model and our result, the mean error is 1.65mm and the root mean squared error(RMSE) is 4.62mm.

In the real data experiment, we evaluate our algorithm with three subjects. The result is shown in Fig. 6.5. For each subject, we use a neutral face and a face with an expression. Since eyes and mouth have very complicated structures, in all experiments, we ask the subjects to close their eyes and mouth. We can see from the result that the normal maps have high-resolution, but our template is comparatively sparse. Consequently, our algorithm can only recover the most significant features of a face, some subtle features such as skin foldings are not captured. All computations are performed on a single consumer-level computer, and the running time of the complete algorithm is about 5 minutes. The most time-consuming operation is to solve the large sparse linear systems. In this paper, we solve these system directly using the conjugate gradient method. If a factorization step, such as Cholesky factorization, is applied, the execution time can be further reduced.

6.5 Summary

We have presented a semi-automatic approach to fit a template mesh to multi-view normal data. This approach reduces the acquisition time compared to state-of-the-art approaches which employ structured light scanning to generate the low-resolution 3D reconstruction. The method consists of three steps: feature-based registration, normal registration, and multi-view shape refinement. In the feature-based registration, we match a few manually selected feature points. In the normal registration, we deform the template to align semantic positions to the normal data. Since the resulting cost function is highly non-linear, we propose a linearisation method for efficient optimization. In the multi-view refinement step, we further refine the shape by enforcing the consistency of normals in neighbouring views.

Chapter 7

Conclusion

In this thesis, we make contributions for pattern search in visualizatoinin. We aim to find similar patterns in different types of data sets: scalar fields, vector fields, and line fields. To achieve this goal, we propose three algorithms for these types of data individually.

Specifically, for scalar fields, we extract a sparse set of features using the 3D SIFT algorithm (Scale-Invariant Feature Transform). This allows for a memory-saving description of prominent features in the data with invariance to translation, rotation, and scaling. Then, the user can define a pattern as a set of SIFT features in multiple scalar fields by e.g. brushing a region of interest. The proposed algorithm can therefore locate and rank matching patterns in the entire data set.

For vector fields, 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 order to support patterns of arbitrary shape and extent. We assemble these patterns using several smaller spheres.

For line fields, we propose a method that allows users to define flow patterns in form of a sparse sets of stream line segments. They are defined sparsely and can have a significant extent, i.e., they are integration-based and not local. This allows for a greater flexibility in defining features of interest. Our method starts with splitting stream lines using globally-consistent segmentation criteria. It strives to maintain the visually apparent features of the flow as a collection of stream line segments. Most importantly, it provides similar segmentations for similar flow structures.

Additionally, we also investigate the shape recovery problem from multiple views. In order to reduce the acquisition time, we omit the structured light scanning step to obtain low-frequency 3D information and rely solely on normal maps from multiple views. The normal map is a 2D vector field, which is generated using spherical gradient illuminations in a light stage. In contrast to shape reconstruction algorithms, we propose a semi-automatic method to fit a template mesh to such normal maps, which includes three main steps: feature point registration, normal registration, and shape refinement. This reduces the acquisition time by over 50 percent. In our experiments the proposed algorithm is successfully applied to real faces of several subjects. Experiments with synthetic data show that the fitted face template can closely resemble the ground truth geometry.

7.1 Future Work

Despite the algorithms proposed in this thesis improve the application for both problems, i.e., pattern search and shape recovery, there are still some challenging directions for further exploration. We list some of them as follows and discuss the potential future work.

For pattern search in flow visualization, we have discussed scalar, vector, and line fields. Another widely existed type of data, i.e., tensor field, has not been discussed. A possible solution can be decompose a tensor field into several scalar fields. We also can use the eigen values decomposed from the tensor field. Moreover, for specific applications, we can use some domain specific quantities to comprise the tensor fields. Then we can use the algorithm in Chapter 3 to find the similar patterns in multiple scalar fields.

Another potential future work is to automatically find similar patterns in the data set without given a reference pattern. Given an unknown dataset, and without prior understanding of it, users have often difficulties with finding an appropriate reference pattern. It is better for users to find all the similar patterns by only indicating some quantities of the desired reference, e.g., size, extent, and density. Related applications have been discussed in computer vision such as [18, 44]. They aim to find robust feature matchings in two sets of features. We can apply their methods into the algorithm introduced in Chapter 3, and group these feature into clusters based on their matchings. Then, we can further analyze these clusters, and show the group of clusters which are similar to each other.

For shape recovery problem, in the real data experiment, to prevent errors, we have asked our subjects to close their eyes and mouth. To relax this constraint, a reliable face contour tracker could help. By restricting the movements of eyes and mouth to tracked contours, many complicated expressions could be captured. Moreover, our camera set-up is currently only capable of covering the frontal face. However, it can be expected that adding more cameras allows fitting a complete 3D head template with the same approach. In future work, we would like to apply this technique to a large database of subjects to build a morphable face model that features very high resolution geometry.

Bibliography

[1] O. Alexander, M. Rogers, W. Lambeth, M. Chiang, and P. Debevec. Creating a photoreal digital actor: The digital emily project. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 69–80, 2009. 9, 75, 76

[2] B. Amberg, S. Romdhani, and T. Vetter. Optimal step nonrigid ICP algorithms for surface registration. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2007. 76

[3] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):698–700, 1987. 41, 65

[4] S. Battiato, G. Gallo, G. Puglisi, and S. Scellato. Sift features tracking for video stabilization. In *Image Analysis and Processing, 2007. ICIAP 2007. 14th International Conference on*, pages 825–830. IEEE, 2007. 7

[5] V. Blanz, C. Basso, T. Poggio, and T. Vetter. Reanimating faces in images and video. *Comput. Graph. Forum (Proc. Eurographics)*, 22(3):641–650, 2003. 9

[6] V. Blanz and T. Vetter. A morphable model for the synthesis of 3d faces. In *Proc. SIGGRAPH*, pages 187–194, 1999. 9, 75, 76

[7] S. Born, M. Pfeifle, M. Markl, M. Gutberlet, and G. Scheuermann. Visual analysis of cardiac 4d mri blood flow using line predicates. *IEEE Transactions on Visualization and Computer Graphics*, 19:900–912, 2013. 8

[8] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International journal of computer vision*, 74(1):59–73, 2007. 7

[9] A. Brun, H. Knutsson, H.-J. Park, M. E. Shenton, and C.-F. Westin. Clustering fiber traces using normalized cuts. In *MICCAI*, pages 368–375. Springer, 2004. 72

[10] R. Bujack, I. Hotz, G. Scheuermann, and E.Hitzer. Moment invariants for 2d flow fields using normalization. In *Proc. IEEE Pacific Visualization 2014*, 2014. 69, 72

[11] R. Bujack, I. Hotz, G. Scheuermann, and E. Hitzer. Moment invariants for 2d flow fields using normalization. In *Pacific Visualization Symposium (PacificVis), 2014 IEEE*, pages 41–48. IEEE, 2014. 8, 11

[12] R. Bujack, J. Kasten, I. Hotz, G. Scheuermann, and E. Hitzer. Moment Invariants for 3D Flow Fields Using Normalization. In *IEEE Pacific Visualization Symposium, PacificVis 2015 in Hangzhou, China*, 2015. 8

[13] S. Camarri, M.-V. Salvetti, M. Buffoni, and A. Iollo. Simulation of the three-dimensional flow around a square cylinder between parallel walls at moderate Reynolds numbers. In *XVII Congresso di Meccanica Teorica ed Applicata*, 2005. 36

[14] S. Candelaresi and A. Brandenburg. Decay of helical and nonhelical magnetic knots. *Phys. Rev. E*, 84(1):16406–16416, 2011. 32

[15] H. Carr and D. Duke. Joint contour nets. *Visualization and Computer Graphics, IEEE Transactions on*, 20(8):1100–1113, 2014. 8

[16] W. Chen, S. Zhang, S. Correia, and D. S. Ebert. Abstractive representation and exploration of hierarchically clustered diffusion tensor fiber tracts. *Comput. Graph. Forum (Proc. EuroVis)*, 27(3):1071–1078, 2008. 73

[17] W. Cheung and G. Hamarneh. n-sift: n-dimensional scale invariant feature transform for matching medical images. In *In Proceedings of the Fourth IEEE International Symposium on Biomedical Imaging: From Nano to Macro, 2007 (ISBI 2007*, pages 720–723, 2007. 5, 7, 11, 12, 15, 16, 30, 31, 98

[18] M. Cho and K. M. Lee. Progressive graph matching: Making a move of graphs via probabilistic voting. In *CVPR*, pages 398–405. IEEE Computer Society, 2012. 88

[19] B. De Decker, J. Kautz, T. Mertens, and P. Bekaert. Capturing multiple illumination conditions using time and color multiplexing. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 2536–2543, 2009. 9

[20] D. Decarlo and D. Metaxas. Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision*, 38:99–127, 2000. 9

[21] M. Dekker. *Mathematical Programming*. CRC, 1986. 80

[22] Z. Deng, P.-Y. Chiang, P. Fox, and U. Neumann. Animating blendshape faces by cross-mapping motion capture data. In *Proc. Interactive 3D Graphics and Games*, pages 43–48, 2006. 75

[23] M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice–Hall, 1976. 41

[24] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *Proc. VisSym 03*, pages 239–248, 2003. 8

[25] J. Ebling and G. Scheuermann. Clifford convolution and pattern matching on vector fields. In *Proc. IEEE Visualization*, pages 193–200, 2003. 7, 11, 22, 23, 24, 28, 98

[26] J. Ebling and G. Scheuermann. Clifford fourier transform on vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):469–479, July 2005. 7

[27] J. Ebling and G. Scheuermann. Segmentation of flow fields using pattern matching. In *Proc. EuroVis*, pages 147–154, 2006. 7, 69, 72

[28] J. Flusser and T. Suk. Rotation moment invariants for recognition of symmetric objects. *IEEE Transactions on Image Processing*, 15(12):3784 – 3790, 2006. 11

[29] G. Fyffe, T. Hawkins, C. Watts, W.-C. Ma, and P. Debevec. Comprehensive facial performance capture. *Computer Graphics Forum (Proc. Eurographics)*, 30:425–434, Apr. 2011. 9

[30] D. M. Gavrila. A bayesian, exemplar-based approach to hierarchical shape matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1408–1421, 2007. 65

[31] M. Griebel, T. Dornseifer, and T. Neunhoeffer. Numerical simulation in fluid dynamics, a practical introduction. In *SIAM*, 1998. 22

[32] D. Günther, A. Jacobson, J. Reininghaus, H.-P. Seidel, O. Sorkine-Hornung, and T. Weinkauf. Fast and memory-efficient topological denoising of 2D and 3D scalar fields. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE VIS)*, 20(12):2585–2594, December 2014. 8

[33] T. Günther, C. Rössl, and H. Theisel. Opacity optimization for 3d line fields. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 32(4):120:1–120:8, 2013. 57

[34] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part IV*, ECCV'12, pages 459–472, Berlin, Heidelberg, 2012. Springer-Verlag. 11

[35] E. Heiberg, T. Ebbers, L. Wigstrom, and M. Karlsson. Three dimensional flow characterization using vector pattern matching. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):313–319, 2003. 7, 11, 22, 23, 24, 28, 69, 72, 98

[36] B. K. Horn, H. M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *JOSA A*, 5(7):1127–1135, 1988. 64

[37] H. Huang, J. Chai, X. Tong, and H.-T. Wu. Leveraging motion capture and 3d scanning for high-fidelity facial performance acquisition. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 30:74:1–74:10, 2011. 9

[38] L. Huettenberger, C. Heine, H. Carr, G. Scheuermann, and C. Garth. Towards multifield scalar topology based on pareto optimality. *Computer Graphics Forum*, 32(3pt3):341–350, 2013. 8

[39] J. Hunt. Vorticity and vortex dynamics in complex turbulent flows. *Proc CANCAM, Trans. Can. Soc. Mec. Engrs*, 11:21, 1987. 17

[40] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998. 42

[41] J. Jeong and F. Hussain. On the identification of a vortex. *J. Fluid Mechanics*, 285:69–94, 1995. 17

[42] R. Jianu, C. Demiralp, and D. H. Laidlaw. Exploring 3d dti fiber tracts with linked 2d representations. *Transactions on Visualization and Computer Graphics*, 15(6):1449–1456, 2009. 8, 73

[43] J. J. Kuffner. Effective sampling and distance metrics for 3d rigid body path planning. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 4, pages 3993–3998. IEEE, 2004. 21

[44] J. Lee, M. Cho, and K. M. Lee. Hyper-graph matching via reweighted random walks. In *CVPR*, pages 1633–1640. IEEE Computer Society, 2011. 88

[45] Y. Li, C. Wang, and C. Shene. Streamline similarity analysis using bag-of-features. In *Proc. SPIE*, volume 9017, pages 90170N–90170N–12, 2013. 8

[46] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society. 7

[47] D. G. Lowe. Local feature view clustering for 3d object recognition. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–682. IEEE, 2001. 7

[48] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004. 7, 11, 12, 15, 29

[49] K. Lu, A. Chaudhuri, T. Lee, H. Shen, and P. C. Wong. Exploring vector fields with distribution-based streamline analysis. In *IEEE Pacific Visualization 2013*, 2013. 8, 72

[50] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *Proceedings of the 33rd international conference on Very large data bases*, pages 950–961. VLDB Endowment, 2007. 42

[51] W.-C. Ma, T. Hawkins, P. Peers, C.-F. Chabert, M. Weiss, and P. Debevec. Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination. In *Proc. Eurographics Symposium on Rendering Techniques*, pages 183–194, June 2007. 9

[52] W.-C. Ma, A. Jones, J.-Y. Chiang, T. Hawkins, S. Frederiksen, P. Peers, M. Vukovic, M. Ouhyoung, and P. Debevec. Facial performance synthesis using deformation-driven polynomial displacement maps. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 27:121:1–121:10, 2008. 9

[53] T. Malzbender, B. Wilburn, D. Gelb, and B. Ambrisco. Surface enhancement using real-time photometric stereo and reflectance transformation. In *Proc. Eurographics Symposium on Rendering Techniques*, pages 245–250, 2006. 9

[54] T. McLoughlin, M. W. Jones, R. S. Laramee, R. Malki, I. Masters, and C. D. Hansen. Similarity measures for enhancing interactive streamline seeding. *IEEE Transactions on Visualization and Computer Graphics*, 19(8):1342–1353, 2013. 8, 57, 72

[55] T. McLoughlin, R. S. Laramee, R. Peikert, F. H. Post, and M. Chen. Over two decades of integration-based, geometric flow visualization. *Computer Graphics Forum*, 29(6):1807–1829, 2010. 55, 62

[56] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *In VISAPP International Conference on Computer Vision Theory and Applications*, pages 331–340, 2009. 21

[57] D. Nehab, S. Rusinkiewicz, J. Davis, and R. Ramamoorthi. Efficiently combining positions and normals for precise 3d geometry. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 24:536–543, July 2005. 9

[58] L. Paulevé, H. Jégou, and L. Amsaleg. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognition Letters*, 31(11):1348–1358, 2010. 21

[59] O. Pele and M. Werman. Fast and robust earth mover's distances. In *Proc. ICCV*. IEEE, 2009. 61

[60] F. Pighin, R. Szeliski, and D. H. Salesin. Resynthesizing facial animation through 3d model-based tracking. In *Proc. IEEE International Conference on Computer Vision*, volume 1, pages 143–150, 1999. 9

[61] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007. 62

[62] C. Rössl and H. Theisel. Streamline embedding for 3d vector field exploration. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):407–420, 2012. 8, 62, 72

[63] H. Saikia, H.-P. Seidel, and T. Weinkauf. Extended branch decomposition graphs: Structural comparison of scalar data. *Computer Graphics Forum (Proc. EuroVis)*, 33(3):41–50, June 2014. 8

[64] H. Saikia, H.-P. Seidel, and T. Weinkauf. Fast similarity search in scalar fields using merging histograms. In H. Carr, C. Garth, and T. Weinkauf, editors, *TopoInVis*, pages 1–14, Annweiler, Germany, May 2015. 8

[65] T. Salzbrunn, C. Garth, G. Scheuermann, and J. Meyer. Pathline predicates and unsteady flow structures. *The Visual Computer*, 24(12):1039–1051, 2008. 8

[66] T. Salzbrunn and G. Scheuermann. Streamline predicates. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1601–1612, 2006. 8

[67] N. Sauber, H. Theisel, and H.-P. Seidel. Multifield-graphs: An approach to visualizing correlations in multifield scalar data. *IEEE Transactions on Visualization and Computer Graphics*, 12:917–924, 2006. 8, 19

[68] M. Schlemmer, M. Heringer, F. Morr, I. Hotz, M. Hering-Bertram, C. Garth, W. Kollmann, B. Hamann, and H. Hagen. Moment invariants for the analysis of 2d flow fields. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1743–1750, 2007. 8, 69, 72

[69] M. Schulze, J. M. Esturo, T. Günther, C. Rössl, H.-P. Seidel, T. Weinkauf, and H. Theisel. Sets of globally optimal stream surfaces for flow visualization. *Computer Graphics Forum (Proc. EuroVis)*, 33(3):1–10, June 2014. 8

[70] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th International Conference on Multimedia*, MULTIMEDIA '07, pages 357–360, New York, NY, USA, 2007. ACM. 5, 7, 11, 12, 15, 16, 30, 31, 98

[71] O. Sorkine, D. C. Or, Y. Lipman, M. Alexa, C. Rossl, and H. P. Seidel. Laplacian surface editing. In *Proc. 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, 2004. 81

[72] D. Stalling and T. Steinke. Visualization of vector fields in quantum chemistry. Technical report, ZIB Preprint SC-96-01, 1996. 69

[73] J. Tao, C. Wang, and C. Shene. Flowstring: Partial streamline matching using shape invariant similarity measure for exploratory flow visualization. In *Proc. IEEE Pacific Visualization*, March 2014. 8, 72

[74] J. R. Tena, F. D. la Torre, and I. Matthews. Interactive region-based linear 3d face models. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 30:76:1–76:10, July 2011. 9

[75] H. Theisel and H.-P. Seidel. Feature flow fields. In *Data Visualization 2003. Proc. VisSym 03*, pages 141–148, 2003. 8

[76] D. Thomas and V. Natarajan. Multiscale symmetry detection in scalar fields by clustering contours. *Visualization and Computer Graphics, IEEE Transactions on*, 20(12):2427–2436, Dec 2014. 8

[77] D. M. Thomas and V. Natarajan. Symmetry in scalar field topology. *IEEE TVCG*, 17(12):2035–2044, 2011. 8

[78] R. Tsai. An efficient and accurate camera calibration technique for 3d machine vision. In *Proc. IEEE Computer Vision and Pattern Recognition*, pages 364–374, 1986. 78

[79] D. Vlasic, M. Brand, H. Pfister, and J. Popović. Face transfer with multilinear models. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 24:426–433, 2005. 9

[80] Z. Wang, J. M. Esturo, H.-P. Seidel, and T. Weinkauf. Pattern search in flows based on similarity of stream line segments. In *Proc. Vision, Modeling and Visualization*, 2014. 11

[81] J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963. 61

[82] J. Wei, C. Wang, H. Yu, and K. Ma. A sketch-based interface for classifying and visualizing vector fields. In *Proc. IEEE Pacific Visualization*, pages 129–136, 2010. 8, 72

[83] T. Weinkauf, Y. Gingold, and O. Sorkine. Topology-based smoothing of 2D scalar fields with c1-continuity. *Computer Graphics Forum (Proc. EuroVis)*, 29(3):1221–1230, June 2010. 8

[84] T. Weinkauf, H.-C. Hege, and H. Theisel. Advected tangent curves: A general scheme for characteristic curves of flow fields. *Computer Graphics Forum (Proc. Eurographics)*, 31(2):825–834, April 2012. Eurographics 2012, Cagliari, Italy, May 13 - 18. 36, 73

[85] T. Weinkauf and H. Theisel. Curvature measures of 3D vector fields and their applications. *Journal of WSCG*, 10(2):507–514, 2002. 19, 62

[86] T. Weinkauf and H. Theisel. Streak lines as tangent curves of a derived vector field. *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization 2010)*, 16(6):1225–1234, November - December 2010. Received the Vis 2010 Best Paper Award. 73

[87] T. Weinkauf, H. Theisel, A. V. Gelder, and A. Pang. Stable feature flow fields. *IEEE Transactions on Visualization and Computer Graphics*, 2010. accepted. 8

[88] T. Weinkauf, H. Theisel, and O. Sorkine. Cusps of characteristic curves and intersection-aware visualization of path and streak lines. In R. Peikert, H. Hauser, H. Carr, and R. Fuchs, editors, *Topological Methods in Data Analysis and Visualization II*, Mathematics and Visualization, pages 161–176. Springer, 2012. 73

[89] T. Weise, S. Bouaziz, H. Li, and M. Pauly. Realtime performance-based facial animation. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 30:77:1–77:10, July 2011. 9

[90] C. A. Wilson, A. Ghosh, P. Peers, J.-Y. Chiang, J. Busch, and P. Debevec. Temporal upsampling of performance geometry using photometric alignment. *ACM Transactions on Graphics*, 29:17:1–17:11, Apr. 2010. 9, 78

[91] R. J. Woodham. Shape from shading. chapter Photometric method for determining surface orientation from multiple images, pages 513–531. MIT Press, Cambridge, MA, USA, 1989. 9

[92] Z. Yi, C. Zhiguo, and X. Yang. Multi-spectral remote image registration based on sift. *Electronics Letters*, 44(2):107–108, 2008. 7

[93] M. Zöckler, D. Stalling, and H. Hege. Interactive visualization of 3D-vector fields using illuminated stream lines. In *Proc. IEEE Visualization*, pages 107–113, 1996. 62

List of Figures

List of Tables