INFORMATION VISUALIZATION OF LARGE DATA STREAMING

CHENHUI LI

Ph.D

The Hong Kong Polytechnic University

2018

The Hong Kong Polytechnic University

Department of Computing

Information Visualization of Large Data Streaming

Chenhui Li

A Thesis Submitted
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy

May 2017

CERTIFICATE OF ORIGINALITY

I hereby declare that this thesis is my own work and that, to the best of my knowledge and belief, it reproduces no material previously published or written, nor material that has been accepted for the award of any other degree or diploma, except where due acknowledgement has been made in the text.

_____ (Signed)

Chenhui Li (Name of student)

ABSTRACT

The visualization of increasingly large streaming data has become a challenge for traditional static visualization algorithms and in the cognitive process of understanding features at various scales of resolution. In this thesis, we focus on stream visualization and study a class of visual analytic techniques that provide rich visual patterns to help common users better comprehend the main features of stream data. In stream visualization, we look for significant temporal patterns via several classical methodologies in the modern context of rapidly changing information content. However, the current methodologies for stream visualization are still limited. We summarize four critical problems for which the currently appropriate solutions require further improvements: (1) How to visually cluster useful patterns from streaming data; (2) How to smoothly map between data frames to provide a continuous visual effect in dynamic visualization and detect smooth trends and patterns; (3) How to automatically retarget the significant content of streams and make it suitable for different resolutions; and (4) How to visually query the patterns among streaming data

To answer the first question, we present two approaches to support the stream-based visualization and visual analysis. One approach is a density map estimation method to accurately cluster the high-density data. Another approach is a module-based clustering method to detect graph patterns and emphasize the interconnecting structures between local modules. Since the first problem is fundamental, the answer to it provides the foundation for answering the other three questions. For the second question, we propose a novel algorithm called StreamMap that utilizes the diffusion model to smoothly morph frames among dynamic data. We also present a trend representation that can help convey the flow directions. The approach for the third question is a mesh-based energy optimization method. We propose a visual-saliency map to mark the regions with different significances. Auxiliary triangles are used to retarget the elements in visualizations, and the optimized result is achieved by solving a large sparse linear system. To answer the fourth question, we outline a

new framework with two interactions for interactively querying the streaming data. We also designed a pair of representations to visualize and explore the query results. The effectiveness of the presented methods are demonstrated on several real datasets when dynamic visualization and a visual analysis of structured and unstructured patterns in streaming data are required.

LIST OF PUBLICATIONS

1. **Chenhui Li**, George Baciu, and Yu Han. StreamMap: Smooth Dynamic Visualization of High-Density Streaming Points. IEEE Transactions on Visualization and Computer Graphics, 2017. (Accepted)

2. **Chenhui Li**, George Baciu, Yunzhe Wang, and Xiujun Zhang. Fast Content-Aware Resizing of Multi-layer Information Visualization via Adaptive Triangulation. Journal of Visual Languages and Computing, 2017. (Accepted)

3. **Chenhui Li**, George Baciu, and Yunzhe Wang. Module-Based Visualization of Large-Scale Graph Network Data. Journal of Visualization, 2017, 20(2), pp.205-215.

4. **Chenhui Li** and George Baciu. VisQuery: Visual Querying of Streaming Data via Pattern Matching. IEEE Digital Media Industry Academic Forum, Santorini, Greece, July 4-6, 2016, pp.161-165.

5. **Chenhui Li**, George Baciu, and Yunzhe Wang. ModulGraph: Modularity-Based Visualization of Massive Graphs. SIGGRAPH Asia 2015 Symposium On Visualization in High Performance Computing, Kobe, Japan, 2015, pp.11:1-11:4.

6. **Chenhui Li**, George Baciu, and Yu Han. Interactive Visualization of High Density Streaming Points with Heat-Map. Proceeding of IEEE International Conference on Smart Computing, Hong Kong, 2014, pp.145-149.

7. **Chenhui Li** and George Baciu. VALID: A Web Framework for Visual Analytics of Large Streaming Data. IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2014), Beijing, China, 2014, pp.686-692.

8. George Baciu, **Chenhui Li**, Yunzhe Wang, and Xiujun Zhang. A Cloud-Driven Visual Cognition of Large Streaming Data. International Journal of Cognitive

Informatics and Natural Intelligence, 2016, 10(1), pp.12-31.

9. George Baciu, **Chenhui Li**, Yunzhe Wang, and Xiujun Zhang. Cloudets: Cloud-Based Cognition for Large Streaming Data (**Best Paper Award**). 14th IEEE International Conference on Cognitive Informatics and Cognitive Computing (ICCI*CC 2015), Beijing, China, 2015, pp.333-338.

10. George Baciu, Yunzhe Wang, and **Chenhui Li**. Smooth Animation of Structure Evolution in Time-Varying Graphs with Pattern Matching. SIGGRAPH Asia 2017 Symposium On Visualization, Bangkok, Thailand, 2017.

11. George Baciu, Yunzhe Wang, and **Chenhui Li**. Cognitive Visual Analytics of Multi-dimensional Cloud System Monitoring Data. International Journal of Software Science and Computational Intelligence, 2017, 9(1), pp.20-34.

ACKNOWLEDGEMENTS

I would never have completed this research study without the help from many people.

Firstly, I thank my supervisor, Prof. George Baciu, for his years of encouragement and mentoring. Prof. George Baciu always gives me the strongest support and insightful suggestions on my Ph.D. courses, idea finding, experiments, research paper writings, and presentations. Either discussing with us or reviewing the research manuscript, he is full of energy and patient.

I thank all Lab members and friends at PolyU – Dr Yu Han, Miss Yunzhe Wang, Dr Jianlin Liu, Dr Yunliang Cai, Dr Xiujun Zhang, Mr Yushi Li, Miss Zhixiang He, and many others.

Finally, I thank my parents for their selfless support and care. Without their understanding and help, it would have been impossible to finish this dissertation. In addition, I want to express my sincere gratitude to my wife for her long-term encouragement, understanding, and a great patient.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Information visualization can be defined as any mental representation for communicating a message via charts, plots, tables, graphs, animation, or any combinations of visual shapes, symbols, or glyphs that form patterns consistent with the underlying information being represented and transmitted. Early foundational issues in information visualization field were addressed by Bertin [18]. He presented a visualization theory through a study of graphic techniques. A number of current visualization designs follow the ideas outlined by Bertin [18]. In the big data era, large-scale information visualization has become the main challenge in computational analytics for various applications, ranging from search and rescue (e.g., Malaysia Airlines Flight MH370) to medical diagnosis, on-line retail, internet of things, network security, and in big data science (e.g., cosmology, high-energy physics, genomics), as addressed in many other publications [86, 90, 153].

Research on streaming data visualization is becoming more significant with the increasing volume of time-varying data from real-time data collectors and monitors. Data streams offer new opportunities for the development of tools that can be used to observe and explore temporal data from various perspectives, thus helping data analyzers to visually explore the historical and predicted data leading to quicker and smarter decisions.

Streaming data contains a vast amount of time-varying, unstructured information. Hence, it is becoming increasingly difficult to analyze, visualize, explore, and interpret the data patterns in the data streams. Although displaying raw streaming data can help in understanding relevant patterns, the volume, variety, velocity, and variability of the streams preclude the deep interpretation of the data patterns. For example, simply displaying two-dimensional raw streaming data at different time

steps, as shown in Fig. 1.1, does not properly show the variation information among streaming frames. In order to cope with these problems in big data analytics and identify new general directions in visualiation research, in this thesis we propose a series of approaches toward a better information visualization through large-scale data streaming.

In this chapter, the motivation of this thesis is first addressed, overcoming the overlapping of high-density data visualization, smoothing the sequence-by-sequence representation, and discussing the aggregation requirement from large-scale data visualization, the difficulties encountered in content retargeting and enhancement, and the visual querying on spatio-temporal data. Moreover, the contributions of this thesis are discussed. The chapter closes with a brief overview of the thesis structure.

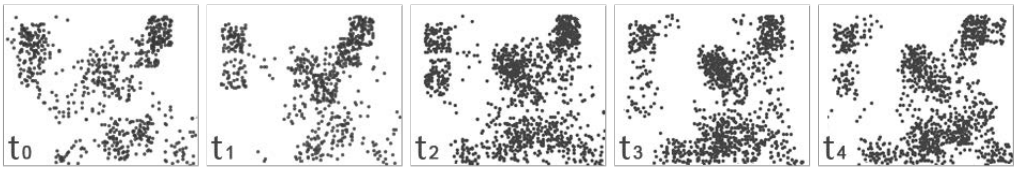


Figure 1.1: Two-dimensional raw streaming data at different time steps.

1.1 Motivation

Traditional information visualization algorithms and tools mainly focus on static data visualization. Static data refers to data that does not include a temporal dimension. Since most useful data expands in real time, such as data from climate monitoring, social networks, news, surveillance, and tracking, it is becoming increasingly important to prioritize, filter, cluster, enhance, and query relevant information in data streams before pursuing higher levels of visual analytics. The extension from static data visualization to stream visualization is not trivial since stream visualization is a time-dependent process that poses various problems. In addition, a coherent method has not been presented to solve different forms of stream visualization problems. It motivates us to present a coherent approach that is the foundation of the other methods such as smoothing, content-aware resizing, and visual querying.

Overlapping in high-density Data

Information visualization suffers from overlapping when the data stream contains high-density data structures. This problem is known as overdrawing, and it has become more significant as the size of data has exploded. Although the existing approaches achieve high performance and acceptable visual results, there are still a few limitations in the visualization of important patterns and structures from large volumes of continuous data, such as streaming point data (Fig. 1.2) and graph network data (Fig. 1.3).

For streaming point data, *kernel density estimation* (KDE) is one of the statistical methods often used to overcome the overlapping problem. However, KDE requires manual bandwidth adjustment to estimate the density of streaming points. Therefore, it is necessary to improve KDE to calculate the density of streaming data by using the adaptive bandwidth approach. The adaptive bandwidth approach offers two advantages. First, it improves the visual disparity in cluttered point clouds because sparse points are estimated using small kernel sizes, thus avoiding abrupt discontinuous changes and reducing the overall number of calculations. Second, it also avoids the need for manual bandwidth adjustment for different point sets. For static data visualization, manual adjustment may work well; however, for stream visualization, adaptive bandwidth adjustment is both convenient and necessary because of the number of sequential point sets that need to be estimated.

(a) (b)

Figure 1.2: Two frames of high-density geographical data.

(a) (b)

Figure 1.3: Two large networks at different time steps.

Analysis and visualization of the large-scale network have become important challenges with the rapid increase in network data, including the data generated from social networks, modern transportation networks, document referencing, and academic citations. A practical approach to displaying graph content is partitioning the network according to well-defined domain-dependent attributes. However, graph visualization in the presence of incomplete information is an open challenge, and applications in this area can be found in abundance. To better visualize and understand the patterns in large-graph discovery, a representation of local patterns is required to avoid overlapping in high-density network data. This is a critical step in determining the structural components of a graph visualization, particularly when the graph network data are dynamic.

Abrupt changes in animation-based visualization

Directly visualizing data streams as dynamic frames without interpolation leads to a significant problem of abrupt changes because visual continuity is missing between two data frames. A dynamic morphing (dynamic blending) approach between frames is often adopted to produce patterns that are easier to observe and evaluate in streaming data visualization. For dynamic morphing, conventional linear interpolation between two frames is a practical solution. However, when the data streams contain large variations, this approach produces visual ghosting patterns. In addition, the

information generated by linear interpolation does not always produce visually acceptable trend patterns, especially in point cloud regions.

The advantages of smooth dynamic data visualization with visual continuity can be summarized as follows. First, the human visual system is well adapted to identify changes in the shapes of dynamic regions compared with coarse, non-smooth visualizations of dynamic data. This advantage was reported by Tversky et al. [135]. Second, as the size of data increases exponentially, it becomes increasingly difficult to show sufficient information in a single image frame. Although some techniques such as binning and summarization [153] perform re-sampling and data reduction for the target display, the number of pixels for a static image will always remain finite, and the points from data streams can easily exceed the display capacity. Third, morphing operations produce intermediate patterns compared to static visualizations. Moreover, these additional patterns can include data trends, as presented by Thirion [129]. Therefore, it is vital to find an effective algorithm to retarget one frame to another frame to avoid abrupt changes that are uncomfortable for the human eye system.

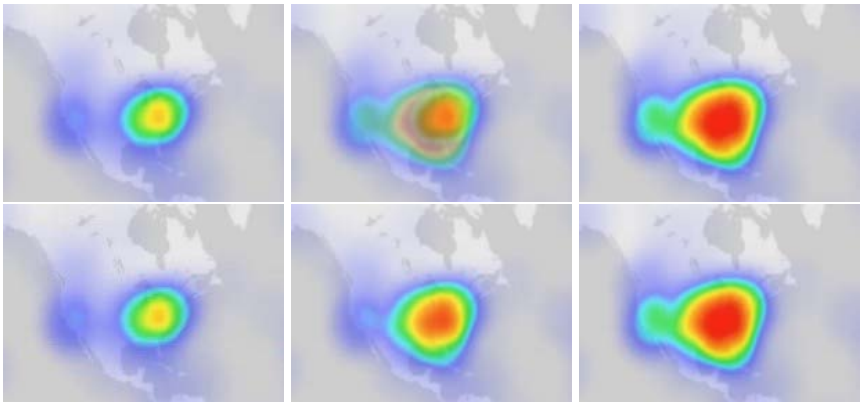


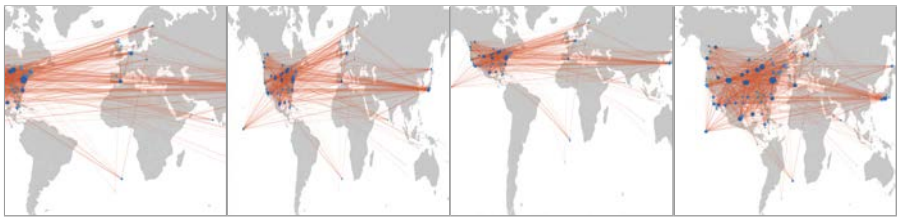
Figure 1.4: Comparison of linear blending (upper) and smooth morphing with minimal artifacts (lower). Left and right: input density maps. Middle: transition sequence.

Content-aware resizing requirement

Content resizing is required when the stream representation should effectively adapt for different ratios of displays, such as mobile phones, tablet pads, projector screens,

and widescreen televisions. The basic content resizing techniques, such as cropping and linear scaling, often result in losing information content and introducing distortions. Cropping (Fig. 1.5b) is the simplest operation for visualization resizing. Cropping can be used to adapt to different types of displays, but it often removes important content. Linear scaling (Fig. 1.5c) is another approach, but distortions often appear when more important content has the same scaling rate as less important regions. Missing content and introduced distortions in visualizations quickly lead to a loss of attention or, worse, the complete misinterpretation of the information presented. Hence, it is paramount to adopt a robust approach for information visualization that not only resizes the content appropriately but also retains the important content.

(a) Original visualization.



(b) Cropping method. (c) Scaling method. (d) Grid-based [145]. (e) Expected result.

Figure 1.5: Results of four different strategies for the resizing of visualization outputs.

In addition, for the stream representation, the temporal coherency of the stream frames should be considered to avoid artifacts. Similar methods have been proposed in the works of seam carving for video retargeting [117] and motion-based video re-

targeting [144]. However, the motion-aware resizing technique has not been studied for information visualization, especially for dynamic information visualization.

Visual stream querying

Interactive querying of streaming data is becoming an increasingly important function in visual analytics applications. However, the conditional selection for querying spatio-temporal data is currently an open challenge. Relational queries require a new visualization paradigm to make use of the native data properties and features. Simply listing the query results as a histogram cannot fulfill the spatio-temporal querying requirements because of missing time sequences and feature details. In addition, a static histogram cannot represent related potential information, such as variations and evolutions. Therefore, a practical approach for retrieving spatio-temporal data is required to provide a more intuitive querying interaction to better identify regions of interest.

1.2 Contributions

Our visual clustering work is proposed in [77] and later improved upon in [79]. The study of smooth dynamic visualization is presented in [75] and [76]. Content-aware resizing work is addressed in [80]. Visual querying work is proposed in [78]. The data processing frameworks for visualization are published in [74], [13], and [14].

Visual clustering methods for large-scale data representation

Aiming at two basic data types, points and graphs, we propose two visual clustering methods to overcome the overdrawing problem in high-density data. First, an adaptive kernel density estimation method is developed to aggregate high-density data at a time step as a density map. The adaptive kernel density estimation method avoids the trivial manual operations in the data stream visualization. Second, a flexible graph visualization framework that aggregates the components of large graphs into

modules helps users overcome graph size limitations and effectively gain insights into attributed graph data. The main idea is to partition the graph into several clusters and visualize the relationships between clusters to emphasize the graph patterns and allow users to understand the structure of a large graph in detail. The study of graph clustering also supports further dynamic graph visualization. Most of the proposed approaches adopt visual clustering results as input, so the visual clustering is the foundation of the other methods such as smoothing, content-aware resizing, and visual querying.

Dynamic data smoothing

We show an approach, entitled StreamMap, to effectively avoid abrupt changes in the data stream visualization and help in revealing the key patterns needed to understand the information contained in streams. We define the task of streaming data visualization as a problem in creating a smooth interpolation between a pair of frames that contain two sets of data during a given time interval. A novel algorithm for smoothing is presented through density map morphing. In addition, the variation patterns among in-between sequences are visualized to gain insights into data trends. The experiments demonstrate the effectiveness of StreamMap when dynamic visualization and the visual analysis of trend patterns in streaming data are required.

Content-aware resizing for information visualization

We present a content-aware resizing approach for information visualization. The basic idea is to resize the visualization via the adaptive adjustment of a superimposed reference mesh for each detected layer. The contributions of this work are fourfold. First, an abstract multi-layer model for the resizing problem of information visualization. Our model can be used to resize the output from a visualization system to automatically match the native aspect ratio of any external target display. Second, a set of criteria called the *visual saliency map* (VSM) to describe the features of information visualizations in different saliency layers. Third, a triangle mesh-based

energy optimization method to achieve a better visual distribution of information after resizing. Fourth, the resizing model has been extended to re-target the dynamic information visualization.

Visual querying of temporal data

We present a saliency-based interactive querying framework, called SalQuery, to address the requirements of visual interactions and temporal data analysis. SalQuery can aid users in obtaining insights into the pattern evolution and similarity according to the selected regions. The contributions are three-fold. First, we construct a density estimation kernel that transforms the raw two-dimensional temporal data into a fixed-size saliency map and encodes the saliency region information as saliency blocks. We present a novel saliency block structure to encode the features of a saliency region. Second, a pair of interactions is presented to make the querying process convenient. A robust saliency matching method has been addressed to ensure the querying speed and accuracy on a large-scale, spatio-temporal dataset. Third, a hybrid visualization method is designed to enrich the query results from two perspectives: flow-oriented and similarity-oriented perspectives.

1.3 Organization

The remainder of the thesis is organized as follows. Chapter 2 outlines the prior related work. Chapter 3 introduces a novel smooth dynamic visualization method. Chapter 4 describes a module-based large-scale graph data visualization method. Chapter 5 addresses a resizing method for information visualization. Chapter 6 introduces a visual querying framework for the exploration of streaming data. Chapter 7 concludes the thesis and discusses future work.

CHAPTER 2

LITERATURE REVIEW

Stream-based visualization has become one of the main research topics in IEEE and ACM publications, and it has been a primary field of study in information visualization, computer graphics, and data mining. It requires several techniques such as stream processing, data clustering, stream smoothing, content enhancement, and querying. In this chapter, the prior work on the information visualization of streaming data is reviewed. First, the related work on large-scale visualization frameworks is discussed. When the data are large-scale, a robust framework is required to effectively process the data. Second, the visual clustering methods are reviewed. Visual clustering could be considered a fundamental research topic for large-scale information visualization because of the space limitations of the display. The main objectives of the visual clustering are overcoming the overdrawing problem in high-density data visualization. Third, three related research topics on information visualization are addressed: dynamic data representation, information visualization resizing, and visual querying.

2.1 Large-Scale Information Visualization Frameworks

Recently, many frameworks have been presented on large-scale data visualization, as shown in Fig. 2.1. imMens [90] uses multivariate data tiles to process data loading in advance to support effective interaction. By using a GPU to accelerate querying computing, it sustains real-time brushing and linking of diverse visualization techniques. Lins [86] and his colleagues presented an algorithm to calculate and query nanocubes in streaming data. They demonstrated that nanocubes can be used with real-world large datasets via visual heat maps, histograms, and parallel coordinate

plots. Wickham [153] addressed a statistics-based framework for big data visualization. He introduced a simple but highly effective framework called *bin-summarise-smooth* to reduce the amount of data to still discover meaningful hidden patterns in large datasets. Zinsmaier [178] inspired by the technique of Level-of-Detail in the computer graphics discipline introduced straight-line graph drawing that can be rendered interactively with the level of detail needed to visualize large-scale contents. Although these frameworks have been shown to work well on static datasets, few frameworks are equipped for dynamic data.

(a) imMens [90]. (b) Nanocubes [86].

Figure 2.1: Two large-scale information visualization frameworks.

Dynamic data usually occur in data streams with relatively short time stamps. Sensors, for example, are naturally suited for delivering data streams. However, formatting large data sets for streaming and handling data streams for large data is a relatively new topic in database management [10] and information visualization [35, 155]. Specifically, in information visualization, the visualization of streaming data has become one of the most challenging problems.

Babcock et al. [10] gave a comprehensive overview of data stream systems. They outlined that data streams could not directly support persistent relations. Wong et al. [155] introduced a data stratification approach to speed up the processing of the data streams. Arasu et al. [8] presented a complex stream system that deals with well the continuous unstructured data. Mortar [91] was proposed by Logothetis and Yocum to handle streaming queries across large distributed systems. Although Mortar is a stable system for data stream processing and querying, its scalability and

elasticity could be further improved. Spark [170] is a novel memory-based parallel computing architecture that is designed for stream processing, but it is difficult to configure and program for the usage of the information visualization. The company of Alooma [4] developed a real-time stream visualization system, as shown in Fig. 2.2, to present the statistics of a data flow. They also provide tools for searching and filtering data streams. However, further visualization of dynamic data is required to explore the potential patterns, while the majority of the prior work barely addresses the problems found in dynamic data streams.

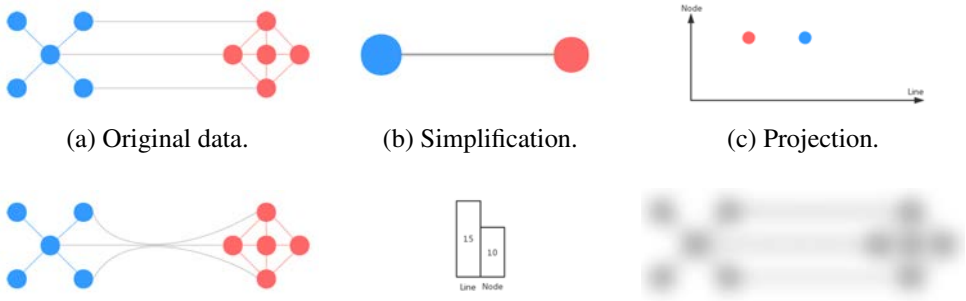
Figure 2.2: A real-time stream visualization system developed by Alooma [4].

2.2 Visual Clustering

Data clustering is a fundamental research topic in a variety of fields such as data mining, visual analytics, and image processing. K-means [94] is the most popular method to cluster data. Diverse variations and improvements have been presented to better cluster data such as DBSCAN [43], CLIQUE [3], Level Set [102], and Density Peaks [112].

Compared with the data clustering methods, visual clustering aims to overcome the overdrawing problem in high-density data visualizations. The study in [15] addressed the concern that overlapping data may lead to decision mistakes. Visual clustering can be considered a hybrid method that combines a data clustering method and a visualization technique and can be categorized into four main types: simpli-

fication, projection, bundling, and statistics. Figure 2.3 shows an example of these four visual clustering methods.



(a) Original data. (b) Simplification. (c) Projection.

(d) Bundling. (e) Statistics: histogram. (f) Statistics: KDE.

Figure 2.3: Visual clustering methods.

Simplification

Simplification is defined as a method that reduces the amount of original data. Aggregation, sampling, and summarization methods can be classified as simplification methods. The simplification method reduces the data amount according to their position distribution, time stamp, or other feature channels. The binning and summarization [153] method sampled data to overcome the overlapping problem and reduce the computational time. Chen et al. [29] proposed a visual abstraction and exploration system that samples the original data through blue noise calculation, which can represent multi-class data distributions. Cottam et al. [34] proposed a simple aggregation process that enables the concise expression of the alpha composition. When the data are from moving objects such as taxis, flights or animals, the sampling approaches presented by Andrienko et al. [7, 5] can be used to explore the locations of significant changes.

Various methods for data simplification have been developed to visualize large-scale data. The most prominent one is graph summarization [131], which allows the user to interactively control the resolution of each aggregation in a large graph. HiMap [123] was presented to visualize large-scale social networks through a hierarchical summarization. Complexity reduction via k-Core [37] was used to re-

move nodes with fewer than k links. Zhang and his colleagues [172] proposed a new model to calculate the core structure with a million or more nodes. Unlike the k-Core method, this model can convert low-degree vertices into core nodes.

The community detection algorithm is usually used to simplify large-scale data, especially network data. It is aimed at partitioning a large graph into communities. Newman [99] converted the community detection problem into an optimization problem using a modularity-based method titled *Modularity Classes*. This method can measure the performance of the algorithm on the community detection problem. Blondel et al. [19] presented an acceleration method called *Louvain* to reduce the computational complexity of the community detection problem and demonstrate its performance on several large datasets. Dynamic graph evolution visualization was achieved in [139, 146] through community detection.

Projection

Different from simplification, the projection method converts data from one space to another space. Keim et al. [64] presented a method called PixelMap that projects high-density points to surrounding empty regions to improve and smooth the visual effect. Another projection method in [46] considered readers' impressions by adjusting the aspect ratio. In addition, for high-dimensional data visualizations, a projection technique proposed by [97] represented the point data in 5D attribute space. A projection matrix and tree methods were proposed in [169] to provide insights into high-dimensional data. For graph visualization, a dimensionality reduction method was proposed in [136] to project abstract information into points in two-dimensional space.

Bundling

Edge bundling is a visual clustering method in which edges are organized into groups to achieve an uncluttered visualization layout. Many edge clustering methods have been presented from different perspectives, as summarized in the work of Zhou et

al. [176]. Holten [53] presented an early work on edge bundling. Later, force-directed edge bundling (FDEB) [54], a physical force-based approach, was developed to reduce edge clutter in large graphs. Following the emergence of FDEB, many edge-bundling methods arose, most of which were focused on increasing the speed and effectiveness of the bundling. For example, Telea and Ersoy [128] proposed an image-based approach (IBEB) for rendering a skeleton of bundled edges. Ersoy et al. [42] then extended the work on IBEB and presented a skeleton-based method of iteratively transforming edges toward the skeleton of the line set. Polygonal strips were presented by Palmas et al. [104] to guide the clustering of data in each dimension for line bundling. Hurter and his colleague [59] presented an image-based edge-bundling method for spreading control points while clustering edges. They were the first to implement edge bundling on a GPU. Later, a CUDA-based edge bundling framework, CUBu [137], was presented by Van et al. to speed up the visualization of large graphs.

Instead of pursuing speed improvements, Hurter et al. [58] performed edge bundling for time-varying data. Later, Bach et al. [11] presented a confluent drawing method for visualizing a graph by considering the network connectivity and information preservation. To further reduce the complexity of bundled edges, a module-based edge bundling method was proposed by Dwyer et al. [40]. A similar cluster-based approach was later considered by Sun et al. [126]. Böttger et al. [22] and Zielasko et al. [177] extended the edge bundling technique to a three-dimensional space using a mean-sift method and a 3D force-directed method, respectively. Most recently, Kwon et al. [70] applied edge bundling to an immersive environment with illumination. Their work represented an early attempt to combine virtual reality techniques with the edge bundling method.

Statistics

Histograms and KDE are two statistical methods often used to overcome the overlapping problem. However, the histogram has some disadvantages, as mentioned

in [101] and [71]; it is less smooth and limits the bin selection. Lampe and his colleagues [71] visualized large-scale traffic data on a map using KDE to overcome the overlapping problem. Similarly, the KDE technique was adopted by Willems et al. [154] to deal with the problems involved in visualizing moving objects. The KDE method also has been used to implement line bundling, as presented in [59]. Because a set of data may belong to different groups, [96] presented a splatter plot to visualize group contours using an extended KDE method. In addition to the general KDE, Correll and Heer [33] presented a Bayesian-based method to further improve the KDE effectiveness in information visualization.

The choice of bandwidth for KDE is extremely important in stream-based visualization because it determines the accuracy of the density estimation. For streaming data visualizations, adaptive bandwidth estimation was adopted by Lampe et al. [71] to estimate the density of data with respect to the level of detail. However, their method did not include adapting the bandwidth of KDE for each region at one level, which may lead to inaccurate density maps.

2.3 Representations of Dynamic Data

Dynamic data representation is a major research topic in information visualization, particularly the visualization of streaming data. Their categories are summarized in this section and include interpolation methods, variation feature tracking, and timeline visualization.

Interpolation methods

Much work has been performed to achieve dynamic effects from streaming data through interpolation. Figure 2.5(left) shows sequences of dynamic data, and Figure 2.5(middle) shows an illustration of the process of interpolation. Krstajic and Keim [69] addressed the challenges involved in measuring changes and maintaining context in dynamic information visualization. Assuming that each frame is a

scatterplot representation and each point's position in each frame is known, position interpolation is a suitable method to overcome ghosting. Position interpolation, as explained in the works of Robertson et al. [111] and Du et al. [38], involves dynamically computing the movement trajectory for each point. Figure 2.4 shows a line bundling method to interpolate the point trajectories. However, a constraint of this method is that each point in one frame should match a point in the next frame; in reality, not all point sets meet this constraint. For example, when one user's record (represented as a point in the visualization) appears in only one frame, it is difficult to interpolate its position to an unknown position in the next frame. Furthermore, point interpolation may lead to visual confusion as the number of animated points increases.

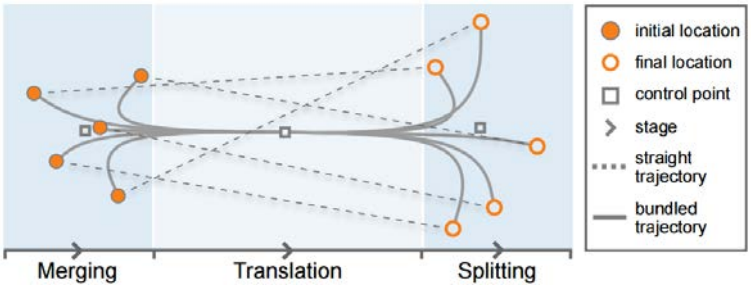


Figure 2.4: A trajectory bundling method for point transitions [38].

Generating density maps frame by frame is a practical solution candidate to overcome visual confusion because KDE normalizes all of the points in a frame into a structured density map. Still, visual ghosting might appear, as shown in Fig. 3.3(middle), if we adopt linear interpolation between two density maps when the frame displacements are large.

A nonlinear retargeting algorithm such as optical flow [55] could be used to establish accurate correspondences between two density maps to avoid visual ghosting. However, the time-consuming iteration is required to morph density maps and the miss-convergence problem makes optical flow impractical for meeting the requirements of dynamic visualizations. Mahajan et al. [95] described a path-based moving gradient approach that can handle complex non-rigid morphing. Liao et al. [82]

presented a semi-automated morphing approach for images that is suitable for large-scale transitions with only a few interactive operations. Unfortunately, the moving gradient [95] method requires further improvement when content exhibits large-scale changes between frames. Moreover, content-based interactive operations [82] are not convenient for users because the contents change frequently over time in dynamic visualizations.

Apart from the temporal interpolation discussed before, spatial interpolation is another direction to generate missing data. Zheng et al. [174] applied linear/Gaussian and an artificial neural network (ANN) for the interpolation of air pollution data, produced by the sparse air quality monitoring stations in a city. Another work called BlueAer [57] improved the accuracy of interpolation toward 3D space by implementing a PM2.5 monitoring prototype system. Spatial interpolation was also used for noise diagnosis, as described by Zheng et al. [175]. They provided a method to estimate city noise based on hybrid data sources such as road networks, points of interest (POI), user check-ins, and a noise complaint database. However, the current spatial interpolations have paid less attention to visual effects and aesthetic criteria.

Therefore, for dynamic data, the existing interpolation techniques require more improvement to achieve better visual effects and data enrichment.

Variation feature tracking

Scientific visualization studies such as those in the field of feature tracking are also related to the visualization of dynamic data. Extracting features such as trend information from streaming data can be of great benefit to users. Figure 2.5(right) describes the feature tracking among sequences of data. Similar to scientific visualization, the variation features in information visualization can be summarized as follows: crossing, moving, disappearing, appearing, division, and aggregation. Woodring et al. [157] used the wavelet transform to change point sets into curve sets along a time axis to track time-varying trends. Flow-based scatterplots [28] were presented to highlight variations in flow data. Samtaney et al. [118] proposed an

algorithm to extract the coherent features from unstructured time-dependent scalar fields; they summarized these interaction features as continuation, creation, dissipation, bifurcation, and amalgamation. Based on the work of Samtaney et al. [118], Ozer and his colleagues [103] tracked clusters of features from time-varying 3D flow fields to improve the performance. Grottel et al.[51] addressed flow groups to study molecular dynamics by visualizing the cluster evolution over time. Cluster structural variations were visualized by Turkay et al. [133] using an interactive cluster viewing design. For tracking variation features among spatio-temporal data, Cao et al. [26] proposed a tensor-based approach to dynamically detect the anomalous patterns.

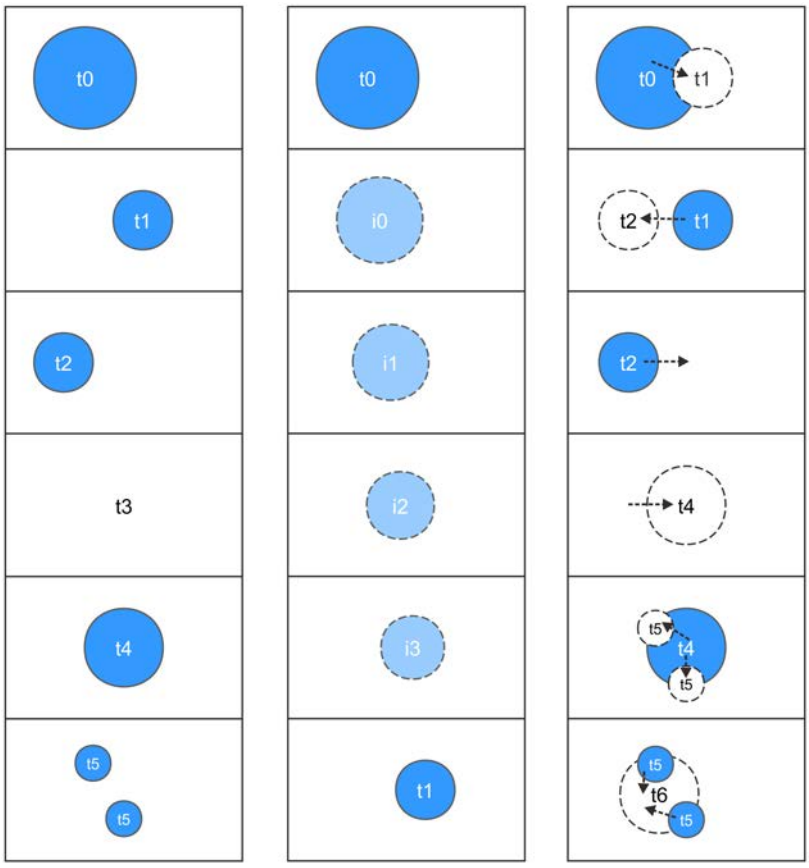


Figure 2.5: An illustration of the methods of interpolation and feature tracking: (left) original time series data; (middle) the interpolation process between two frames; (right) the tracked variation features. Dashed circles in the middle part are generated data. Dashed circles in the right part are the target data of feature tracking. Dashed arrows indicate the variation trends.

Timeline visualization

The method of timeline visualization transforms temporal data into a two-dimensional space. Figure 2.5(left) is the simplest timeline visualization, visualizing the data sequence by sequence. Figure 2.6 shows a more abundant result using a timeline-based approach called CloudLines [68], which is a suitable tool to visualize event episodes among streaming data. Another work [93], called Event Cueing, studied the spatio-temporal distribution of evolving media discourse. Event Cueing utilizes the benefit of the timeline and allows users to explore underlying spatial patterns. Brehmer et al. [23] surveyed a number of timelines and designed a hybrid timeline representation that combines different timeline representations in a three-dimensional design space. Apart from the direct visualization timeline, the similarity measurement of time series records has been discussed in [156], which is beneficial to the compatible visualization of continuous data. Visualizing patterns through timeline deformation is put forward by Bach et al. [12]. Their work overcame the space limitation problem of the prior timeline visualization while preserving the time information.

Another frequently used tool is the Sankey flow diagram, as mentioned in [24] and [140], which is used to visualize the evolution of time-dependent data. Figure 2.7 shows an example of a Sankey diagram that represents the check-in region variations of a website in the U.S.A. at different time steps. Different from CloudLines [68], the Sankey flow includes node aggregation and division. Vehlow et al. [139] adopted the Sankey flow to visualize the dynamic graph evolution and enhanced the Sankey flow through relationship representation. A similar work is egoSlider [160], which aims at the analysis and exploration of egocentric network evolution.

2.4 Information Visualization Resizing

When the visualization display is changing, content-aware resizing is required to protect and enhance the salient regions. For stream visualization, it is essential to implement content-aware visualization because it is more sensitive to the size of the

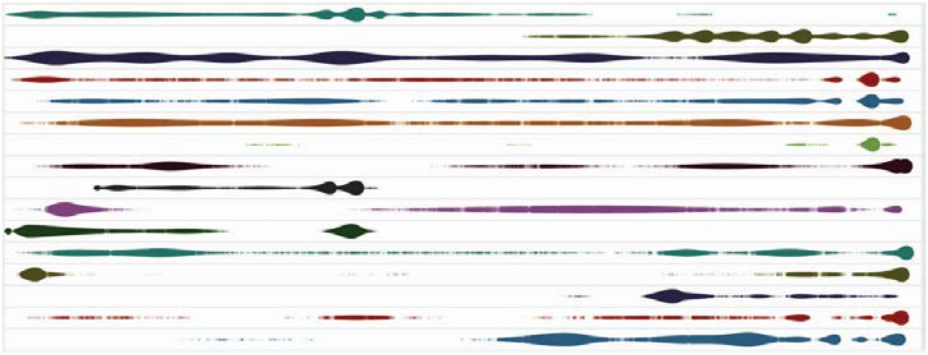


Figure 2.6: CloudLines [68].

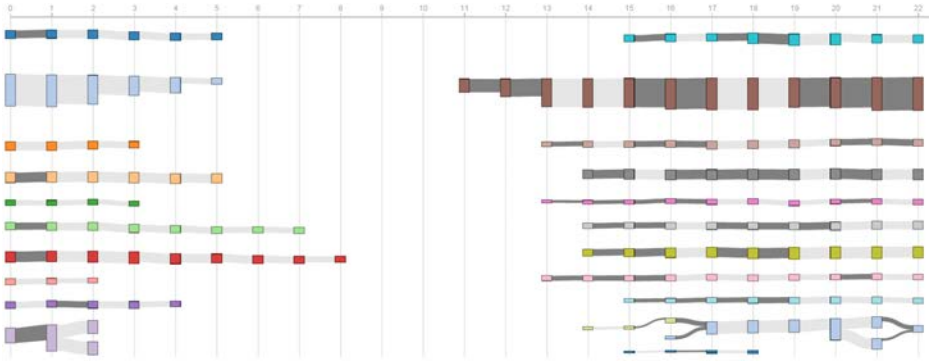


Figure 2.7: An example of a Sankey diagram.

data and displays. In the following subsections, we review the methods of information visualization resizing such as content-aware resizing and the saliency map. Content-aware resizing is a technique that re-targets the visualization content while preserving the important parts. The method of a saliency map is used to define which parts on a canvas are important. These two techniques are frequently used in the image processing field [116] and have recently been adopted to deal with the content resizing problem in information visualization, as mentioned in the work of Wu et al. [162].

Content-aware resizing

Many researchers have recently been working on the content-aware resizing problem [116]. This problem is also known as *focus+context resizing* or *saliency-aware resizing*. Generally, content-aware resizing methods can be classified into (a) pixel-based and (b) mesh-based. Pixel-based methods are discrete. Seam carving [9] was the

first proposed pixel-based method that is related to the content-aware resizing of images. Rubinstein and his colleagues improved seam carving by using the forwarding energy [117]. Seam carving was also improved by Xu et al. [165] by transforming the extracted image structure. Battiato et al. [16] adopted gradient vector flow to generate the path of seam carving and achieved better results. Unfortunately, these methods are based on some form of pixel energy or intensity levels and are not applicable to vector-based visualizations such as graphs and geographical maps (GGM). In addition, the content cannot be further enhanced when it is resized by seam carving. Furthermore, the iteration of seam carving is time-consuming and puts severe constraints on real-time interactive editing. Some improvements have been presented by Yael et al. [108] and Wu et al. [159], which achieved better visual results than the original pixel-based methods, but these still lead to missing information in information visualizations.

On the other hand, mesh-based methods for resizing provide a degree of continuity for the underlying regions. Gal et al. [48] presented a novel resizing method using a manual feature mask and an underlying grid. Wang et al. [145] described an optimized resizing approach that overcomes the edge distortion problem that was not considered before. Zhang et al. [171] used edge similarity constraints to obtain better results on object edges. Panozzo and his colleagues [105] used axis-aligned deformations to resize images with content preservation in real time. Since the axis-aligned deformation method has fewer degrees of freedom for deformation than other common mesh-based methods, it cannot be easily extended to fit various target displays and multi-layer visual information. Kaufmann and his colleagues [63] used a FEM model to reduce the degrees of freedom in resizing to perform the real-time resizing of images.

Currently, the focus is increasingly put on content-aware resizing for information visualization. Examples can be found in tree maps [132], metro maps [142, 158, 31], word clouds [89], and road networks [52, 85]. These methods provide effective algorithms for spatial information visualizations. Wu et al. [162] utilized a significance

map and quad-based deformation to put forward a general resizing framework for visualization. However, for complex elements in multiple layers, such as geographical information and large graphs, the existing methods are still challenged.

Saliency map

In content-aware retargeting for both images and visualizations, the saliency map is treated as a form of energy of pixels, which can be used to build an energy function, such as shown by Wang et al. [145]. Itti et al. [60] took into consideration the human visual system and denoted the significance of points from a natural image. They also presented an effective feed-forward feature-extraction method to compute a saliency map from it. Frintrop et al. [47] extended the work of [60] and computed the saliency at the pixel level with high performance. Wang et al. [145] used the method explained in [60] to assign a significance threshold to quads. Jänicke and Chen [61] proposed an effective method to measure the visualization quality via a saliency map approach. Wu et al. [162] combined a clutter map and a DOI map into a significance map, which is another type of saliency map. Engelke et al. [41] studied fixation density maps and showed that a saliency map can be generated by eye tracking devices. Zhang et al. [173] used an anisotropic diffusion equation to further improve the accuracy of saliency detection.

Achanta et al. [1] summarized five basic requirements for saliency detection and proposed a simple implementation, called FT. FT could be easily extended to consider multiple visual features in images. Goferman et al. [49] presented a novel method to detect a context-aware saliency map that aimed to represent the dominant objects in an image. They argued that the context of a region should also be considered to generate a more accurate saliency map and demonstrated that their approach has potential applications in image resizing. Yan et al. [166] proposed a hierarchical saliency detector that can generate a multi-layer saliency map for natural images. A cellular automaton with different layers was used by Qin et al. [109] to detect saliency among similar image patches. However, in information visualizations, the methods

of [49], [166], and [109] require further adaptation for visual saliency detection.

2.5 Visual Querying

For large-scale streaming data, interactive querying is essential to rapidly find useful information. In this section, we review related methods on visual querying that includes content as follows: stream mapping, feature matching, querying interaction, and query representation.

A frequently used data aggregation and organization method is kernel density estimation (KDE), as presented in [125]. Many KDE-based studies have been presented to solve visualization problems. Hurter et al. [59] provided a KDE-based visual clustering method to overcome the overlapping problem in complex graph drawing. Cottam et al. [34] presented an aggregation process that enables the concise expression of the alpha composition. An extension of KDE [96] has been presented to further overcome the overdrawing problem in visualizing high-dimensional data. However, the hidden details of the density map have not yet been discussed.

For feature matching, the popular SIFT [92] method can be adopted to generate feature points from saliency maps generated by the data visualizer. However, this method requires time-consuming calculations. In particular, if a saliency map includes fewer features, such as a smooth density map generated using KDE, SIFT is not suitable for feature matching for the purpose of effective visual querying. Moreover, the main advantages of SIFT, such as detecting translational or rotating features, cannot generally be used in visual querying because the querying areas are nearly always fixed relative to the streaming data. Similarly, other improved image matching methods, such as those mentioned in [124], are also not suitable for solving the visual querying problem.

Correll and Gleicher [32] introduced a sketch-based visual query framework for the understanding and exploration of time series data. Recently, researchers have begun to focus more on interactive querying for large-scale data due to the demand for

big data analytics. Ferreira et al. [45] constructed a spatio-temporal system that supports the interactive visualization of data patterns and potential details. Liu et al. [90] provided an interactive querying method on a large-scale dataset. Their system can support a rectangle-based interaction. Lins et al. [86] further presented *nanocubes* to support very large geographical data querying and browsing and demonstrated their system on a large Twitter location dataset. When the query results are spatio-temporal data, the interaction in imMens [90] requires an improvement to better visualize them.

Numerous line-based visualizations can be adopted to present query spatio-temporal data. CloudLines [68] is a suitable tool for visualizing event episodes in streaming data. Although the representation of CloudLines is simple, it provides a suitable timeline visualization tool to visualize the evolution of events. Other similar works, such as Storyline [127] and StoryFlow [88], have also been presented to trace the states of time-varying data.

2.6 Conclusion

This chapter presented the major methodologies that have been widely used in the information visualization of large-scale dynamic data. We first presented the related frameworks of data processing and visualization. We then introduced the visual clustering theories. The common dynamic data representation approaches were also discussed, including those for interpolation, variation feature tracking, and timeline visualization. We then introduced the concept of content-aware resizing that has commonly been used to preserve and enhance the important content when the size of the display is changing. We finally summarized the related work of visual querying and addressed real interactive applications on temporal data querying, especially on spatio-temporal data.

CHAPTER 3

SMOOTH DYNAMIC VISUALIZATION OF STREAMING DATA

3.1 Introduction

Research on streaming data visualization is becoming significant with the increasing volume of time-varying data from areas such as social media networks, air quality monitoring, GPS tracking, and real-time online retailing. When visualizing streaming data, the two-dimensional point data model is the most commonly used model in practice because most of the features in the data stream can be described as points on a two-dimensional spatial grid, such as geographical locations, nodes in network graphs, and atmospheric or environmental sensor data.

Scatterplots have been used to study two-dimensional data for many years, but they suffer from overlapping (Fig. 1.2) when the data stream contains high-density point structures. This problem is known as overdrawing, and it has become more significant as the size of data has exploded. In addition, directly visualizing streaming points as dynamic scatterplots without interpolation leads to a significant problem of sudden sharp changes because visual continuities are missing between two scatterplots, as shown in Fig. 3.3(a).

To create visual continuity, conventional linear interpolation between two frames is a practical solution. Nevertheless, when the data streams contain large variations, this approach produces visual ghosting patterns, as shown in Fig. 3.3(b). In addition, the information generated by linear interpolation does not always produce visually acceptable trend patterns, particularly in point cloud regions. Therefore, a smooth morphing (smooth blending) approach between frames is necessary to produce patterns that are easier to observe and evaluate in data streaming visualization.

To address the above issues, we present a novel framework to represent streaming points, called *StreamMap* (Fig. 3.1). StreamMap offers a smooth dynamic visualization in large regions of point clouds, as shown in Fig. 3.3(c). We define the task of streaming point visualization as both a density estimation problem and a problem in creating a smooth interpolation between a pair of frames that contain two sets of points during a given time interval. Our method for visualizing the high-density point streams is to morph the two frames smoothly, thus overcoming overlapping and sudden changes in the dynamic visualization. Compared with prior algorithms, our approach not only overcomes the overdraw problem in high-density point visualizations but also reduces the artifacts common in dynamic visualizations. In addition, we also dynamically show the evolution of features in two frames.

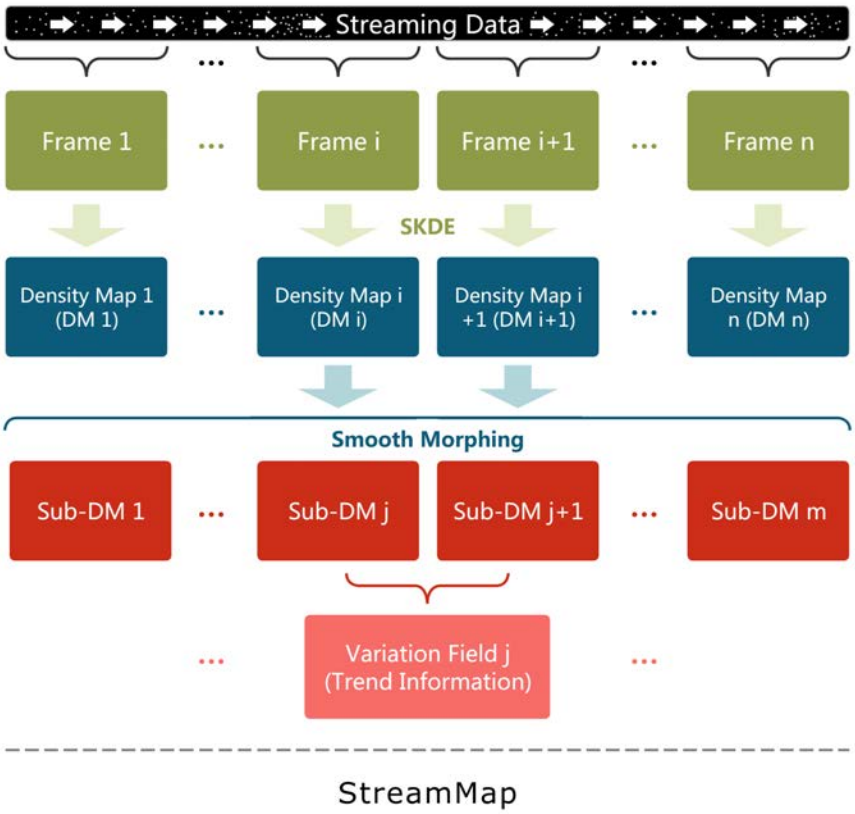


Figure 3.1: The definition of StreamMap. We assume that points in a stream are similar to moving darts and that the frame is the collection of darts on a projected 2D plane representation, as shown at the top of the figure. Streaming data are organized as a density map through data aggregation and density estimation. StreamMap's diffusion model supports the composition of sub-DMs between two density maps.

StreamMap is more suitable for streaming data with a "flow" nature. For example, when the data sets are streaming photo locations, such as from Flickr.com, interpolated sub-frames would not reflect valid states because the points of photo locations and the point clouds are normally independent. Hence, we assume that the input data set of StreamMap has a "flow" nature. In addition, because our work focuses on the streaming point visualization, we assume that points in different frames are not necessarily linked. A trajectory is an example of a linking data set. This assumption is different from the related works of Willems et al. [154] and Andrienko et al. [6]. Without the existence of point links, StreamMap conventionally offers a smooth representation of changes.

We applied our method to three cases, all of which include 2-dimensional points and have a "flow" nature. Nevertheless, these cases have their own characteristics. Artificial data (Sec. 3.4.1) include explicit continuous point distributions; therefore, we adopt artificial data to evaluate the effectiveness of the morphing approach. Because a crowd of people (Sec. 3.4.2) includes heterogeneous point densities, it was used to demonstrate the density estimation method. Air pollution (Sec. 3.4.3) is more unusual because the point positions are fixed at different time steps.

3.2 Definition

We assume that the streaming point data used for visualization were pre-accumulated over a set of time intervals $\{t_1, ..., t_i\}$. The accumulated points within a given consistent time interval t_i can be defined as a frame F_i. We assume that the boundary of each frame is fixed. The *super kernel density estimation* (SKDE) method is used to transform a frame F_i into a density map D_i through adaptive density estimation. Each density map is a grayscale image with the same size.

In addition, we provide a robust method to smoothly morph between each pair of density maps, such as from D_i to D_{i+1}. To simplify the notation, we define each pair of density maps as I and T in our morphing model, as shown in Fig. 3.2. The input

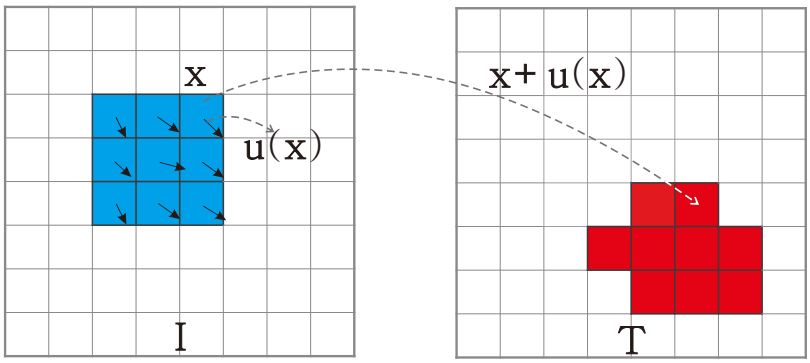


Figure 3.2: An example of the density diffusion between two density maps. $\mathbf{x}$ means a pixel in a density map, and $\mathbf{u}$ indicates the transformation value. I and T are a pair of inputs in the morphing model.

to our method is a density map I, and T is the target density map. We assume that the morphing process from I to T can be achieved by applying a transformation to I. We define it as $I_{\mathbf{u}}$, which transforms the input density map using the deformation field function $\mathbf{u}(\mathbf{x})$,

$$\mathbf{u}\{I\}(\mathbf{x}) = I(\mathbf{x} + \mathbf{u}(\mathbf{x})), \mathbf{x} \in \mathbf{\Omega}, \tag{3.1}$$

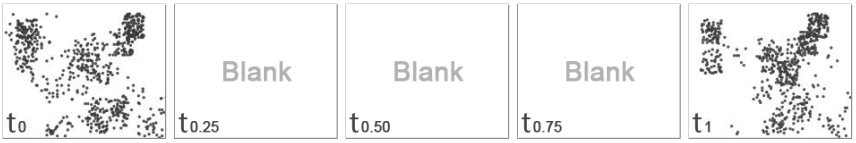
where $\mathbf{x}$ indicates the position of a pixel in the density map, $\mathbf{u}$ can be written as $(u_a, u_b)^T$, u_a denotes the horizontal component, and u_b denotes the vertical component. Furthermore, we define the in-between density map (sub-DM) generated by the morphing process between I and T as S_i, where i indicates the in-between density map index.

3.3 StreamMap

Our *StreamMap* model is constructed as follows:

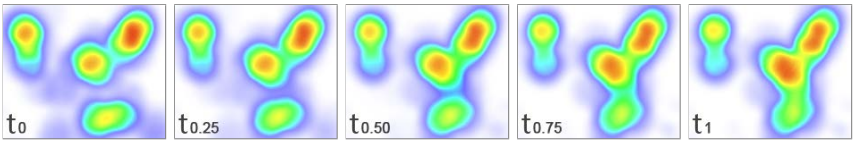
(1) To overcome the overdraw problem, we propose a superpoint-based estimation method called SKDE to achieve an accurate density map from a time period of streaming point data. SKDE achieves an accurate density map by using adaptive kernel selection with a fast point-clustering method.

(2) To create the visual continuity and solve the visual ghosting problem, we use a



(a) Visualizing two frames of points via scatterplots without an interpolation.

(b) Linear blending of two density maps.



(c) Smooth morphing of two density maps using StreamMap.

Figure 3.3: Smooth dynamic visualization compared with the scatterplot and the linear blending methods. In-between frames of two scatterplots are blank, if no interpolation is applied. The leftmost and the rightmost density maps are inputs of the blending.

smooth process to dynamically visualize data streams.

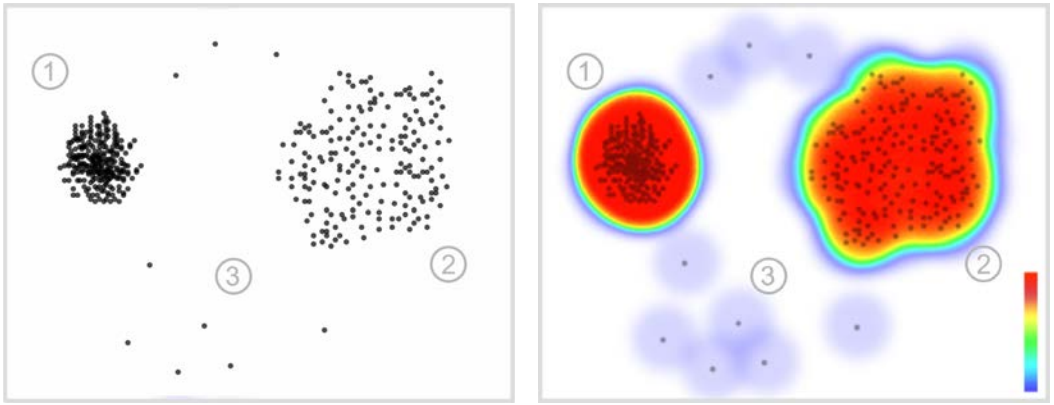
(3) To identify and represent the trend in point streams, we design a trend representation that can help users obtain insights into the variation of point streams.

3.3.1 Super kernel density estimation

The *super kernel density estimation* (SKDE) approach for visualizing high-density streaming point data uses single pixels to represent multiple data points. The basic idea behind SKDE is to achieve an adaptive estimation of the density in a region by aggregating the value of each influential point. For this purpose, we generate point clusters called superpoints from the point set and assign clusters with different estimated kernel sizes with respect to the point number in the cluster.

Adaptive kernel density estimation

We improve the prior KDE approach [71] by making it possible to estimate density using adaptive bandwidth. The input of SKDE is a set of points $\boldsymbol{F}$, and the output is



(a) Original points. (b) A density map estimated using KDE.

Figure 3.4: KDE result with fixed bandwidth. The number texts on the figure indicate three different density points.

a grayscale density map $\boldsymbol{D}$, where the size of the density map is defined as having $dwidth$ and $dheight$. In our experiments, $dwidth$ is 1200 and $dheight$ is 780. We formulate SKDE as $K(\mathrm{x})$, as follows:

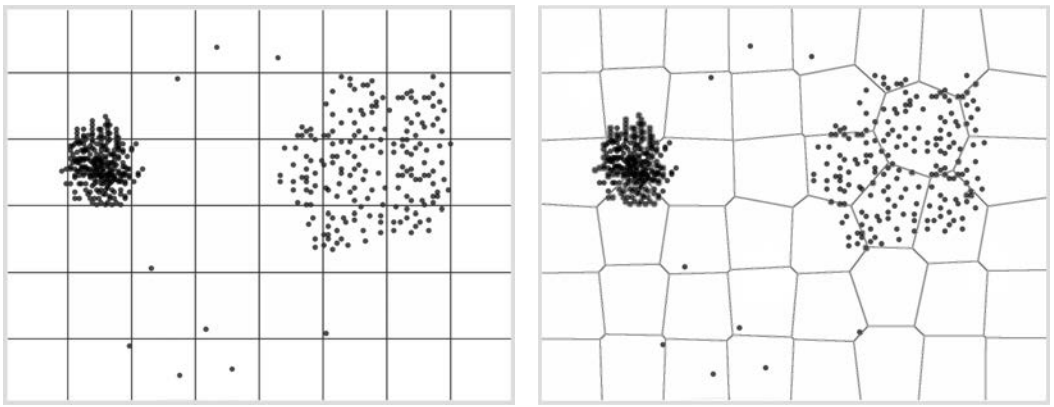
$$K(\mathrm{x}) = \frac{1}{\mathrm{n}} \sum_{\mathrm{j}=1}^{\mathrm{n}} \frac{1}{\mathrm{h_j}} \mathrm{G}(\frac{|\mathrm{x}-\mathrm{x_j}|}{\mathrm{h_j}}), \mathrm{x_j} \in \mathrm{F}, \mathrm{x} \in \mathrm{F}, \tag{3.2}$$

where n is the number of points in the set $\boldsymbol{F}$, $G(\mathrm{x})=\frac{1}{\sqrt{2\pi}}\mathrm{e}^{-\frac{\mathrm{x}^2}{2}}$ is a standard Gaussian kernel, and h_j is the bandwidth of SKDE that defines the range of the kernel function. Each point has its own bandwidth.

Traditional KDE with a fixed bandwidth suffers from the creation of artificial blocks as shown in set-3 of Fig. 3.4(b) as there are some independent points in spatial space. In addition, the KDE result in Fig. 3.4(b) shows a poor visual result because medium-density points at set-2 were estimated with high density that is similar to high-density points at set-1. Accurate manual bandwidth adjustment for each frame may achieve a better visual estimation effectiveness. However, it is difficult to achieve in a large-scale stream visualization. Hence, SKDE is more suitable for estimating the density of streaming points because it provides automatic bandwidth assignment.

Superpoint generation

The adaptive bandwidth setting is achieved through *superpoint generation* (SG), which clusters the points into nearly uniform-area superpoints. SG is inspired from the superpixel algorithms [110] and [2] used in the image processing field. Superpixel, which was first presented by Ren and Malik [110], is a method that can segment an image into nearly uniform superpixels (from pixel level to region level). Because each superpixel can represent its region, the difficulty of an image segmentation is reduced to the region level. SLIC [2] is an improvement of the work of Ren and Malik [110], which limits the search region to accelerate the superpixel generation. We assign a bandwidth to each superpoint. All points inside the superpoint will then be estimated with the superpoint's bandwidth. Superpoints with high point densities will be assigned larger bandwidths. Conversely, superpoints with sparse points will be assigned smaller bandwidths.



(a) Initial regular grids. (b) Irregular grids.

Figure 3.5: Irregular grid division of the SG clustering methods.

We assume that F will be clustered into k superpoints. Initially, the points are distributed to k regular superpoints (regular grids) of size 64, in which k initial superpoint centers will be calculated. We can calculate k through $k = \frac{\sqrt{\text{dwidth}\cdot\text{dheight}}}{8}$. The superpoint center is equal to the mean location of points inside the superpoint. We assume that each point is in a 2D space. Each point will be assigned to its closest superpoint by calculating the distances between it and its neighboring superpoint

centers. We then recalculate the superpoint centers and repeat the point assignment process. When no points have moved to new superpoints with this iterative process, the iteration stops.

We found that SG can achieve convergence for most point sets in 8 iterations; therefore, we applied 8 iterations in our experiments. Then, we calculate the bandwidth of superpoint sh_i as follows:

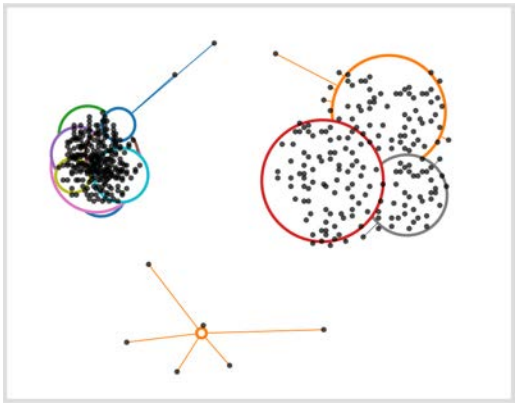
$$sh_i = \frac{n_i}{\sum_{j=1}^{n_i} \|\mathrm{p_{ij}} - \mathrm{c_i}\|}, i \in [1, k], \tag{3.3}$$

where n_i indicates the number of points associated with superpoint i, $\mathrm{p_{ij}}$ is a point inside superpoint i, $\mathrm{c_i}$ is the superpoint center, and $\sum_{j=1}^{n_i} \|\mathrm{p_{ij}} - \mathrm{c_i}\|$ is the variance of points from their superpoint center. Because we assume that the points belonging to the same superpoint have a consistent bandwidth, all point bandwidths (h_j in Eq. 3.2) can be achieved after calculating sh_i.

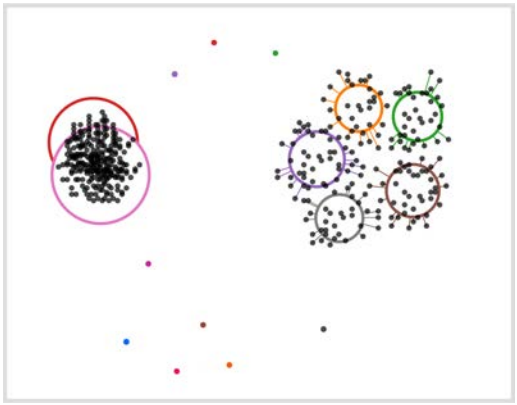
Finally, a grayscale density map $\boldsymbol{D}$ for each frame can be generated using Eq. 3.2. SG is an improvement of k-means [94] that exhibits improved performance because the required distance calculations are reduced by limiting the search region. Figure 3.5(a) presents the initial regular grids of SG, and Fig. 3.5(b) shows an example of the irregular grids in the SG process. Figure 3.6(a, b) shows a comparison of the k-means and SG clustering methods. Each circle in Fig. 3.6(a, b) represents a cluster. The circle size indicates the point number in the cluster. All cluster members are linked with their cluster circle. As shown in Fig. 3.6(d), the SG method achieves a better density map than the k-means method. Set-2 in Fig. 3.6(d) are estimated with a medium density that is more accurate than the one in Fig. 3.6(c).

3.3.2 Smooth morphing

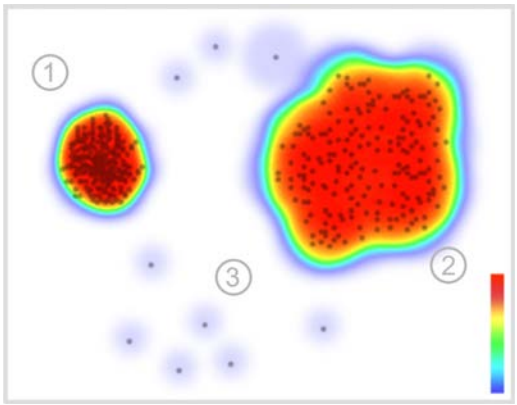
We now provide the details of our smooth morphing model. First, a basic morphing model is proposed to solve the morphing problem. Second, we propose a helper seed method to compensate for a weakness in the basic morphing model. We also



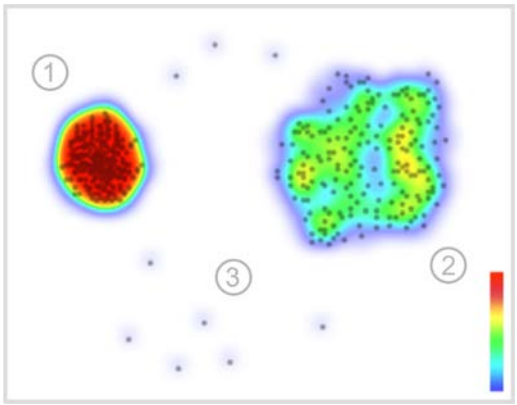
(a) K-means clustering result.



(b) SG clustering result.



(c) Adaptive KDE result using k-mean method.



(d) Adaptive KDE result using SG method (SKDE).

Figure 3.6: A comparison of the k-means and the SG clustering methods.

further improve the effectiveness of the morphing process by overcoming density nonconformity in the morphing process.

Diffusion model

Inspired by the Demons diffusion model [129], we formulate the morphing operation between a pair of density maps as the following optimization problem:

$$\begin{cases} \boldsymbol{\delta_{\mathbf{u}}}^{(n+1)} = \arg\min_{\boldsymbol{\delta_{\mathbf{u}}}} \{ \underbrace{E_d(I, T, \boldsymbol{\delta_{\mathbf{u}}})}_{data} + \underbrace{\lambda E_r(\boldsymbol{\delta_{\mathbf{u}}})}_{\text{regularization}} \} \\ \mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \boldsymbol{\delta_{\mathbf{u}}}^{(n+1)} \end{cases}, \tag{3.4}$$

where E_d is a data term that guarantees the accuracy of the transformation, E_r is a regularization term that ensures the smoothness of the transformation, n indicates the

iteration step, $\mathbf{u}^{(n)}$ is the transformation of the density map, and λ is a free parameter used to adjust the smoothness. We define the data term, E_d, as follows:

$$E_d = \int_\Omega \left\| I_{\mathbf{u}^{(n)}} + (\nabla I_{\mathbf{u}^{(n)}})^{\mathrm{T}} \boldsymbol{\delta}_{\mathbf{u}} - T \right\|^2 \mathbf{dx}, \tag{3.5}$$

where T denotes the value in a target density map, $I_{\mathbf{u}^{(n)}}$ denotes the value in a transformed density map after applying the transformation $\mathbf{u}^{(n)}$ to I, and ∇ is the gradient operator. $\nabla I_{\mathbf{u}^{(n)}}$ indicates $(\partial_x I_{\mathbf{u}^{(n)}}, \partial_y I_{\mathbf{u}^{(n)}})^{\mathrm{T}}$, where $\partial_x I_{\mathbf{u}^{(n)}}$ and $\partial_y I_{\mathbf{u}^{(n)}}$ are two components of $\nabla I_{\mathbf{u}^{(n)}}$. $I_{\mathbf{u}^{(n+1)}}$ is defined as $I_{\mathbf{u}^{(n+1)}} = I_{\mathbf{u}^{(n)}}(\mathbf{x}+\mathbf{u}^{(n+1)}(\mathbf{x}))$. We then define the regularization term, E_r, as follows:

$$E_r = \int_\Omega \|\boldsymbol{\delta}_{\mathbf{u}}\|^2 \mathbf{dx}. \tag{3.6}$$

By minimizing the functional $E(\boldsymbol{\delta}_{\mathbf{u}}) = E_d + \lambda E_r$ with respect to the vector function $\boldsymbol{\delta}_{\mathbf{u}}$, we can obtain $\boldsymbol{\delta}_{\mathbf{u}^{(n+1)}}$. We define two components of $\boldsymbol{\delta}_{\mathbf{u}^{(n+1)}}$, which are ${\boldsymbol{\delta}_{\mathbf{u}y}}^{(n+1)}$ and ${\boldsymbol{\delta}_{\mathbf{u}y}}^{(n+1)}$. According to the theory of the calculus of variations, the Euler-Lagrange equation of $\boldsymbol{\delta}_{\mathbf{u}}$ is obtained by setting $E'(\boldsymbol{\delta}_{\mathbf{u}}) = \mathbf{0}$, as shown in Eq. 3.7.

$$E'(\boldsymbol{\delta}_{\mathbf{u}}) = \mathbf{0} \Rightarrow (I_{\mathbf{u}^{(n)}} - T)\nabla I_{\mathbf{u}^{(n)}} + (\nabla I_{\mathbf{u}^n} \cdot \boldsymbol{\delta}_{\mathbf{u}})\nabla I_{\mathbf{u}^n} + \lambda \boldsymbol{\delta}_{\mathbf{u}} = \mathbf{0} \tag{3.7}$$

$${\boldsymbol{\delta}_{\mathbf{u}x}}^{(n+1)} = \frac{T - I_{\mathbf{u}^{(n)}}}{(\partial_x I_{\mathbf{u}^{(n)}})^2 + \lambda} \partial_x I_{\mathbf{u}^{(n)}} \tag{3.8}$$

$${\boldsymbol{\delta}_{\mathbf{u}y}}^{(n+1)} = \frac{T - I_{\mathbf{u}^{(n)}}}{(\partial_y I_{\mathbf{u}^{(n)}})^2 + \lambda} \partial_y I_{\mathbf{u}^{(n)}} \tag{3.9}$$

Here, we provide a detailed derivation of Eq. 3.8 and Eq. 3.9.

Let $E_{\mathrm{d}} = \int\limits_\Omega \left\| I_{\mathrm{u}^{(n)}} + (\nabla I_{\mathrm{u}^{(n)}})^T \cdot \delta_{\mathrm{u}} - T \right\|^2 d\mathrm{x}$ and $E_r = \int\limits_\Omega \|\delta_{\mathrm{u}}\|^2 d\mathrm{x}$, where ∇ is the gradient operator. The denotation $\nabla I_{\mathrm{u}^{(n)}}$ is $(\partial_x I_{\mathrm{u}^{(n)}}, \partial_y I_{\mathrm{u}^{(n)}})^{\mathrm{T}}$, where $\partial_x I_{\mathrm{u}^{(n)}}$ and $\partial_y I_{\mathrm{u}^{(n)}}$ are two components of $\nabla I_{\mathrm{u}^{(n)}}$. Through minimizing the functional $E(\delta_u) = E_d + \lambda E_r$ with respect to the vector function δ_u, we can get ${\delta_u}^{(n+1)}$, namely, ${\delta_u}^{(n+1)} = \arg\min\limits_{\delta_{\mathrm{u}}} \{E_d + \lambda E_r\}$.

According to the theory of the calculus of variations, the Euler–Lagrange equation of $E(\delta_u)$ is computed as follows.

Firstly, we introduce $\alpha\eta$ as a permutation of δ_{u}, where $\alpha \in R$ is a number, and $\eta = (\eta_x, \eta_y)$ is a vector function similar to $\delta_u = (\delta_{ux}, \delta_{uy})$. Let $J(\alpha) = E(\delta_{\mathrm{u}} + \alpha\eta)$, where δ_{u} and η are fixed. Then, according to the Gatuex derivation of $E(\delta_u)$, $E'(\delta_{\mathrm{u}})$, is defined by $E'(\delta_{\mathrm{u}}) = \frac{dJ(\alpha)}{d\alpha}\Big|_{\alpha=0} = 2\int_\Omega [(I_{\mathrm{u}^n} - T)\nabla I_{\mathrm{u}^n} \cdot \eta + (\nabla I_{\mathrm{u}^n} \cdot \delta_{\mathrm{u}})(\nabla I_{\mathrm{u}^n} \cdot \eta) + \lambda\delta_{\mathrm{u}} \cdot \eta]d\mathrm{x}$.

The Euler-Lagrange equation of $E(\delta_u)$ is obtained by setting $E'(\delta_{\mathrm{u}}) = 0$, namely, $2\int_\Omega [(I_{\mathrm{u}^n} - T)\nabla I_{\mathrm{u}^n} \cdot \eta + (\nabla I_{\mathrm{u}^n} \cdot \delta_{\mathrm{u}})(\nabla I_{\mathrm{u}^n} \cdot \eta) + \lambda\delta_{\mathrm{u}} \cdot \eta]d\mathrm{x} = 0$, for $\forall\eta$.

Then, the Euler-Lagrange equation of $E(\delta_u)$ is formulated as follows: $(I_{\mathrm{u}^n} - T)\nabla I_{\mathrm{u}^n} + (\nabla I_{\mathrm{u}^n} \cdot \delta_{\mathrm{u}})\nabla I_{\mathrm{u}^n} + \lambda\delta_{\mathrm{u}} = 0$, from which we can lead to the solution as shown in Eq. 3.8 and Eq. 3.9.

From Eq. 3.7, we arrive at the solution as shown in Eq. 3.8 and Eq. 3.9. We set the initial values as $\boldsymbol{\delta_{\mathbf{u}}}^{(0)} = \mathbf{u}^{(0)} = \mathbf{0}$ and $I_{\mathbf{u}^{(0)}} = I$. The iterative step in Eq. 3.4 can be repeated until $I_{\mathbf{u}^{(n+1)}}$ is nearly equal to T. We observed that 16 iterations are sufficient for most morphing cases; therefore, we apply $n = 16$ in our experiments. Here, λ is set to 0.4, which is an experiential value that can achieve a better smoothing.

In our implementation, we approximate the ∇ operator using the following equations:

$$\begin{cases} g_x(i,j) = \frac{1}{2}(D(i-1,j) - D(i+1,j)), \\ g_y(i,j) = \frac{1}{2}(D(i,j-1) - D(i,j+1)). \end{cases}, \tag{3.10}$$

where $D(i,j)$ indicates the value at position (i,j) in the density map.

Helper seed

The diffusion model is subject to two constraints: an accuracy constraint and a smoothness constraint. The accuracy constraint ensures that the value of a pixel remains constant when it is moved from $\boldsymbol{x}$ to $\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})$, whereas the smoothness constraint ensures that the displacement field of each pixel varies smoothly. However, in density maps created from streaming points, some morphing patterns may

not fulfill these constraints in a finite number of iterations.

As shown in Fig. 3.7, we summarize six basic morphing patterns for density map morphing. The region with the blue color (called I) is defined as the original region, and the red region (called T) is defined as the target region. Although we use circles to present the original and target regions, the contour of morphed areas could be curves or other irregular shapes. Because a complex morphing operation can be divided into independent basic morphing operations, we focus only on these basic morphing patterns.

As shown in Fig. 3.7(a,b), growth and contraction are the most common patterns. Here, I completely belongs to T with respect to the growth pattern. Conversely, T would completely belong to I in a contraction pattern. A cross pattern, as shown in Fig. 3.7(c), means that I and T overlap. When $I \cap T = \emptyset$, as shown in Fig. 3.7(d-f), this diffusion model is inefficient. Finally, for the pattern in Fig. 3.7(d), it is difficult to ensure that the diffusion animation fulfills the smoothness constraint when gaps exist between I and T. For the examples in Fig. 3.7(e, f), the energy constraint is not satisfied because the morphing is from empty to T or from I to empty. Therefore, an optimized diffusion model is required to achieve smooth morphing.

Our approach for improving the diffusion model is to add some suitable seeds into I and T to make sense of the $I \cap T \neq \emptyset$ throughout the morphing process. By adding the helper seed, each pattern in the bottom of Fig. 3.7 can be divided into two patterns that match the top patterns in Fig. 3.7. For example, if we add two helper seeds called e_i and e_t to the moving pattern region (Fig. 3.8a), we obtain two sub-patterns as shown in Fig. 3.8(b). Because these sub-patterns belong to common patterns such as contraction and growth, the morphing process will work well. The seeds required are normally quite small; hence, their visual influence on I and T is negligible.

Because the highest density area normally plays an important role in visualization, helper seeds could be generated in the saliency region of the density map, par-

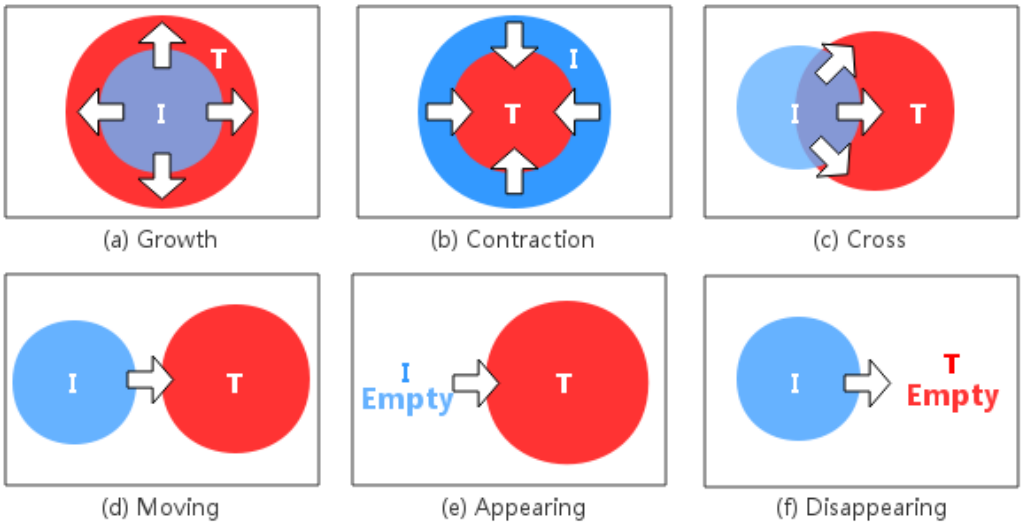
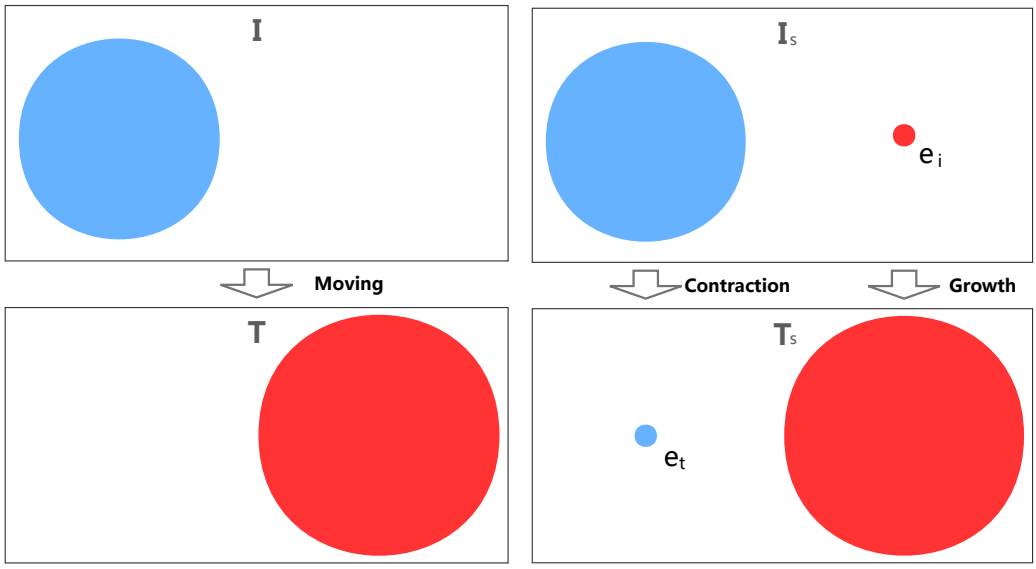


Figure 3.7: Six morphing patterns. The blue region indicates the area in the original density map. The red region shows the area in the target density map. A complete morphing procedure for two density maps normally includes various morphing patterns.

ticularly in the peak-value pixels. We adopt the *steepest descent* (SD) method [150] (which is also known as gradient descent) to detect the peak pixels in a density map. We define the steepest descent method as a function $SD(X)$, where X is a density map and the output is a set of detected peak pixels. $E(SD)$ indicates a sparse density map that only contains the pixels in SD. In addition, we use β as a free threshold (with an empirical value of 0.6) to control the saliency regions. We assume that pixel values larger than β are in a saliency region. We define $Sr(X, \beta)$ as a density map that only contains the saliency regions of a density map X. Hence, the helper seed adding operation on I and T can be defined as follows:

$$\begin{cases} I_s = I + E(SD(Sr(T, \beta))) \\ T_s = T + E(SD(Sr(I, \beta))) \end{cases} \tag{3.11}$$

We define the density maps with helper seeds as I_s and T_s. Fig. 3.9 shows a result of peak pixel detection on a density map using the steepest descent method [150]. Fig. 3.10 shows six morphing results using our seed-based morphing model (I_s and T_s are inputs).



(a) Original I and T.

(b) New I_s and T_s with helper seeds.

Figure 3.8: After adding the seeds (e_i and e_t), the moving pattern has been divided into the contraction pattern and the growth pattern. Hence, the morphing process from I_s to T_s becomes feasible.

Overcoming density nonconformity

By adding the helper seeds, the diffusion model may still suffer from the *density nonconformity* (DN) problem, as shown in Fig. 3.11(a). DN means that the final morphing sequence will not match the target density map (T) using only limited calculation iterations. As shown in Fig. 3.11(a), the DN problem makes it difficult to obtain satisfactory morphing results. Figure 3.11(c), left, shows a magnified result.

DN is another weakness of the diffusion model. The reason for why DN occurs is that the *accuracy constraint* (AC) is not fulfilled for some pairs of density maps. The AC means that the variation in energy $E_{ve} = \sum_{k \in P} |i_k - t_k|^2$ should nearly equal 0, where P is the pixel set of the density map and i_k and t_k represent the density values of pixel k in I and T, respectively. Figure 3.11(a) shows a pair of density maps that will not fulfill the AC. The morphing result generated using the diffusion model is consequently distorted, as shown in Fig. 3.11(a). Hence, an improved method is required to overcome the DN problem.

In linear blending, although distortion and ghosting appear frequently in the mor-

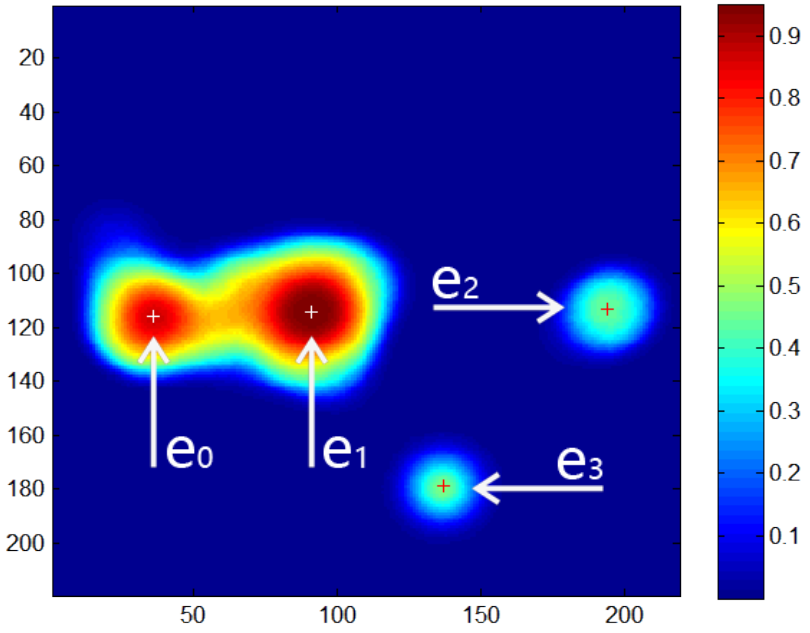
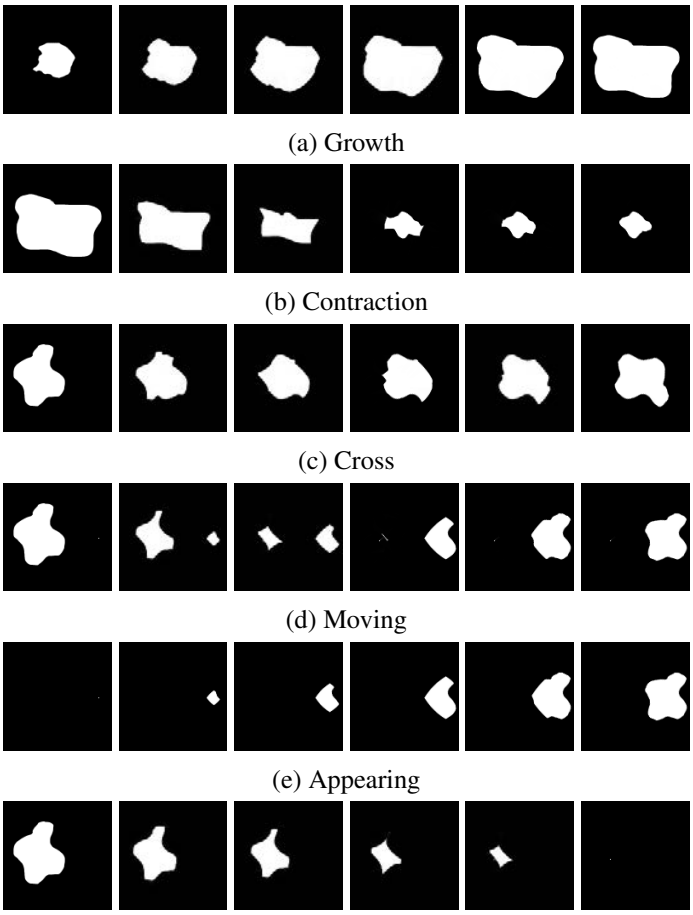


Figure 3.9: A result of peak pixel detection on a density map using the steepest descent method [150]. e_0, e_1, e_2, and e_3 are the detected peak pixels.

phing process, the final morphed density values are always close to the target density map. Consequently, we present a hybrid morphing model that takes advantage of linear blending to overcome the DN problem. Combined with the diffusion model and the helper seed method, the improved hybrid model can be formulated as follows:

$$\begin{cases} S_{i+1} = (1 - (\frac{i}{\tau})^{\psi}) S_{i\mathbf{u}_i} + (\frac{i}{\tau})^{\psi} T, S_0 = I_s \\ \mathbf{u}_{i+1} = \mathbf{u}_i + \frac{T_s - S_{i\mathbf{u}_i}}{(\nabla S_{i\mathbf{u}_i})^2 + \lambda} \nabla S_{i\mathbf{u}_i}, \end{cases} \tag{3.12}$$

where S_i indicates the sub-DM of the morphing process, i is the iteration index, I_s is the input density map with helper seeds, T_s is the target density map with helper seeds, $S_{i\mathbf{u}_i} = S_i(\mathbf{x} + \mathbf{u_i}(\mathbf{x}))$, τ is the number of iterations (initiated with 16), and $\psi \in [1, +\infty]$ is a free parameter that is used to adjust the convergence speed to the target density map. The smaller ψ is, the more artificial ghosting will appear. The larger ψ is, the slower a target density map is achieved. $\psi = 2$ achieved satisfactory results in our experiments. By taking advantage of linear blending, the density maps used for morphing will satisfy the AC and will be close to the target density map in the last morphing step. The better results generated via this hybrid morphing model are shown in Fig. 3.11(b). We find that the sixth sequence of the morphing



(a) Growth

(b) Contraction

(c) Cross

(d) Moving

(e) Appearing

(f) Disappearing

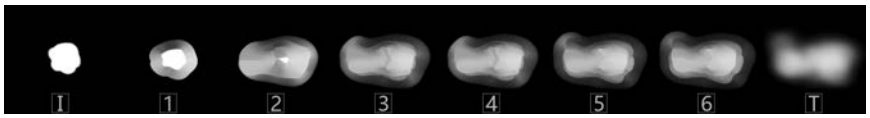
Figure 3.10: Examples of six morphing patterns, each of which includes four sub--DMs. The left column shows the density map of I_s, and the right column shows the density map of T_s. After adding the seeds, all of the morphing processes become feasible.

result, as shown in Fig. 3.11(c), right, is better than using the diffusion model without overcoming DN as shown in Fig. 3.11(c), left.

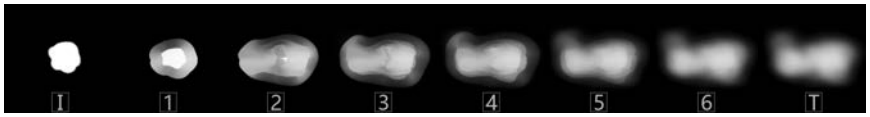
The entire StreamMap algorithm is summarized in Algorithm 1. The input of StreamMap is a pair of point sets, such as F_j and F_{j+1}. The output of StreamMap is S, which is a sequence of smooth in-between density maps.

3.3.3 Trend representation

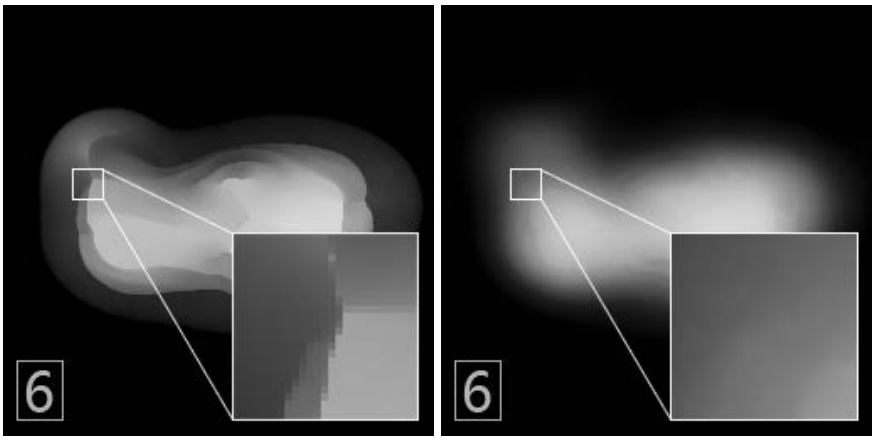
Dynamic smooth morphing, which is similar to a video, helps users notice the obvious trends in the flow of information. We designed a trend representation to improve



(a) Morphing results using the diffusion model [129].



(b) Morphing results using our method with density nonconformity overcoming.



(c) The one on the left is based on the diffusion model, and the one on the right is based on our improved method. Clearly, the result on the left includes artifacts, whereas the result on the right is smooth.

Figure 3.11: Figure showing density map morphing between I and T using two methods. In the final morphing step 6, our result is smooth and more similar to T than using the diffusion model. Hence, our method is able to handle density nonconformity, with minimal artifacts.

user understanding of the variations between two density maps. The data source of trend representation (TR) is based on a variational vector field, which is defined as $\boldsymbol{u}$ in the smooth morphing model, where $\boldsymbol{u}$ is a vector set that presents the instantaneous velocity of each pixel on the density map. Each iterative calculation between two density maps in the morphing model will generate an updated $\boldsymbol{u}$. Based on the generated vector field data, we designed an arrow-based representation called TR to emphasize the trend variation. The basic TR design, as shown in Fig. 3.12(a), is similar to the wind and current visualization. We define the basic visual element of TR as a trend representation particle (TRP). Each TRP is represented by an arrow with a special size and direction on a density map. We define c_{si} as a sampling interval pixel number that defines the distance between neighboring TRPs. Fig. 3.12(a) and

Algorithm 1 StreamMap algorithm

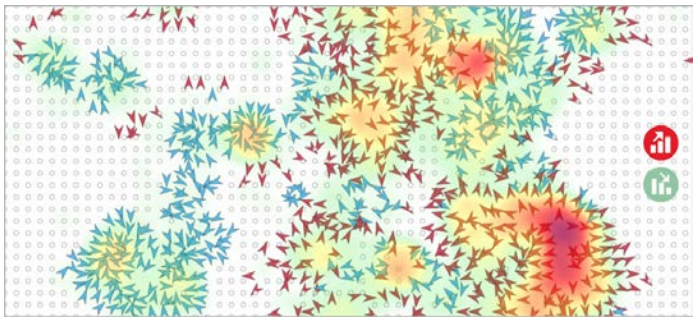
procedure STREAMMAP(F_j, F_{j+1})
  $D_j \leftarrow SKDE(F_j)$
  $D_{j+1} \leftarrow SKDE(F_{j+1})$
  $I \leftarrow D_j$
  $T \leftarrow D_{j+1}$
  $I_s \leftarrow AddSeedI(I, T)$
  $T_s \leftarrow AddSeedT(I, T)$
  $\mathbf{u_0} \leftarrow \mathbf{0}$, $S_0 \leftarrow I_s$, $i \leftarrow 0$, $\lambda \leftarrow 0.4$, $\tau \leftarrow 16$, $\psi \leftarrow 2$
  while $i \leq \tau$ **do**
    $S_{i+1} \leftarrow (1 - (\frac{i}{\tau})^{\psi})S_{i\boldsymbol{u_i}} + (\frac{i}{\tau})^{\psi}T$
    $\boldsymbol{u_{i+1}} \leftarrow \boldsymbol{u_i} + \boldsymbol{\delta}(\boldsymbol{u_i}, I_s, T_s)$
    $i \leftarrow i + 1$
  end while
  return $\boldsymbol{S} = \{S_0, S_1, ..., S_\tau\}$
end procedure

Fig. 3.12(b) show two TR results with different c_{si} values. In addition, as shown in Fig. 3.12, a red TRP means an increasing trend, whereas a cyan TRP indicates a decreasing trend. A white circle indicates no variation in the related region. The size of the TPR indicates the intensity of the variation.

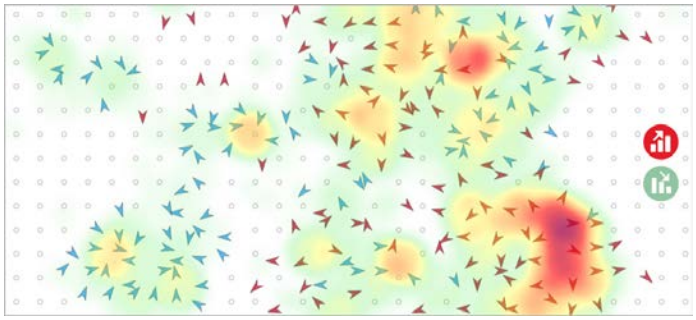
We further improved the TR design by presenting a non-linear trend representation (NTR) method, which enhances the visualization of the salient trend variation and accelerates the rendering by reducing the number of explicit TRPs. We adopt the density map to create an NTR distribution to enhance the trend content. The rules of NTR distribution on the density map are defined as follows:

(1) We define $\boldsymbol{u}$ as a variation vector field between two density maps, I and T. Then, $\boldsymbol{d} = T - I$ is defined as a difference density map.

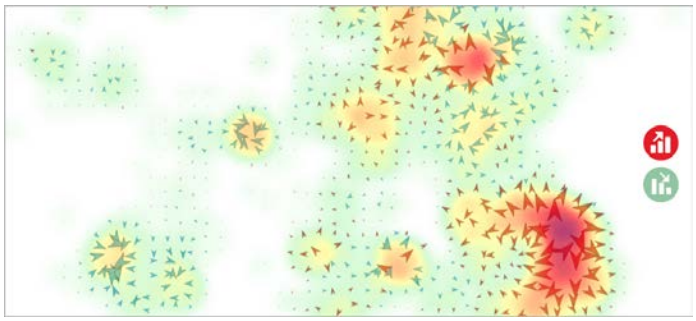
(2) The sampling interval $c_{si} = (1 - d_{avg})(\mu - \nu) + \nu$ is calculated, where d_{avg} is the average density of I, μ indicates the upper bound of c_{si}, and ν defines the lower bound. In our experiment, we set μ to 30 and ν to 10. TRPs will be assigned to a pixel on the density map one by one according to the sampling interval c_{si}. Figure 3.12(a) shows an example of a TRP distribution with $c_{si} = 10$.



(a) Uniform distribution of TRP ($c_{si} = 10$)



(b) Uniform distribution of TRP ($c_{si} = 20$)



(c) Non-linear distribution of TRP

Figure 3.12: Trend direction representation.

(3) We define $\lambda(\|\boldsymbol{u}_{id}\|)$ to determine whether a TRP, generated in the second step, will be shown on the density map.

$$\lambda(\|\boldsymbol{u}_{id}\|) = \begin{cases} 0, \|\boldsymbol{u}_{id}\| < \theta \\ 1, \|\boldsymbol{u}_{id}\| \geq \theta \end{cases} \tag{3.13}$$

In Eq. 3.13, id is the sampled pixel's id, 0 means hiding the TRP on the density map, and θ is a threshold. We assign θ with 0.001 in our experiments.

(4) The size of a TRP is calculated according to $\|\boldsymbol{u}_{id}\|$.

(5) The color of the TRP is determined by the value of $\boldsymbol{d}$. When $\boldsymbol{d}_{id}$ is positive, we assign a red TRP; otherwise, we assign a cyan TRP.

Following the rules presented above, we visualize the NTR as shown in Fig. 3.12(c), which enhances the salient region and improves the performance.

3.4 Use Cases

This section demonstrates usages for StreamMap. We show how to apply our approach to three large stream datasets. The first dataset is an artificial dataset, while the other two are real-world datasets. Tab. 3.1 shows a summary of the experimental datasets. The stream processing framework used in this work is VALID [74]. All visualizations are implemented using JavaScript and the D3 [21] and Leaflet [72] libraries. All experiments were run using a Google Chrome browser on a MacBook Pro with Intel Core i7 2.5 GHz CPU, 16 GB RAM, and Intel Iris Pro integrated graphics.

To help the user explore and understand the streaming points easily, StreamMap is visualized as a stream animation. A stream animation is designed to explore data sequentially, similar to a video player, according to a time step. The user can switch to each processed frame. To enhance the animation of the density map, we adopt color mapping to indicate the different density scales in a sequence. The color range is divided into a distribution of "warm" and "cool" colors in which warm colors, such as dark red, are selected to indicate high-density areas and cool colors, such as cyan and blue, are used to indicate low-density areas. The contrast between the warm color and cool colors can draw users' attention appropriately, as reported in [66].

To reduce the computational complexity of SKDE, we can aggregate points in the same position as a point with a grayscale value in advance. Furthermore, because it is time-consuming to estimate the densities of large quantities of points at the same time, we were able to improve the speed of the SKDE calculations by rendering different sizes of pre-rendered Gaussian kernel images. Each Gaussian kernel image will be blended with an alpha value (0.1 in our experiments) in the final implementation to accelerate the density estimation.

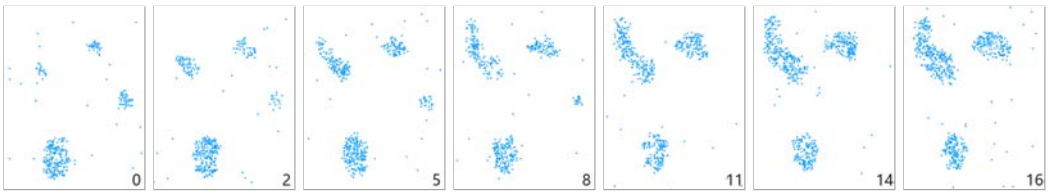
Table 3.1: Summary of datasets used in experiments.

Dataset	Records (Millions)	Period (Date)	Data Interval	Elements
Artificial Data (AD)	93.6M	07/2016-07/2016	6 seconds	Point
People Crowd (PC)	189.3M	07/2015-08/2015	1 minute	People Location
Air Pollution (AP)	11.2M	03/2016-04/2016	1 hour	Location, AQI

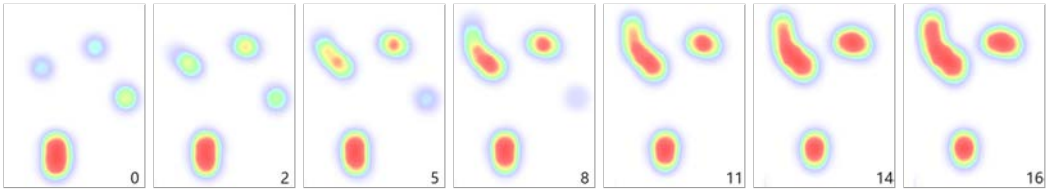
3.4.1 Artificial Data (AD)

We use Perlin noise (PN) [106] to artificially generate a variety of time-varying point sets to test and evaluate the StreamMap method. PN is frequently used to create natural object surfaces in the field of computer graphics; however, we use it here to make the two-dimensional testing points more realistic than random points.

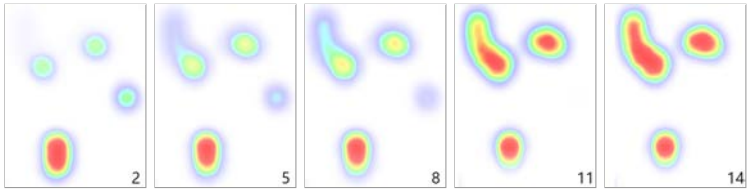
Figure 3.15 shows a comparison of the clustering results of a large-scale AD frame. We find that SKDE can achieve higher accuracy result than using the k-means method, as shown in Fig. 3.15(c). Figure 3.13 shows an example of the morphing result (some of the morphing sequences are selected). Figure 3.13(a) shows sequential point sets generated using the Perlin noise method. We applied the SKDE method to $frame_{0-16}$ to create DM_{0-16}, as shown in Fig. 3.13(b). In this case, we apply the morphing model to a pair of generated density maps: DM_0 and DM_{16}. The remaining DMs (from DM_1 to DM_{15}) are used as benchmarks to evaluate the effect of the morphing model. Our morphing results (sub-DMs) are shown in Fig. 3.13(c). The results are smooth and contain less artificial ghosting compared with the result using the linear blending method, as shown in Fig. 3.13(d). In addition, our morphing result is similar to the benchmarks, as shown in Fig. 3.13(b). We highlight a smooth morphing density map with the corresponding frame in Fig. 3.14(a), which demonstrates that the morphing results closely match the point sets. In Fig. 3.14(b), we highlight the artifacts generated using the linear blending method.



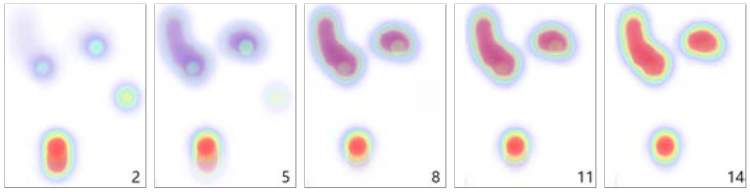
(a) Frames generated using Perlin Noise method.



(b) Density Maps (DM 1-15 are benchmarks).



(c) Morphing results using our method (DM 0 is I and DM 16 is T).



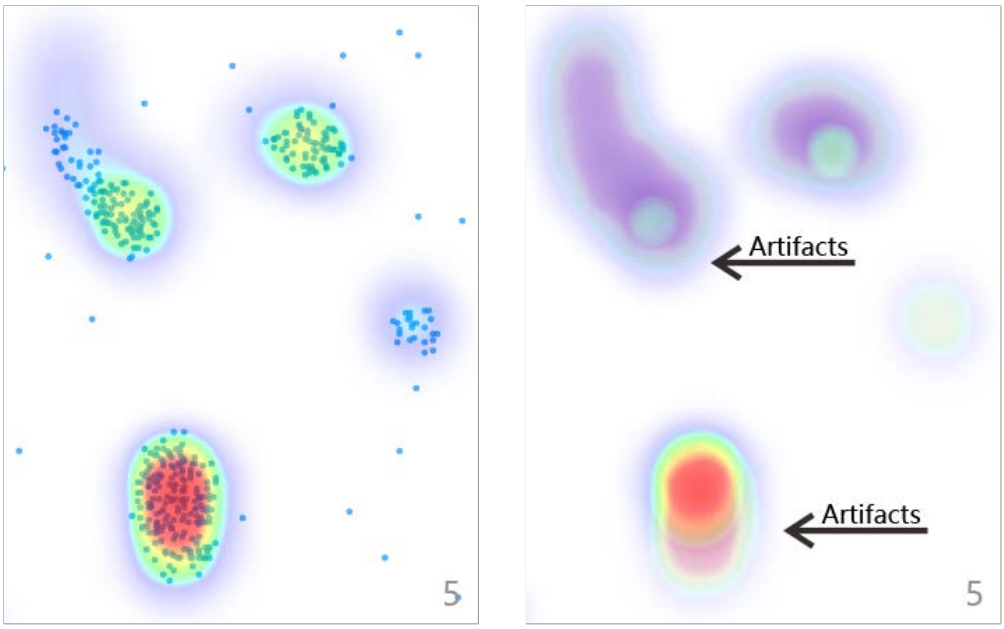
(d) Results using Linear Blending (DM 0 is I and DM 16 is T).

Figure 3.13: Estimated density maps according to the artificial data (AD).

The similarities between benchmarks and sub-DMs will be calculated to evaluate the morphing effectiveness as presented in Sec. 3.5. When sub-DMs are similar to benchmarks, we consider that the morphing model has achieved a good result.

3.4.2 People Crowd (PC)

Our method can easily be applied to visualize the flow of people. We collected a set of locations of people in the center of Shanghai City from the Easygo website. The dataset was collected per minute over ten days. To demonstrate the performance of StreamMap, we integrate the people locations from one hour into a frame and then generate a total of 24 frames to create one full day of data. Each frame will be converted to a density map using SKDE. Figure 3.16(b) shows a density map generated using the SKDE method with adaptive kernel sizes. Figure 3.16(c-d) shows



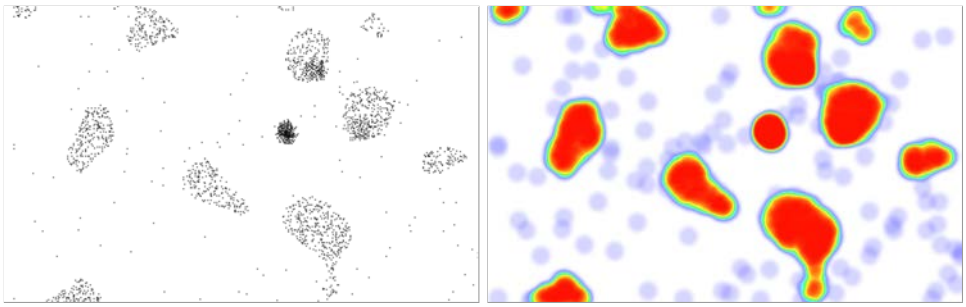
(a) 5th in-between DM merged with frame 5 (Our method).

(b) 5th in-between DM (linear blending).

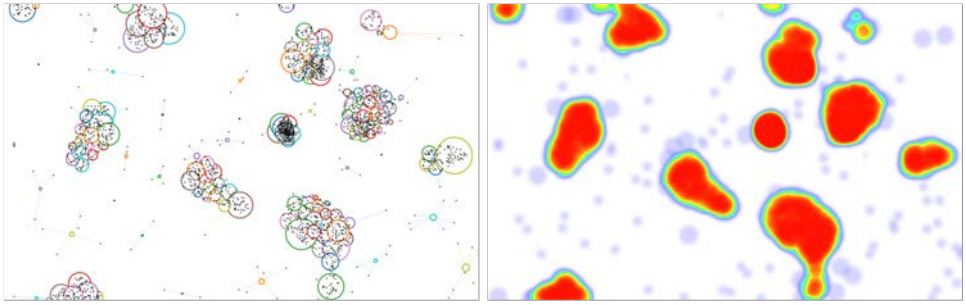
Figure 3.14: The morphing result according to the artificial data (AD).

the results of the KDE method with coincident kernel sizes. Figure 3.16(c) (KDE with a small kernel size) shows unclear salient regions, whereas the result will appear over-estimated, as shown in Fig. 3.16(d), when the kernel size is large. Compared with the KDE results, the SKDE method automatically assigns a suitable kernel size for the detected clusters to avoid obtaining artificial results. The SKDE method also avoids the problem of having to manually adjust the kernel size frequently to estimate the density of streaming data. Consequently, the most crowded regions at different time periods can be accurately estimated using StreamMap. For example, we can clearly find the most crowded regions at the center of Shanghai in Fig. 3.16(b). In addition, the estimated crowded regions are more independent than those when using the KDE method. It is easy to enhance the variety when two density maps used in a morphing process involve independent regions.

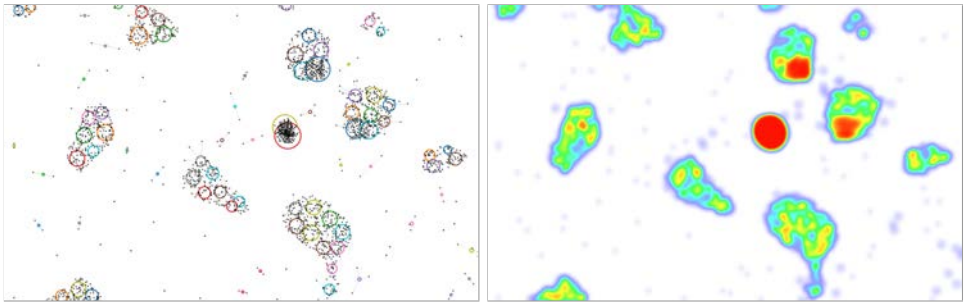
In addition, StreamMap provides a smooth crowd flow movement; thus, a user can easily detect the trends of the people flow. Figure 3.17 shows smooth morphing results on 25 July 2015 at East Nanjing Subway Station (the left rectangle region in Fig. 3.16a). In Fig. 3.17, we observe that people are moving toward the southwest exit



(a) Original point data and corresponding density map using KDE.



(b) K-mean clustering result and corresponding density map using KDE.



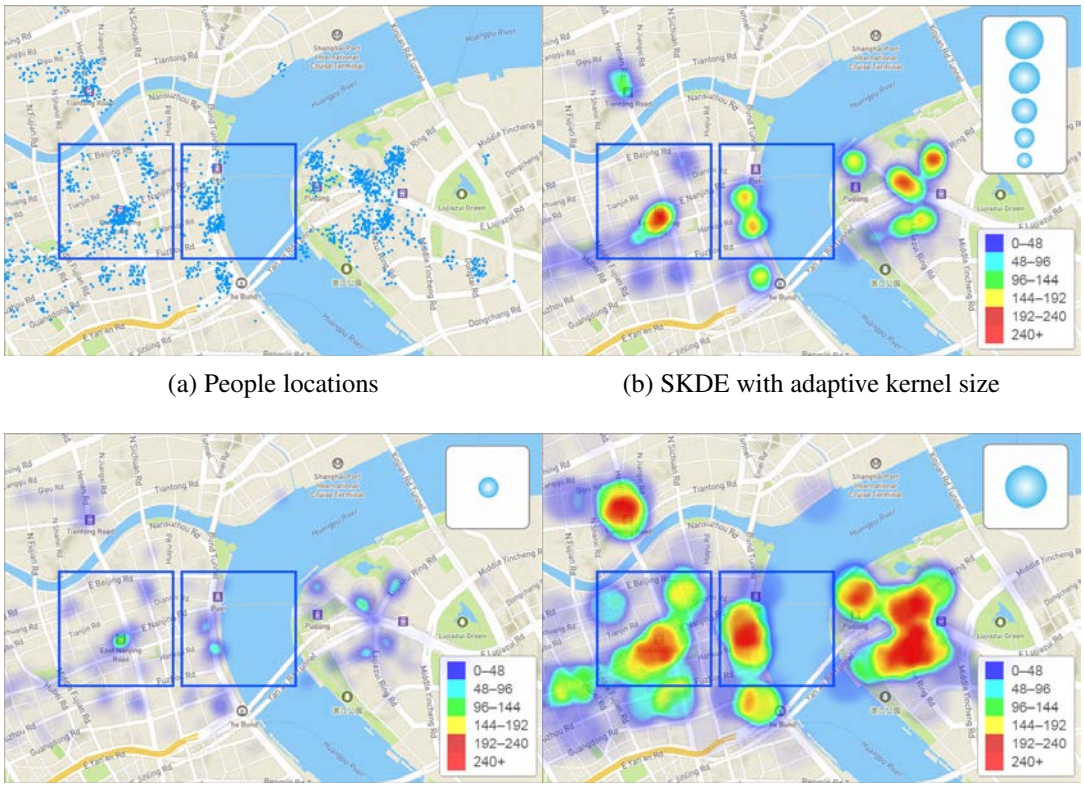
(c) SG clustering result and corresponding density map using SKDE.

Figure 3.15: A comparison of the clustering results of a large-scale AD frame.

from 13:00 to 14:00. Moreover, there are two different crowds of people that appear in the northeast and southeast. From 21:00 to 22:00, the flows of people at East Nanjing Subway Station are stable, whereas people at the northeast are increasing.

Figure 3.17 also shows the crowd variations of people at the Bund (the right rectangle region in Fig. 3.16a), which is a famous attraction in Shanghai. There are two crowds of people at the Bund; one is stable and the other is decreasing from 13:00 to 14:00. From 21:00 to 22:00, the upper people crowd at the Bund is decreasing and the lower one is increasing.

Because the center of Shanghai includes many tourist attractions, knowing the crowd situation around the different destinations can help tourists design their travel



(a) People locations (b) SKDE with adaptive kernel size

(c) KDE with small kernel size (d) KDE with large kernel size

Figure 3.16: Estimated density maps according to the collected PC data between 1:00 PM and 2:00 PM on 25 July 2015.

plans and avoid congestion at peak hours. Similarly, knowing the directions in which moving crowds diffuse could help urban designers optimize routes and traffic flows.

3.4.3 Air Pollution (AP)

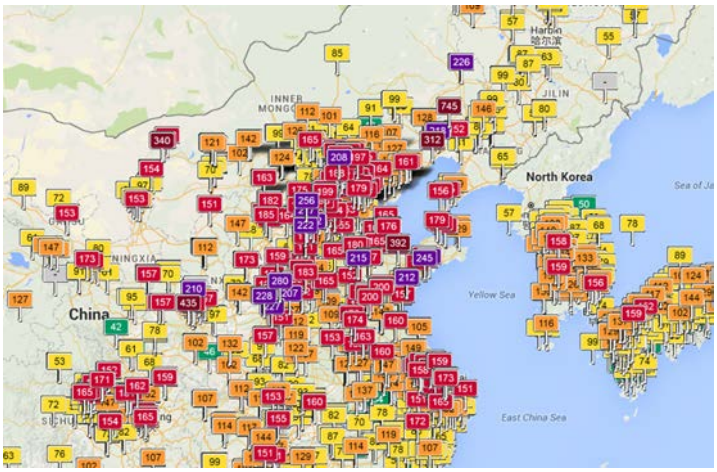
Air quality has severe adverse health effects on a large percentage of the population as the air quality index (AQI) increases. The AQI is used to indicate how polluted the air is. Air pollution in one area may affect neighboring areas. There are nearly eight thousand air quality monitors in the world. Each record from each monitor is visualized as a colored flag at aqicn.org. However, the current air pollution visualization tool at aqicn.org suffers from the overlapping problem, as shown in Fig. 3.18(a). In addition, dynamic representation of AQI data is a difficult challenge.

Using the StreamMap method, we can generate smooth air pollution diffusion

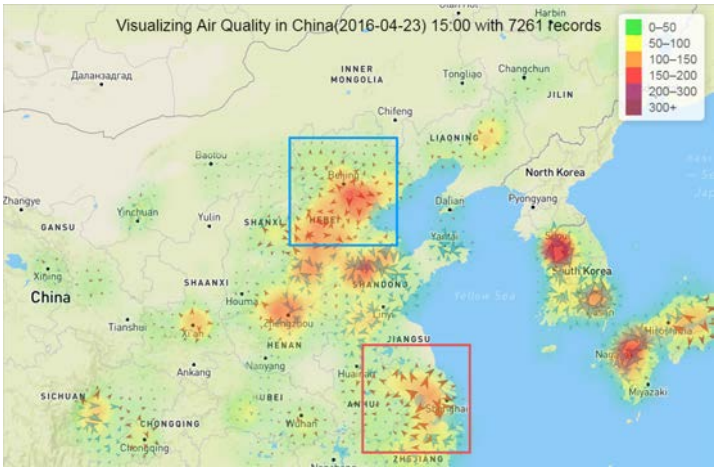
Figure 3.17: Morphing results of the people flow visualization between two time steps. The left and right columns are the input density maps. In-between four columns are transition sequences selected from the iterative morphing operation.

animations to aid viewers in understanding the distribution and variation of air pollution. We collected the AQI records from aqicn.org every hour over a forty-day period from 12 March 2016 to 26 April 2016. The basic elements in each AQI record are the monitor location and the AQI value. Figure 3.18(a) presents an example of the distribution of air quality monitors in China. Using the StreamMap model, we can visualize the streaming AQI data as continuous density maps superimposed over a geographical map.

As shown in Fig. 3.18(a), it is difficult to obtain useful sequential information from the static flags because they overlap. When the display is small, the overlapping problem will be more serious. In addition, if the flags are directly browsed frame by frame, then the transformation between frames is not smooth. Figure 3.18(b) shows the air pollution density map estimated from Fig. 3.18(a) through SKDE. Figure 3.18(b) also shows the air pollution diffusion trend. From the blue rectangle in Fig. 3.18(b), users can easily observe that the air pollution in Beijing is increasing and will diffuse to the surrounding areas. The red rectangle shows the increasing air pol-



(a) Original AQI visualization visualized at aqicn.org.



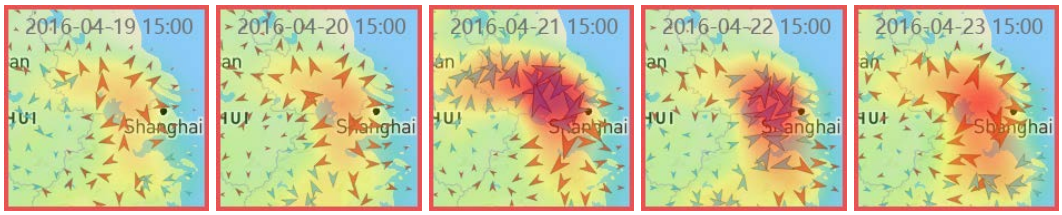
(b) Improved AQI visualization with density map and trend representation.

Figure 3.18: Comparison of two air quality visualization methods.

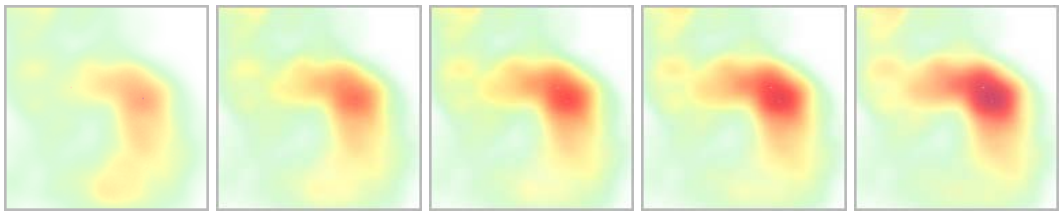
lution trend in Shanghai. The pollution from Shanghai may affect Anhui Province to the west of Shanghai. Fig. 3.19(a) shows 5 trend representations at East China at different time steps, which help convey the air pollution density, the density variations, and the diffusion directions. Fig. 3.19(b) shows a variety of smooth air pollution sub-DMs. Both smooth sub-DMs and trend representations can help users notice an obvious distribution of the air pollution data.

3.5 Evaluation and Discussion

Fig. 3.20 shows the average SKDE time cost in our test. The size of the generated density maps in the tests was 1280×960. The inputs for SKDE are the high-density



(a) Trend representations at East China from 19 April 2016 to 23 April 2016.



(b) Smooth morphing results at East China between 20 April 2016 and 21 April 2016.

Figure 3.19: Air pollution visualization using StreamMap.

random 2D points over the fixed area. The generation of these random 2D points was discussed in Sec. 3.4.1. For each point density, we generated ten random frames for the performance tests. K is set to 300 in the SKDE performance test. As shown in Fig. 3.20, SKDE with the superpoint method can finish on average in near real time. Moreover, we found that as the point number increases non-linearly, SKDE shows a nearly linear increase in computation time. Compared with the slower k-means method, the SKDE with the superpoint method achieves higher performance, thus making SKDE suitable for large-scale and high-density point visualizations.

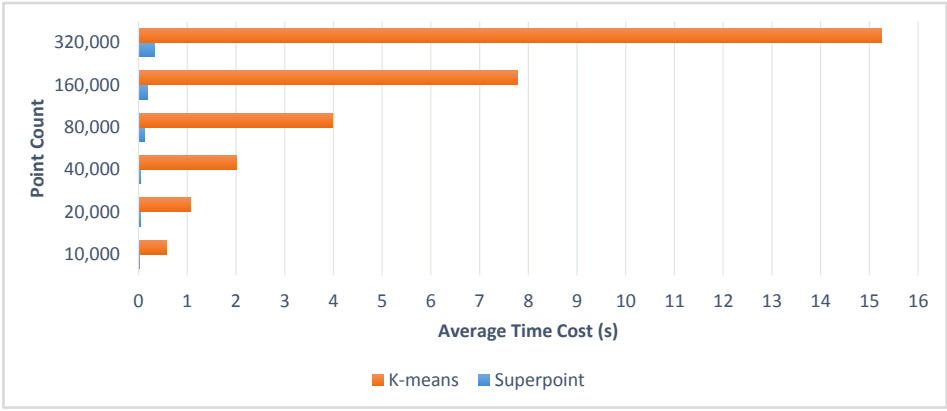


Figure 3.20: Time cost of the SKDE method with different data sizes (the dataset used is AD as shown in Sec. 3.4.1).

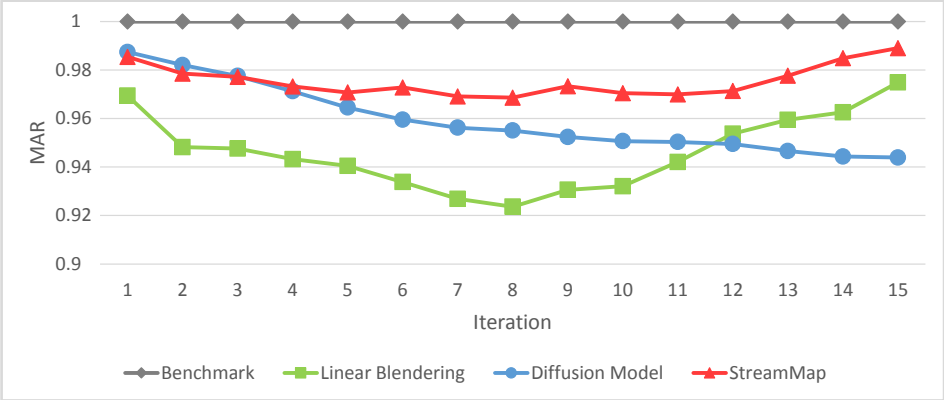
We also computed the structural similarity (SSIM) to evaluate the morphing effectiveness. SSIM was used as a structure similarity measurement between two images,

as discussed in the work of Wang et al. [147]. SSIM is more consistent with the visual perception of a human than peak signal-to-noise ratio (PSNR) [149]. More similar density maps achieve higher SSIM scores. We selected 170 continuous frames from the AD dataset in Sec. 3.4.1 as the testing dataset to evaluate the morphing effectiveness. SKDE was used to generate sequences of density maps according to the selected frames. We defined 17 density maps as a group, in which the first and the last ones are the inputs of the smooth morphing model (I and T). We used the remaining ones as benchmarks. Overall, our testing data included 10 groups. For each group, 15 sub-DMs were generated using the morphing methods from 15 iterations. We define two measurements to evaluate the effectiveness of the morphing process according to the selected testing data.

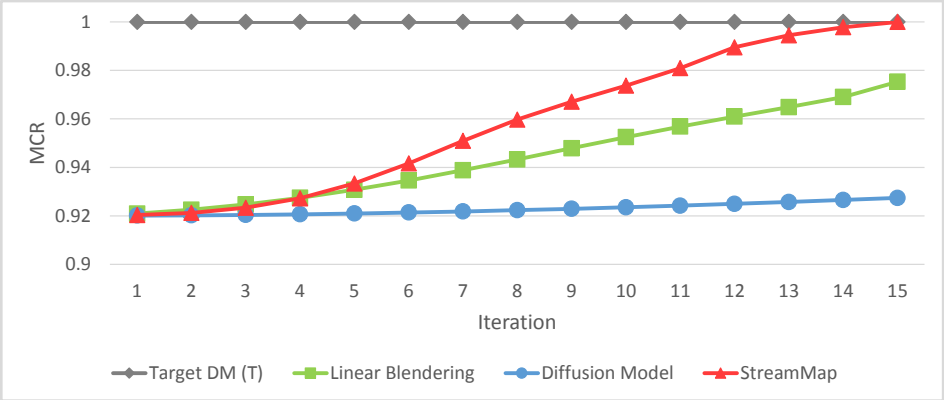
For the first measurement, we define the SSIM of an in-between density map and a benchmark density map as a *morphing accuracy rate* (MAR). MAR is used to evaluate the morphing accuracy. Figure 3.21(a) shows the MAR results obtained using three different methods: linear blending, diffusion model [129], and our method. As these results show, our smooth morphing results closely match the benchmarks.

Another measurement is called morphing completion rate (MCR), which defines the SSIM of an in-between density map and a target density map (T). MCR is used to evaluate whether a morphing operation will be performed in finite iterations. If an in-between density map is similar to a target density map (T), then the MCR will be close to 1. Figure 3.21(b) shows the MCR results obtained using three different methods. These results show that our method achieves the best MCR and that the diffusion model method requires more iterations to finish the morphing.

Measurements of MAR and MCR show that our method achieves a better morphing effectiveness than the other two methods. StreamMap is also user-friendly because the smooth morphing is finished in 16 iterations, whereas the diffusion model [129] suffers miss-convergence in 16 iterations.



(a) Comparison of the morphing accuracy rate: SSIM values of a benchmark DM and an in-between DM generated using different methods.



(b) Comparison of the morphing completion rate: SSIM values of a target DM (T) and an in-between DM generated using different methods.

Figure 3.21: Comparisons of the morphing effectiveness using two measurements.

3.6 Conclusion

This chapter presents a new method for dynamically visualizing high-density streaming points called StreamMap. After a comprehensive overview of the related work, such as scatterplots and linear blending, we show how these techniques lead to the significant problem of sudden sharp changes and ghosting occurring in dynamic visualizations. Then, we present the SKDE method to adaptively cluster the high-density points into regular density maps. We present a novel diffusion-based algorithm to implement smooth morphing between two estimated density maps. Finally, we introduce a method of trend representation that can enhance the visualization of StreamMap. The experiments demonstrate the scalability of our method. For visual analysis, changing patterns can easily be detected through StreamMap's visualiza-

tions.

A future direction of exploration is to visualize the evolution of high-density feature regions to provide an overview of long-period streaming data. High-density features in different frames will be generated by StreamMap, and the visualization could be further complemented with Sankey flow diagrams, as shown in the works of Sebastian et al. [121] and Landesberger et al. [140]. Many other applications can easily be configured to work with StreamMap, such as crowding effects in congested public urban transportation systems. Andrienko et al. [6] presented a good method tailored for visualizing and analyzing trajectories concerning routes of people. We believe that our method could be combined with the method of Andrienko et al. [6] to address streaming trajectory data in the future.

CHAPTER 4

MODULE-BASED LARGE-SCALE GRAPH VISUALIZATION

4.1 Introduction

Graph connectivity patterns allow us to discover and isolate points of interest in massive networks which are otherwise difficult to spot visually on current displays. However, the effect of large network visualization is often limited to the size of the display [178]; thus directly visualizing them will cause an overlapping problem.

The primary objective of large network visualization is the understanding of global and local patterns in dynamic graphs such as connectivity bundles, clustering, boundary formations, and expansions. Zinsmaier and his colleagues [178] are inspired by the Level-of-Detail (LOD) techniques in the computer graphics discipline and introduce a straight-line graph drawing that can be rendered interactively with different levels of detail to visualize large-scale graphs. Because display systems have finite area and are physically limited both in size and spatial dimensions for visualizing large graph structures, a multi-screen solution is adopted in [27]. However, the multi-screen approach imposes restrictions on spatial layouts and interactions.

A practical approach to displaying a large-scale network is partitioning the network according to well-defined domain-dependent attributes. However, graph visualization in the presence of incomplete information is an open challenge and applications in this area can be found in abundance. To better visualize and understand patterns in large-graph discovery, we focus on the representation of local patterns. This is a critical step in deciding the structural components of a graph visualization.

For the representation of community networks, Blondel et al. [19] and Rosvall [115] discuss in detail *community aggregation*. They aggregate the nodes of

a community into a super-node. A machine learning method, such as *Belief Propagation* [56], is adopted to explore large graphs. In addition, Dunne et al. [39] improve the graph visualization readability by drawing different glyphs. Wu et al. [161] utilize Voronoi maps to enhance the community visualization. Compared with the previous methods cited above, our method focuses on designing a module for anticipating the visual effects of the final visualization layout for large-scale networks.

In this chapter, we present the *Module Graph*, a flexible large-scale graph visualization framework that aggregates the community of large graphs into modules. We first address the problem of module-based graph visualization by introducing a *graph modularity measure*. Second, a graph simplification method is adopted to accelerate the module detection. Third, each graph pattern in the module is analyzed using the k-clustering method to enhance the module visualization. Fourth, the visualization of *Module Graph* is constructed by (a) building a module-based graph that indicates the basic structure of the original graph and (b) identifying the sub-graph patterns. We use a symbolic signature instead of a simple node in the community graph. Finally, our experiments on real social networks and spatial networks show that *Module Graph* can effectively transform large graph data into recognizable patterns and shapes that reveal significant structural and topological information.

The remainder of this chapter is organized as follows. We present the details of our Module Graph approach, including the mathematical model and the visual design, in Sec. 4.2. We describe the experimental study to demonstrate the effectiveness of the presented approach in section 4.4. In Sec. 4.5, we further discuss the performance of our methods. Finally, in Sec. 4.6, we conclude our work and present future research directions.

4.2 Module Graph

We define the visualization of large-scale networks as a problem of module detection and clustering. Figure 4.1 is a simple example of module detection. The input of our

framework is the network data and the corresponding number of edges and nodes. First, we aggregate the nodes into modules through modularity measuring. Second, we abstract the features of detected modules and assign the modules with different patterns using a k-clustering method. In this step, we design five patterns to approximatively represent the structure of the module. Finally, the detected information is visualized according to the module design.

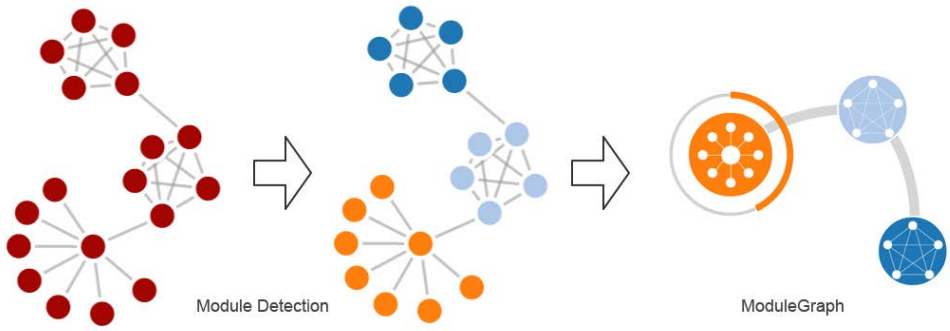


Figure 4.1: A simple example of module detection and Module Graph.

Our method, called *the Module Graph*, can be viewed as an abstraction of a large graph. The right part of Fig. 4.1 is an example of Module Graph. The topology of a large graph is indicated by $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V} = [\mathbf{v}_0^T, \mathbf{v}_1^T, ..., \mathbf{v}_n^T], (\mathbf{v_i} \in \mathbb{R}^2)$ denotes the vertices and $\mathbf{E}$ denotes the undirected edges. The Module Graph is defined as $\mathbf{Mg} = (\mathbf{M}, \mathbf{E})$, where $\mathbf{M} = [\mathbf{m}_0^T, \mathbf{m}_1^T, ..., \mathbf{m}_n^T], (\mathbf{m_i} \in \mathbb{R}^2)$ denotes a set of modules and $\mathbf{E}$ denotes a set of undirected edges, where each edge has a weight of $w = |E|$. Each $\mathbf{m}$ is a vector with several features such as the module pattern, sub-node count, and sub-link count.

Overall, each module can be considered as a community in the graph. Hence, a community detection method called *Modularity Classes*, as described in [99], can be adopted to find the communities from a large graph. The method of community detection is better than the k-means clustering method because the assignment of k is not required in advance. A further improved method for community detection, called the *Louvain*, reduces the calculation of the modularity. This has been shown to be effective for large-scale community detection by Blondel et al. [19]. The process of

finding the modules consists of maximizing the modularity of the linked nodes. The modularity measure Q can be defined as [99]:

$$\mathrm{Q} = \frac{1}{2m} \sum_{i,j} \left[\mathrm{W}_{ij} - \frac{k_i k_j}{2m} \right] \lambda(s_i, s_j) \tag{4.1}$$

where W_{ij} indicates the linking weight of node i and node j; k_i and k_j denote the counts of linked nodes on node i and j, respectively; m indicates the sum of all linking weights; and the function $\lambda(i, j)$ denotes whether two nodes belong to the same community. The range of W_{ij} is $[0.0, 1.0]$. Parameter $\lambda(i, j)$ is set to 0 when node i and node j belong to different communities. Otherwise, $\lambda(i, j)$ is set to 1.

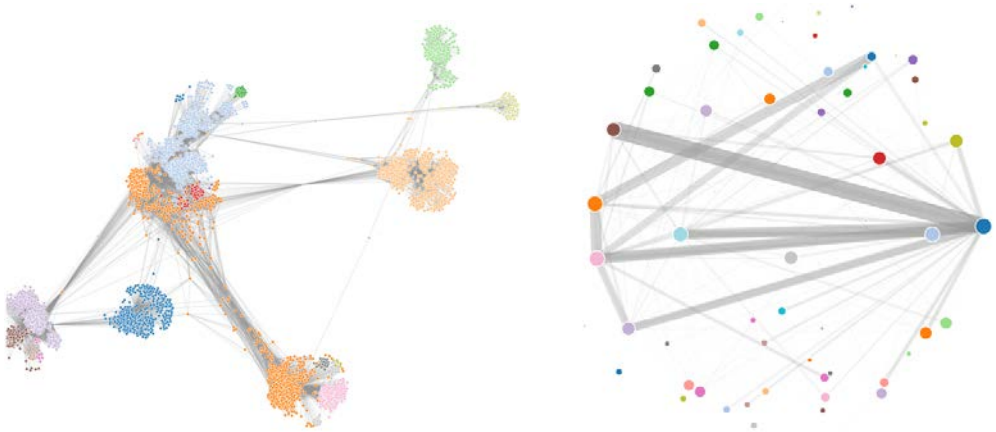
Historically, researchers have paid less attention to community detection on spatial networks. An example of a spatial network is a geographical network such as airlines around the world. We further consider the distance feature for a spatial network by modifying Equation (4.1). The link with a longer length will contribute less modularity. Hence, the improved modularity measure $\mathrm{Q_d}$ can be defined as follows:

$$\mathrm{Q_d} = \frac{1}{2m} \sum_{i,j} \left[\mathrm{W}_{ij} {dist_{ij}}^{-\alpha} - \frac{k_i k_j}{2m} \right] \lambda(s_i, s_j) \tag{4.2}$$

In Equation (4.2), the value of α indicates the network type. The type could be 0 or 1, respectively denoting a non-spatial network or a spatial network. $dist_{ij}$ denotes the Euclidean distance of two nodes in a spatial network. In a non-spatial network, the value of $dist_{ij}$ is 1.

The calculation of *Modularity Classes* [99] for a large-scale graph is time consuming. Although the *Louvain* method [19] achieves a high performance, its precision is lower than that of *Modularity Classes*. We present a hybrid modularity-based method to utilize the two addressed methods. Our method is based on the modularity and is suitable for calculating the modules of a large-scale graph. We can define the hybrid modularity Q_h as follows:

$$Q_h = \begin{cases} Q_d, (c_v + c_l) \leq \varepsilon \\ {Q_d}', (c_v + c_l) > \varepsilon \end{cases} \tag{4.3}$$



(a) Original network (b) Community-based aggregation

Figure 4.2: Visualization of a simplified large network.

In Equation (4.3), c_v denotes the vertex count of the graph, c_l denotes the edge count of the graph, and $Q_d{}'$ is a high-speed community detection measure that removes the function of $\lambda(s_i, s_j)$. Although $\lambda(s_i, s_j)$ removal will slightly affect the accuracy of community detection, it makes sense of the community detection of a large-scale graph. $\varepsilon \in (0, +\infty)$ is a parameter that decides which measure will be used. The setting of ε is performed based on the computer configuration, namely the available CPU and RAM. In our experimental environment, we set ε as 0.1 million to achieve interactive visualization when the input network is a large-scale network. If the experimental computer is more powerful than ours, the user can increase ε to achieve more accurate results.

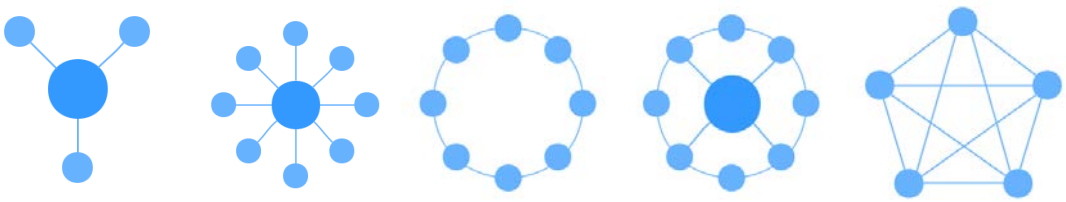
Using the hybrid modularity-based method, we can effectively detect the communities of a large-scale graph, as shown in Fig. 4.2(left). Different communities are assigned different colors. To observe the graph clearly, each community can be replaced with a single node, as shown in Fig. 4.2(right). A simplified community graph allows the user to easily realize the clear structure of the original graph.

When the module sizes become very large, the simplified community graph may remain unsatisfactory on a limited display. Our *Module Detection* method can be extended using the hierarchical module method. A hierarchical module means that the detected communities in the previous detection process can be considered as the

input of a new community detection process.

For hierarchical module detection, we define the iteration of module detection as i, with a default value of 1. Screen width and height are defined as w and h. If the number of nodes in the input network is far greater than ηwh, we increase the module iteration of i until the node count in the new graph is equal to or less than ηwh. η is a free parameter that reflects the blank space of the module visualization in the display. In our experiment, we set η as 0.3, w as 1440, and h as 900.

4.2.1 Pattern definition

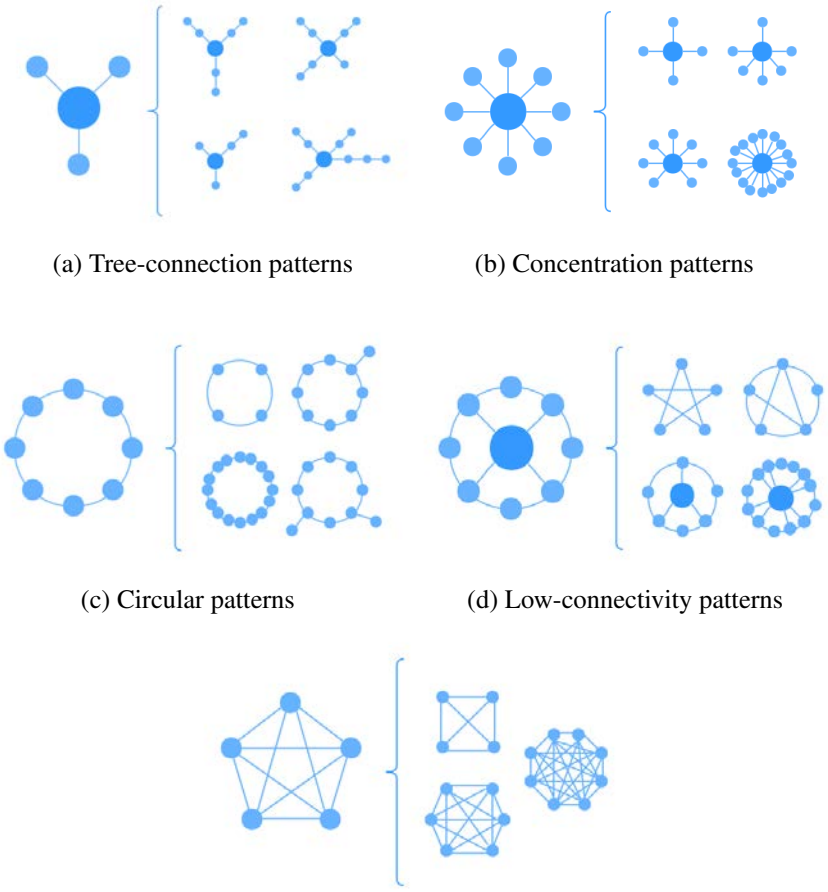


(a) Tree-connection (b) Concentration (c) Circular (d) Low-connectivity (e) Full-connectivity

Figure 4.3: The five types of graph modules.

To further gain insight into modules, we analyze the module patterns. Wernicke [151] presented a method to detect motifs in a large network. Motifs indicate the link patterns in the graph. Unlike motifs, we calculate several pattern factors to analyze each module type according to its internal graph, as well as other significant features such as module size, linking weight, visibility, and occupation percentage. An internal graph represents the structure of the module.

To simplify the module representation, we classify the modules into five agent patterns. The design of agent patterns is according to the most common network structures as presented in network structure theory [164]. Fig. 4.3 shows our designed patterns: tree-connection, concentration, circular, low-connectivity, and full-connectivity. The tree-connection pattern indicates that the graph is extended from a node and that some children of the root still contain the children. The concentration pattern indicates that all children nodes are connected to the root and that each child



(a) Tree-connection patterns

(b) Concentration patterns

(c) Circular patterns

(d) Low-connectivity patterns

(e) Full-connectivity patterns

Figure 4.4: Examples of graph modules: for each example, we show the agent pattern on the left, and example cases on the right.

does not have other connections. The circular pattern indicates that each node in the graph only has two links. The full-connectivity pattern is a complete graph. The remaining structure of the graph will be classified as a low-connectivity pattern. We define the patterns in Fig. 4.3 as the agents to represent other similar structures. We show some cases of similar structures of the agent patterns in Fig. 4.4. The right part of each sub-figure shows some graph cases of the related patterns. The method of mapping the graph to agent patterns will be discussed in Sec. 4.2.2.

4.2.2 Pattern detection via k-clustering

We present a k-clustering method to detect the pattern type of each module. To classify the module into the agent patterns that we have defined, we abstract three features for each module. The feature vector can be defined as $\mathrm{v_f} = [\alpha, \beta, \delta]$. We define the number of connected nodes for a node as c. By following the pattern design, we can denote $\alpha = \frac{\max(e_i)}{\sum e_i}$, $\beta = \frac{count(e_i=2)}{n}$ and $\delta = \frac{1}{n} count(e_i > \frac{n}{2})$, where e_i denotes the neighbor count of node i and n denotes the node count of the module.

We propose a k-clustering method to approximately generate the clusters according to the module feature vector. Each cluster includes the modules with the same pattern. The proposed clustering method includes five steps. First, $\mathrm{v_f}$ is calculated for each module. We define each module as a node in our method and consider $\mathrm{v_f}$ as a 3-dimensional position. Initially, all the nodes are not assigned to any cluster. Second, the cluster count (k) is initialized as 5 because we have designed 5 types of agent patterns for visualization. Third, 5 nodes that match 5 agent patterns are selected, and their positions are assigned as the cluster centers. The selection method can follow the agent pattern definition. Hence, we can obtain 5 initial clusters. Each initial cluster only contains one node. Fourth, the remaining nodes will be assigned to the closest cluster by calculating the distance between its position and the other cluster center. Fifth, the mean position of the cluster's nodes is calculated and defined as the new cluster center.

The fourth and fifth steps should be repeated until the sum of the distances D is minimized. D is defined as $D = \sum_{i=0}^{i<k} \sum_{j=0}^{j<count(i)} dist(p_{ij}, c_i)$, where p_{ij} indicates node j in the cluster i, c_i indicates the cluster center of the cluster i, and $dist$ is the distance calculation function. Figure 4.5 presents an example of 5 cluster distributions in the 3-dimensional space after the k-clustering process is completed. Colored circles denote the cluster centers.

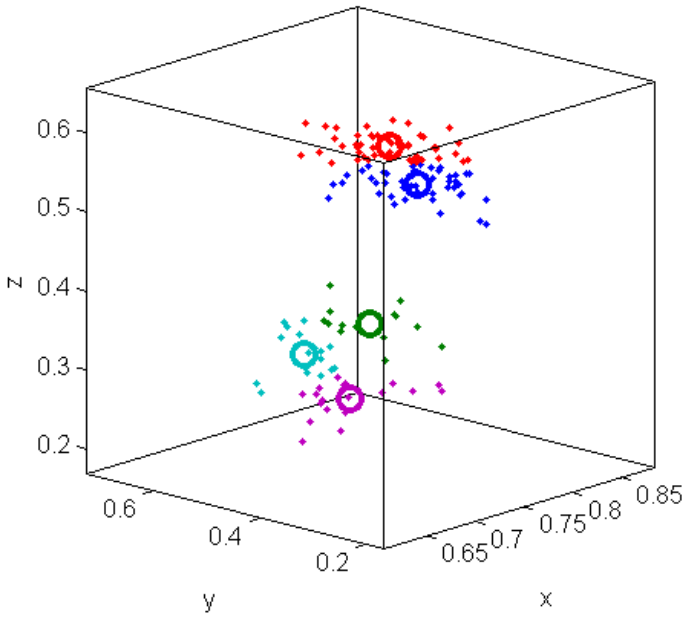


Figure 4.5: An example of k-clustering. The color indicates the pattern type.

4.2.3 Visual design

The basic *Module Graph* layout can be calculated using a direct-force algorithm to ensure that each module will not be overlapping with other modules. The main idea of our visual design for *Module Graph* is to attempt to visualize the specific patterns instead of the crowd nodes in a large graph. The visual elements of each module include a circle with a different radius called m_c, an outside orbit with a different percentage called m_o, an icon called m_p that indicates the module pattern, and a set of edges called m_e that connect the related modules. Each edge between modules has a different width according to the weight. Fig. 4.6 gives a description of the *Module Graph* design.

We choose the color themes from the *Google Color Palette* [50] to represent different modules. Because the color types are limited, the number in the center of the module can be used to indicate the id of the module. When the module graph remains large scale after simplifying the original graph, we design a chord diagram to assist in module visualization. A chord diagram is adopted to present the module relationship and distribution through the circle layout according to the importance weight of the

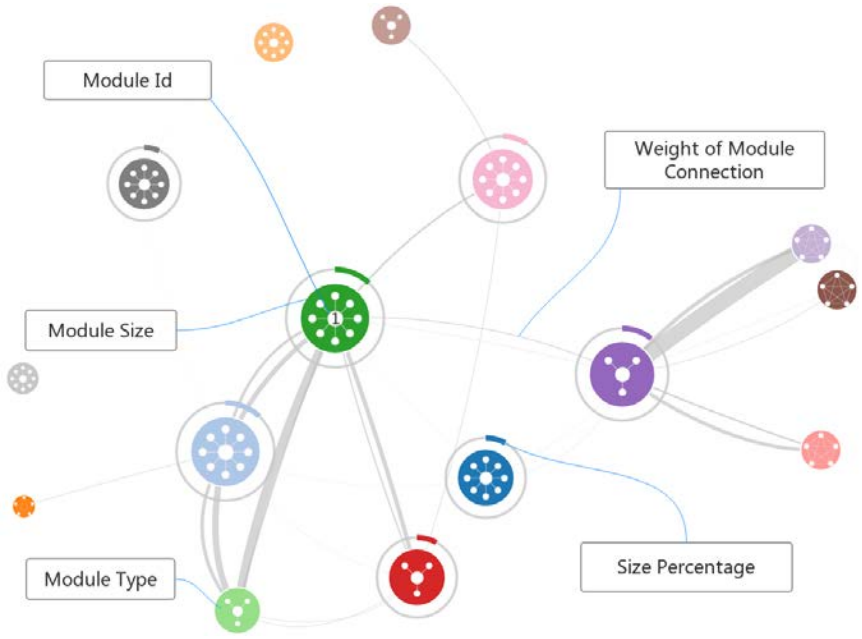


Figure 4.6: Visualizing the *Module Graph* design.

module. The importance weight can be defined according to the neighbor count of a module. The outside circle's color indicates the pattern type. The importance percentage of a module and the relationships among modules can be easily determined via a chord diagram, as shown in Fig. 4.7.

4.3 Super Module Graph

When the graph network data is large-scale and time-varying, it is very difficult to keep the content information stable due to the layout complexity and temporal variation. Feng et al. [44] presented a smooth streaming graph visualization approach to ensure both the temporal coherence and preservation of important content; however, the smooth visualization of a large-scale time-varying graph has not been investigated. Based on the idea of Module Graph, we further present a module-based method for dealing with large-scale time-varying graph visualization that avoids artifacts such as abrupt changes or popping. Since this approach is inspired by a super-graph idea [36], we titled it Super Module Graph (SMG).

The super-graph is a method to aggregate various graphical data together to create

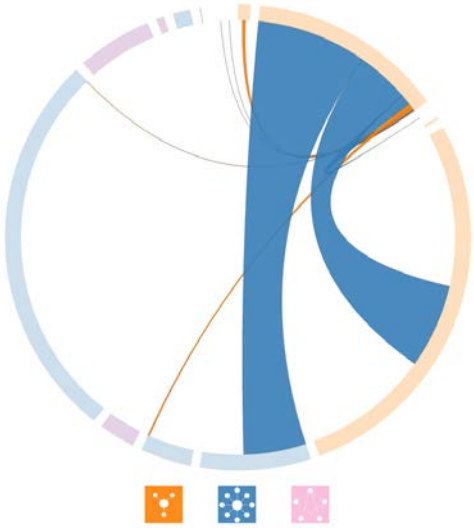
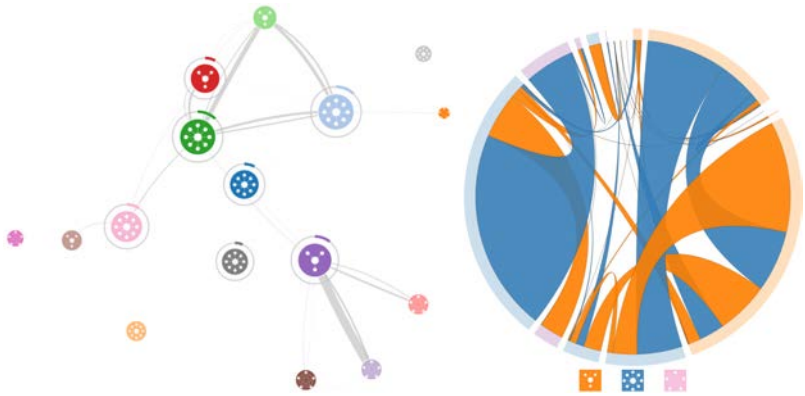


Figure 4.7: The chord diagram representation of Module Graph.

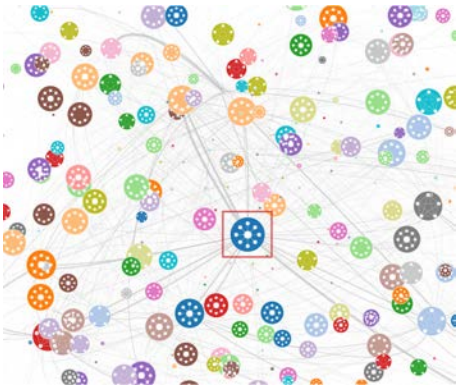
an overview of the whole structure of graphs. We define the streaming graph data at different time steps as $\mathbf{G_t} = (\mathbf{V_t}, \mathbf{E_t})$ and $\mathbf{SG} = (\mathbf{SV}, \mathbf{SE})$ as a super-graph, where $\mathbf{SV} = \{\mathbf{V_1} \cup \mathbf{V_2} \cup ... \cup \mathbf{V_n}\}$, $\mathbf{SE} = \{\mathbf{E_1} \cup \mathbf{E_2} \cup ... \cup \mathbf{E_n}\}$, and $t \in T = [1, n]$. The super-graph not only represents the relationships between different time-varying graph data but also simplifies the graph layout smoothing. Feng et al. [44] used the super-graph approach to smooth the dynamic graph visualization; however, when the super-graph is large-scale, it will exceed the screen display, thus causing an overlapping problem. Hence, we consider applying the community detection method on the super-graph and generating a super community graph. Then, the communities will be assigned for the graph data at each time step according to the super community graph. In the visualization part, we use Module Graph to indicate the graph data at each time step, thus converting a time-varying graph visualization problem into a time-varying Module Graph visualization problem. When the graph communities at different time steps are detected, we calculate their module types to generate a SMG. Since each module in the SMG has a fixed position according to the super community graph, the sudden change of the module variation can be overcome through a linear module blending. Figure 4.9 shows a pipeline of the SMG generation.

The representation of the SMG in our design is a smooth Module Graph anima-

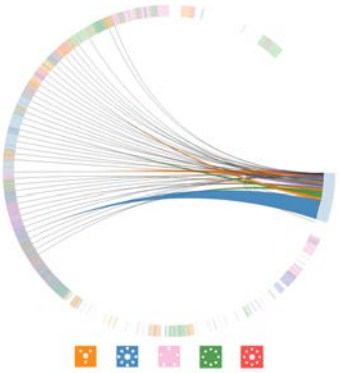


(a) Module Graph of Facebook network.

(b) Chord diagram of Module Graph for Facebook network.



(c) Module Graph of DBLP network.



(d) Chord diagram of Module Graph for DBLP network.

Figure 4.8: Visualizing social networks by Module Graph and Chord Diagrams.

tion that visualizes the large-scale graph through graph structure abstraction. SMG consists of sequences of Module Graphs. Since the pattern of each module is restricted to five types with similar node positions, as shown in Fig. 4.3, the morphing between two different modules should be feasible, as shown in Fig. 4.10. Since the size of each module will be different in a real SMG animation, we first enable the module type morphing with the same size and then enlarge or shrink the module size to ensure the smoothness of the animation.

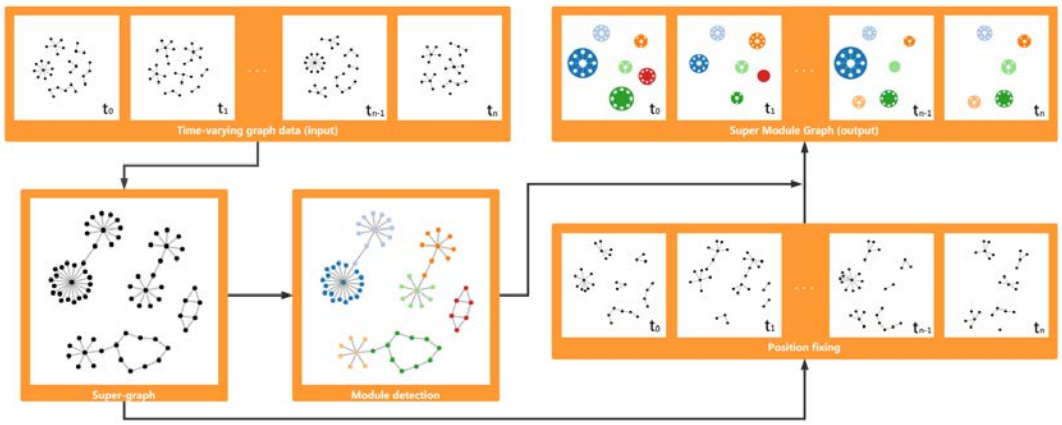


Figure 4.9: A pipeline of the SMG generation.

Table 4.1: The selected social network datasets.

Datasets	Nodes	Edges	Module	Brief Modules
Facebook	4,039	88,234	16	-
DBLP	317,080	1,049,866	442	-
Youtube	1,134,890	2,987,624	9821	8
Orkut	3,072,441	117,185,083	1969	244

4.4 Experimental Study

4.4.1 Social network

We select several large social network datasets from SNAP [73] which is a data library for analyzing large information networks. The selected social network dataset is from Facebook, DBLP, YouTube, and Orkut, as summarized in Table 4.1. The visualization results of Module Graph for these network data are shown in Fig. 4.8. From the results, we can find that the module patterns can be directly presented without any other interactions. From Fig. 4.8(a), we can find that most of the module structure of the Facebook network is a concentration structure. A high percentage of concentration modules shows that most of the people on Facebook like to follow the hot people. Fig. 4.8(b) is an assistant representation of Fig. 4.8(a). The outer circle in Fig. 4.8(b) shows the percentage of each module type. Each chord at the outside circle represents a module. The inside part shows the connection of each module.

(a) From tree-connection to concentration.

(b) From concentration to circular.

(c) From circular to low-connectivity.

(d) From low-connectivity to full-connectivity.

(e) From full-connectivity to tree-connection.

Figure 4.10: Morphing between two types of Module Graphs.

Fig. 4.8(c) shows the Module Graph of DBLP data. The DBLP data include the paper citation relationships of the researchers. Because there are many modules in the final result, we hide the outside circle of the Module Graph. From Fig. 4.8(c), we find that the module pattern appearing most often is the tree-connection pattern. When the user clicks the module, as shown in the red rectangle of Fig. 4.8(c), the related relationship of the selected module can be presented through a chord diagram, as shown in Fig. 4.8(d). The bottom part of Fig. 4.8(d) presents the pattern types.

For large-scale networks, the hierarchical Module Graph representation as shown in Fig. 4.12 produces a clearer visual representation which can aid significantly in querying and visual selections. The left side of Fig. 4.12 shows the visual results

Figure 4.11: An example of the morphing animation between two types of Module Graphs with size changing.

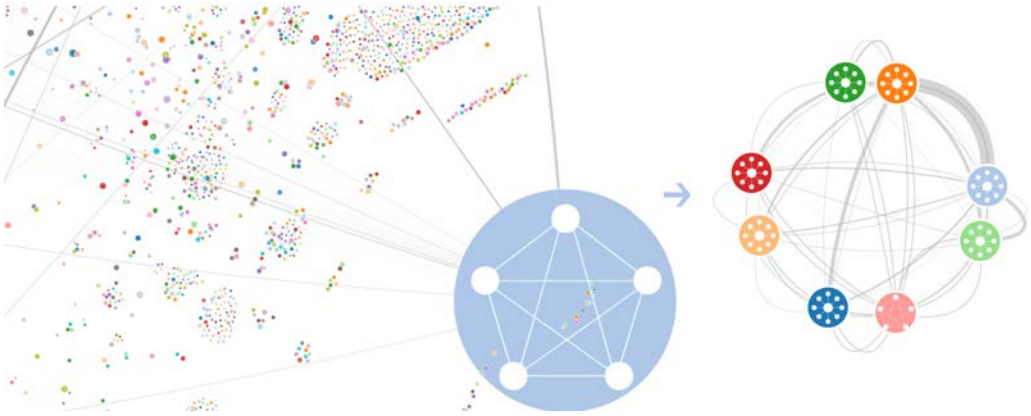


Figure 4.12: Hierarchical Module Graph of YouTube social network.

without the hierarchical process. Because the detected modules are beyond the screen size, we create a new Module Graph by making the previous Module Graph as the input graph. The right side of Fig. 4.12 is the hierarchically processed result.

To further describe the information of the Module Graph, we adopt box plots to show the type distributions of the modules, as shown in Fig. 4.13. Fig. 4.13 is related to the module information of the Orkut network. The five box plots in this Fig. 4.13 represent the five agent pattern statistics of the Module Graph. Each box plot includes

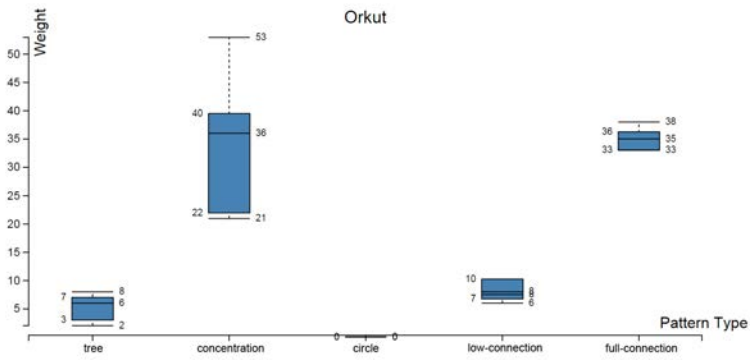
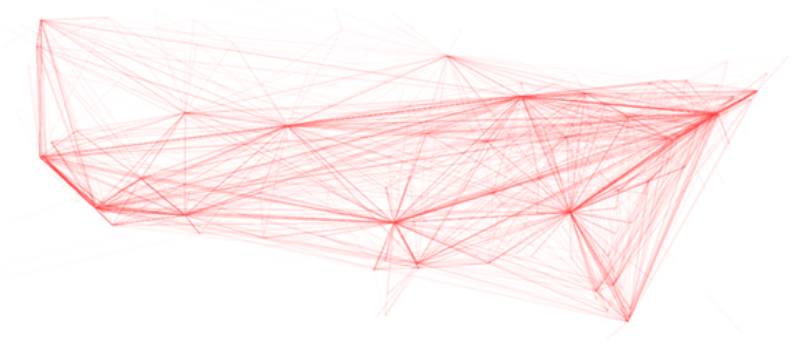


Figure 4.13: Visualization of the size distribution of modules via box plot diagram for the Orkut network.

five values: maximum, second maximum, median, second minimum, and minimum.



(a) Original visualization of the airline network.

(b) ModulGraph representation of the airline network.

Figure 4.14: The spatial network of U.S.A. airline flights for a given period of time.

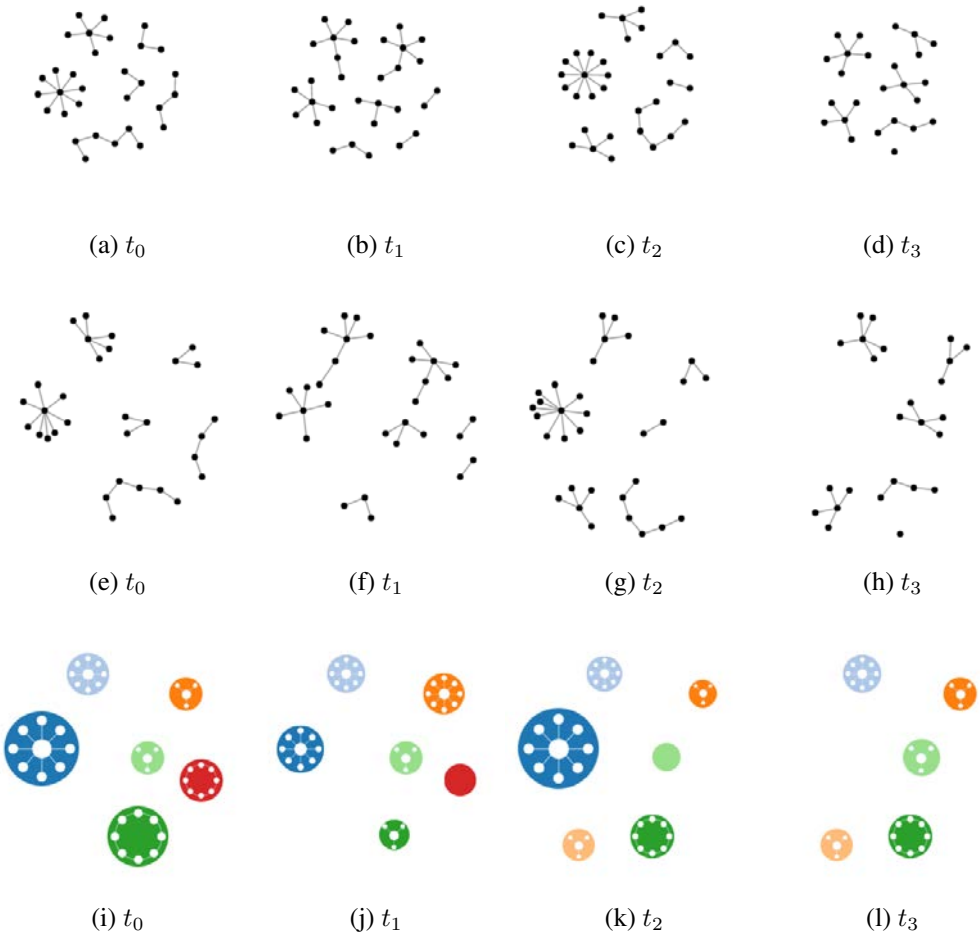
4.4.2 Spatial network

For the case study of a spatial network, we select a dataset with a massive number of flight records in the United States [152] in 2008. There are nearly 0.6 million flight lines, which are related to 3374 airports. Figure 4.14(a) is the original visualization of the airline network. By comparison with the simple representation, our method's result as shown in Fig. 4.14(b) represents the airport community distribution and the bundling airlines. In this case, the outside circle denotes the number of neighboring airports. The airline count inside the module is presented via the size of the inside circle. The pattern icon for each module represents the sub-network structure of the detected area. The information presented through Module Graph can help data analyzers acquire the important patterns from a large-scale network. For example,

the air traffic northwest of New York State can be easily observed from Fig. 4.14(b).

4.4.3 Streaming graph data

For the cast study of the streaming graph data visualization, we artificially create various sequences of graphs. We apply the SMG method on the artificial graph data and compared the result with the visualization result without using SMG as shown in Fig. 4.15.



(a) t_0 (b) t_1 (c) t_2 (d) t_3

(e) t_0 (f) t_1 (g) t_2 (h) t_3

(i) t_0 (j) t_1 (k) t_2 (l) t_3

Figure 4.15: Visualizing streaming graph data using SMG.

4.5 Implementation and Evaluation

We use D3 [21] to visualize the results on a computer with 2.8 GHz Intel i7 CPU and 16 GB RAM. The visualizations are displayed in the Chrome browser (45.0) with a 1440×900 resolution. Module detection is implemented using the C++ language on the Windows 7 operating system.

Table 4.2 summarizes the performance of three module detection methods: community detection [99], fast community detection [19], and our module detection method. We also provide the module detection performance with k-clustering. From the summary, we can find that our method can effectively support large-scale network aggregation and visualization. Because a distance feature has been considered in our method, our method is slightly slower than the fast community detection [19] for large-scale network processing. The time cost of k-clustering is ignorable because it is only related to the number of detected modules.

Table 4.2: Comparison of module detection performance.

Datasets	Community Detection [99]	Fast Community Detection [19]	Module Detection Method	Module Detection with k-Clustering
Facebook	1.1(s)	0.2(s)	1.2(s)	1.2(s)
DBLP	217.4(s)	11.0(s)	12.5(s)	13.1(s)
Youtube	Out of Memory	25.3(s)	27.4(s)	27.7(s)
Orkut	Out of Memory	247.1(s)	251.5(s)	251.6(s)
Flight Lines	54.8(s)	4.2(s)	5.7(s)	5.74(s)

Compared with the full network rendering, Module Graph achieves a satisfactory rendering performance for basic interactions. Timing results are shown in Table 4.3 where fps indicates frames per second. The rendering of the Module Graph of Orkut is performed at high speed because we adopted a hierarchical rendering strategy for the very-large Module Graph. Although in some experiments the massive data sizes reduce the rendering speed, such as DBLP and Youtube in Table 4.3, it is feasible to rewrite our Module Graph rendering applications directly on a GPU subsystem to achieve interactive rates. Since the implementation of SMG is an extension of the

Module Graph, we do not evaluate the time performance of SMG.

Table 4.3: Comparison of the rendering performance.

Datasets	Full Network	Module Graph
Facebook	0.19(fps)	58.82(fps)
DBLP	Out of Memory	0.94(fps)
Youtube	Out of Memory	0.55(fps)
Orkut	Out of Memory	17.86(fps)
Flight Lines	0.15(fps)	30.30(fps)

4.6 Conclusion

The *Module Graph* is an effective visualization tool for aggregating and representing large graph data through clustering and interconnectivity pattern analysis. A module detection method is presented to integrate the nodes that belong to the same group into a new integral module. We further extend the *Module Graph* method to SMG in order to visualize the streaming graph data. We also demonstrate that Module Graph can be applied to spatial network visualization. In the future, the context change between modules can also be considered in SMG implementation. Moreover, our method can be extended to visualize fuzzy networks that include overlapping communities and ego network that consists of a focal node.

CHAPTER 5

CONTENT-AWARE INFORMATION VISUALIZATION

5.1 Introduction

Content-aware resizing is an adaptive technique in image processing that removes less important content and retains more important content. The expression *content* indicates the concept of important interesting regions. This technique has also become a useful tool for information visualization since the diversity of displays for hardware is increasing. In addition, virtual displays of arbitrary size or aspect ratio, in the context of cloud-based visualization output, require content-aware resizing techniques. Although artists, web designers and programmers can design several available layouts for different scenarios, the task is time-consuming and costly.

Existing approaches for content-aware resizing mainly focus on natural images such as portraits, landscapes, and buildings. In information visualization, images normally consist of abstract mathematical representations such as vectors, points, lines, icons and geometrical shapes. The important content often represents the main subject in visual content.

Most of the existing image resizing approaches are not entirely suitable for information visualization. Grid-based methods [145] have been used for resizing such images as geometric distortions and are easily identified. Pixel-based seam carving, as in Avidan et al. [9], is normally used for image resizing. However, this technique cannot easily be extended to account for the layout adjustments of geographical scatterplots and social network graphs. In addition, the criteria for significant regions in visualization are different from those of natural images since their color schemes are different. For example, a blue sky in a natural scene is normally classified as

background. Therefore, it would often be considered less important than a person in the foreground. Unlike in natural images, a blue region rendered by a visualization system may be regarded as an important region.

Information visualization often consists of multiple information layers. For example, a geographical application would normally contain several layers such as water, continents, and various location markers. If we ignore major regions such as continents in resizing, then the results will suffer from distortions. Multi-layer based resizing is rarely discussed in the previous work on image resizing or information visualization resizing. The resizing framework of Wu et al. [162] assumed that the information visualization layer is single such as in a scatterplot, network, or word cloud. However, there are often many abstract layers in information visualization designs such as a scatterplot on a map and a graph with group shapes. Therefore, it is necessary to revisit the multi-layer approaches to detect and preserve the different layers in information visualization. When the resizing content is dynamic and the canvas becomes larger, the time performance becomes more important. Prior work such as that of Wu et al. [162] requires adjustment to rapidly resize the information visualization.

Hence, based on the resizing pipeline of Wu et al. [162], we present a different visualization resizing approach in four aspects. First, we define a visualization-related saliency map. Second, we consider the classes of information to be segregated into multi-layers for visualization. Third, the controlling mesh for resizing in our approach is adaptive so that users can emphasize the content of the visualization with fewer distortions. Fourth, we extend the resizing model for the streaming data visualization. We present the results of our experiments on different genres of multi-layered visualizations to demonstrate the performance of our approach.

The structure of this chapter is as follows. Section 5.2 gives an overview of our method. Section 5.3 discusses the visual saliency detector for visualization. Section 5.4 demonstrates the adaptive resizing model. Section 5.5 presents a stream-aware resizing approach. Section 5.6 compares and discusses the results and perfor-

mance, and Section 5.7 concludes the chapter and presents future work.

5.2 Overview

We define information visualization resizing as a saliency detection and geometric deformation problem. The input of our model is a multi-layered rendering. Multi-layers can be viewed as more than one representation in information visualization. In the example of Fig. 4.1, the input includes a geographical map and several scatterplots with different radii. First, we detect the visual saliency through a hybrid saliency model, the VSM, which can generate different saliencies for different layers in the visualization. In this step, we further improve the accuracy of the visual saliency detector by considering the lightness in color information. Second, we create an adaptive mesh that consists of controlling triangles with different levels of detail over the input visualization. The input data, such as scatterplots, were bound to vertices in the mesh according to their positions. Third, we formulate the controlling triangles for the resizing problem as an optimization problem according to the VSM. Finally, we resize the input visualization by solving a large sparse linear system. After the mesh deformation, the important features of the input image can be well preserved.

5.3 Visual Saliency Map (VSM)

The proposed saliency-based method, called the *visual saliency map* (VSM), adaptively indicates the significant regions in information visualization. For the content-aware resizing method, the deformation of each region is dependent on its corresponding saliency. Although the saliency of each region can be assigned by users manually, it is more effective to automatically detect the important regions.

The saliency concept for visualization is different than the one used in natural images in three aspects. First, the content in data visualizations is more contiguous than that in natural images, and hence it is unnecessary to pre-process the pixels before the

saliency detection. Second, regions with different colors but similar shapes in visualizations may have the same ground-truth saliency, but ordinary saliency detection algorithms for natural images normally define those regions with different saliencies. Third, the resizing approaches for natural images rarely take context into consideration, because context information is usually extracted from the less important regions in multi-layer visualizations.

Normally, the background maps in geographical applications and communities of social networks could be defined as context since they are closely related to important regions that users are interested in. Hence, we focus on two parts of the VSM. One is the saliency detector, which can detect the most important regions, and the other is the context detector, which first identifies the sharp edges in the visual rendering and then detects its adjacent context.

The methods of [49] and [166] can hierarchically detect the importance from images, but we should also consider the performance and the special features such as the color theme of visualizations. Although the method of clustering [162] works well for single-layer visualizations, it is time-consuming for multi-layer visualizations since the large area of less important layers is subjected to the same calculations. We are inspired by the work of [1] and propose an effective importance detector for information visualizations. Since the work of [1] requires further improvement when the information visualization has light colors, we consider more features for the saliency detector. Our saliency detector fulfills seven basic requirements.

1. It enhances the most important salient regions.

2. It considers frequently appearing global features that are salient in visualizations.

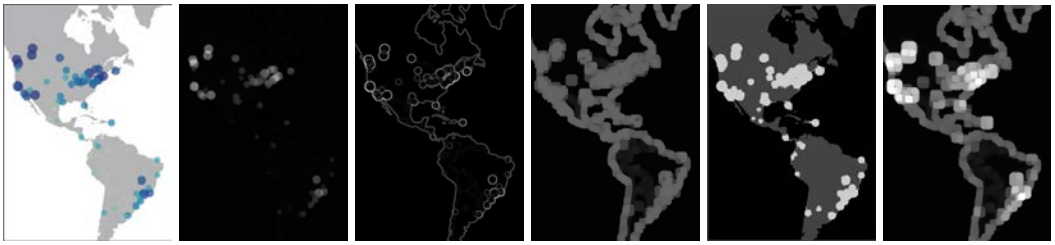
3. It considers local features such as sharp edges.

4. It detects secondary layers to avoid resizing distortion of multi-layer visualizations.

5. It considers light colors in information visualization.

6. It considers depth in information visualization.

7. It creates the saliency map with high efficiency and robustness.



(a) (b) (c) (d) (e) (f)

Figure 5.1: Different parts of VSM. (a) Original visualization image. (b) Important regions. (c) Sharp edges. (d) Context regions. (e) Content depth. (f) Final visual saliency map.

5.3.1 Importance detector

We adopt a frequency-based method [1] to detect the most important region (as shown in Fig. 5.1b). This method is formulated as follows:

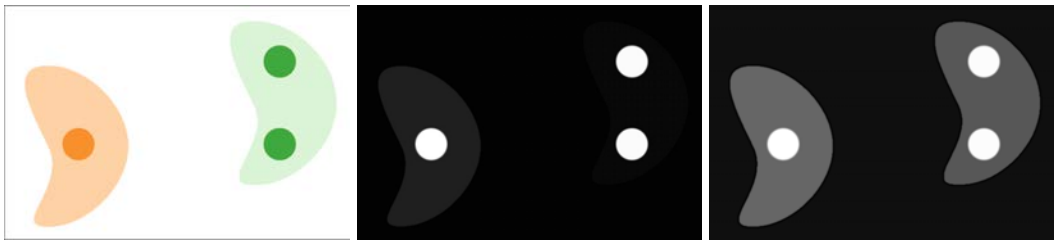
$$\mathrm{G(i,j,k)} = \frac{1}{2\pi \mathrm{k}^2} \mathrm{e}^{-(\mathrm{i}^2+\mathrm{j}^2)/2\mathrm{k}^2} \tag{5.1}$$

$$\mathrm{S_m(i,j)} = \|\mathbf{G}(\mathrm{i,j},\infty) - \mathbf{G}(\mathrm{i,j},\varepsilon)\|_2 \tag{5.2}$$

where $\mathbf{G}(i,j,\varepsilon)$ represents the Gaussian blurred values of pixel (i,j) for image rendering following the Gaussian filter function, as shown in Equation 5.1. The function $\mathbf{G}(i,j,\infty)$ is approximated through calculating the mean pixel value of the visual image. Since different features such as color and contrast should be considered in the importance detection, $\mathbf{G}$ is defined as a vector. In the method of [1], Achanta et al. adopted the LAB color space with the three features of lightness (L), one color-opponent dimension (A) and another color-opponent dimension (B), as a feature vector. The feature of lightness in the LAB color can represent the high-contrast regions, which are normally considered of high importance in information visualization.

Figure 5.2: Color palette from Tableau. The top palette of deep colors is normally used to represent the most important regions in visualization. On the other hand, the bottom palette of light colors indicates the less important regions or backgrounds in the visualization.

We found that using the LAB color space to maintain feature vectors is not enough to detect the available salient regions in information visualization. For example, if the designers follow the Tableau 20 palette [84], as shown in Fig. 5.2, it is hard to detect the available saliency for the light regions, as shown in Fig. 5.3(b). We define this problem as the similar-shape-different-color (SSDC) problem. We add the mid-channel feature to the feature vector so that we can ignore the extreme values of the color channel among the light colors. The mid-channel feature is represented by calculating the middle value of the three channels in RGB color. Finally, the feature vector in $\mathbf{G}$ can be defined as $\mathbf{v_f} = [l, a, b, m]$. The better visual contrast between features can be achieved by modifying the feature vector, as in [1], see Fig. 5.3(c).



(a) (b) (c)

Figure 5.3: (a) Multi-layer visualization with light color. (b) The saliency map is unavailable since the light green region is not detected. (c) Visual saliency map for solving SSDC problem.

5.3.2 Context detector

The context is normally the surrounding area of the important region. The context should also be considered since the distortion of context while resizing will also

affect the visual results. We present a method called *Edge Maximization* to detect the saliencies of the context.

We assume that the edge of the context should be preserved. Hence, we try to first detect the edges of the context. The *Sobel* and *Canny Operators* can be used to detect edges. We select the *Sobel Operator* since it is a discrete differentiation operator that has higher performance than the *Canny Operator*. The edge (as shown in Fig. 5.1d) of the visualization can be abstracted via the *Sobel Operator* as follows:

$$\mathrm{S_e(i,j)} = \sqrt{\mathrm{g_x(i,j)}^2 + \mathrm{g_y(i,j)}^2} \tag{5.3}$$

where $\mathrm{g}_x(i,j)$ is the gradient along the horizontal direction and $\mathrm{g}_y(i,j)$ is the gradient along the vertical direction. Since the edge is narrow, it is difficult to generate triangles. Thus, we enhance the edges using a dilation operator that extends the edge by the structuring element $\mathbf{B}$. Hence, we can formulate the context saliency as

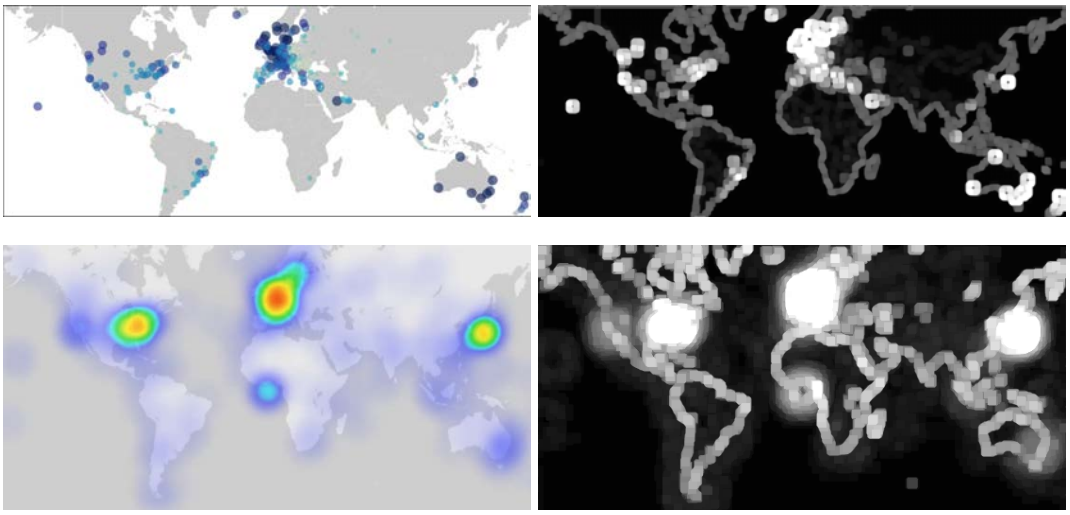
$$\mathrm{S_c(i,j)} = \max\{\mathrm{S_e(i+m, j+n)}\} | (\mathrm{m,n}) \in \mathbf{B} \tag{5.4}$$

where $\mathbf{B}$ is a square of radius r and (m,n) is a point in $\mathbf{B}$.

The depth of the shape in information visualization requires further optimization for resizing. A shape with a closer depth will be assigned with greater saliency. We define the content depth of information as $\mathrm{S_d(i,j)}$. The content depth can be acquired from the data of the original information. As a result, we formulate the final VSM (Fig. 5.1f is an example) as follows:

$$\mathrm{S(i,j)} = \mathrm{S_m(i,j)} + \mathrm{S_c(i,j)} + \mathrm{S_d(i,j)} \tag{5.5}$$

The Gaussian filter can be approximated through a filter template $\frac{1}{4}[1,2,1]$ with $k = 1.6$. The radius of the dilation operator is set to 3. The final value of the saliency map should be normalized to limit the value in $[0,1]$. With the VSM, we can adaptively indicate the saliency layers of various visual elements that are the preparation of our resizing mode. Figure. 5.4 shows the results of our VSM.



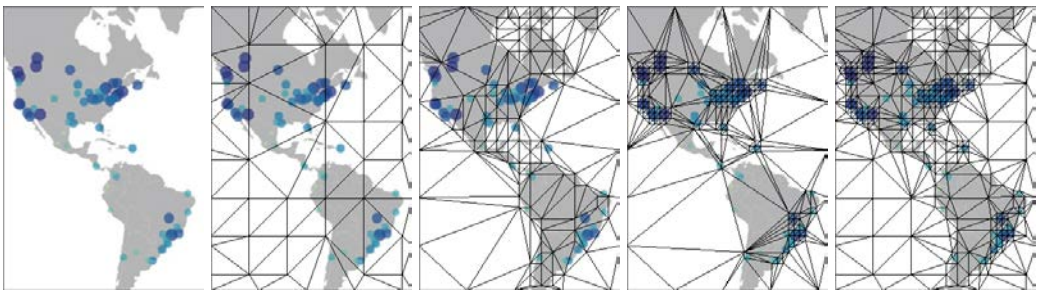
(a) Original visualization images. (b) Visual saliency map.

Figure 5.4: Results of visual saliency map. (top) Visualization of cost of living in different countries as described in Sec 5.6.1. (middle) A heat map of users' locations from Brightkite as described in Sec 5.6.1. (bottom) A graph shows a social network with hundreds of nodes.

5.4 Adaptive Resizing Model

In the following sections, we describe our resizing model in detail. First, we start with the adaptive meshing. We resort to triangular meshes, as they can be more readily adapted to high-density regions.

5.4.1 Adaptive triangulation



(a) (b) (c) (d) (e)

Figure 5.5: Triangulations with different levels of details where $\alpha = 20$. (a) Original visualization. (b) Level=1, $\mathrm{S(i,j)} \in [0, 0.15)$. (c) Level=2, $\mathrm{S(i,j)} \in [0.15, 0.5)$. (d) Level=3, $\mathrm{S(i,j)} \in [0.5, 1.0]$). (e) Level=1-3, $\mathrm{S(i,j)} \in [0.0, 1.0]$.

Here we propose an adaptive triangulation method to reduce the degree of factors (DOF) that are related to the performance of resizing.

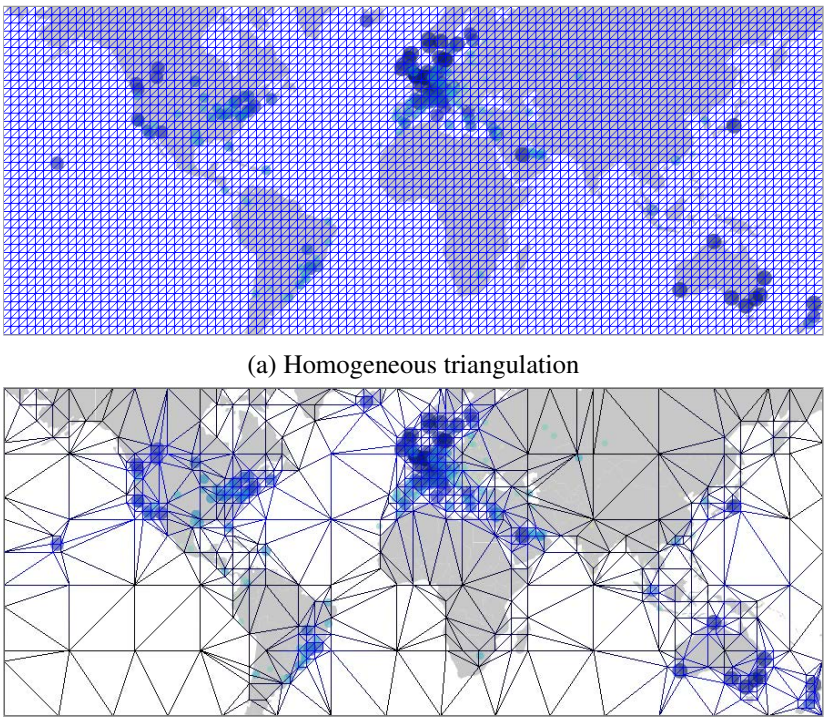
Based on the VSM, we vary the density of triangles for different important regions. First, a point set will be extracted on the basis of the intensities in the saliency map. *Delaunay triangulation* [122] can be used to generate triangles. For two-dimensional triangulation, a set of key points $\mathbf{V}$ will construct a mesh $\mathbf{M}$. $\mathbf{M}$ should fulfill three constraints. First, the edges in $\mathbf{M}$ do not contain any key points in $\mathbf{V}$ except for the start and end points. Second, an edge cannot intersect another edge. Third, all elements in $\mathbf{M}$ are triangles. These elements form the convex hull of $\mathbf{V}$.

We utilize the VSM to create a set of key points, the $\mathbf{V}$ band that satisfies these three constraints to generate $\mathbf{M}$ that includes several triangles. The threshold is a quick method to create key points from a saliency map by extracting the points with high saliency, but a large number of key points will be extracted if there are many pixels with high saliency. Therefore, we construct a constraint to determine the interval of triangulation sampling to limit the total count of points in $\mathbf{V}$. We define the constant count of points in $\mathbf{V}$ through $\mathrm{C}_p = \frac{wh}{\alpha^2}$, where w and h are respectively the width and height of the canvas in the visualization, and α is a free parameter that is used to adjust $\mathrm{C_p}$. Different values of the saliency can be mapped to different levels of triangulations. We use t_{k0} and t_{k1} to indicate the range of saliency values for level k. The interval of the triangulation sampling for different levels k could be formulated as follows:

$$\mathrm{I}_k = \frac{C_p}{\sqrt{\sum_{t_{k0} \leq \mathrm{S}(i,j) < t_{k1}} \mathrm{S}(i,j)}} \tag{5.6}$$

When the saliency of a region is between t_{k0} and t_{k1}, it should follow the interval of sampling $\mathrm{I_k}$. The interval of sampling indicates the count of interval pixels of each key point v_k in $\mathbf{V}$. Figure 5.5 shows an example of triangulation with different levels of detail. In our resizing model, we merge three levels of triangulation into one, as shown in Fig. 5.5(e). By using VSM and *Delaunay triangulation*, we can generate content-aware triangles, as shown in Fig. 5.6(b), which is more flexible than

the homogeneous triangulation in Fig. 5.6(a).



(a) Homogeneous triangulation

(b) Adaptive triangulation

Figure 5.6: Two strategies of triangulation.

5.4.2 Mesh deformation

Deformation energy optimization

We represent the topology of the mesh with $\mathbf{M} = (\mathbf{V}, \mathbf{E}, \mathbf{T})$, where $\mathbf{V} = [\mathbf{v}_0^T, \mathbf{v}_1^T, ..., \mathbf{v}_n^T]$,($\mathbf{v_i} \in \mathbb{R}^2$) denotes the vertices, $\mathbf{E}$ are the edges and $\mathbf{T}$ depicts the triangles that each have three vertices and three edges. We use $\mathbf{V}' = [\mathbf{v}_0'^T, \mathbf{v}_1'^T, ..., \mathbf{v}_n'^T]$ to indicate the deformed vertices. $\mathbf{E} = [\mathbf{e}_0^T, \mathbf{e}_1^T, ..., \mathbf{e}_n^T]$, $(\mathbf{e}_i = \mathbf{v}_a - \mathbf{v}_b)$ was used to indicate the edges. The edges divide the visualization into several patches. The basic idea of mesh resizing is optimizing a linear energy function, as shown in [48]. The linear energy function can be defined as $\int_y^h \int_x^w S(x,y) \left\| \mathbf{J_F} - \mathbf{P} \right\|_2^2 dxdy$, where $\mathbf{J_F}$ is a Jacobian matrix, $\mathbf{P}$ represents the linear transformation, and w and h are the width and height of the canvas, respectively.

Inspired by Wang and his colleagues [145], we assume that the triangles bound

to areas with high saliency will be assigned a smaller scaling factor, while those that cover less important areas can be distorted by the linear scaling operations. Our method is different from that of Wang et al. [145] since we change the basic controlling unit from a rectangular grid to a triangular mesh. The resizing energy for a triangle can be formulated as

$$\mathrm{R(t)} = \sum_{\{\mathrm{m,n}\}\in\mathrm{E(t)}} \left\| (\mathrm{v_m}' - \mathrm{v_n}') - \mathrm{s_t}(\mathrm{v_m} - \mathrm{v_n}) \right\|^2, \tag{5.7}$$

where $t \in \mathbf{T}$ indicates a triangle, s_t is a scaling factor for each vertex v in the triangle and we assume that the scaling factors for the x-axis and y-axis are the same, $\mathrm{E(t)}$ is a set of edges in t, v and v' respectively indicate the vertex of an original triangle and its corresponding deformed vertex, and $\|\cdot\|_2$ is the L_2 norm of a vector. We can differentiate $\mathrm{R(t)}$ and continue getting a direct solution of s_t, as shown in [145]. As we can see, $\mathrm{R(t)}$ calculates the resizing energy of the edges, but different triangles should have different degrees of warping due to their different saliencies. Hence, we need to take the saliency value of each triangle into consideration. The resizing energy of all the triangles is formulated as

$$\mathrm{R} = \sum_{t\in\mathbf{T}} \lambda_t \mathrm{R(t)} = \sum_{t\in\mathbf{T}} \frac{1}{n} \mathrm{R(t)} \sum_{\mathbf{x}\in\mathbf{P_t}} \mathrm{S}(\mathbf{x}), \tag{5.8}$$

where $\lambda_t = \frac{1}{n} \sum_{\mathbf{x}\in\mathbf{P_t}} \mathrm{S}(\mathbf{x})$ is the saliency of each triangle, $\mathbf{P_t}$ is the set of all pixels in t, $\mathbf{x}$ indicates the position of a pixel, and $\mathbf{n}$ is the number of pixels in $\mathbf{P_t}$. Since the saliency of each pixel can be calculated from the VSM, we can use a scan-line algorithm to detect the entire pixel saliency contained in one triangle. This leads to the triangle saliency λ_t. Since R is a quadratic energy function, we can minimize the total resizing energy to obtain the final deformed vertices $\mathbf{V}'$.

Constraints

To get the accurate resizing result, the constraints should be taken into account. The constraints for resizing includes overlapping preventing, boundary and smoothing.

We adopted the method as shown in [62] to guarantee that the mesh is resized accurately without overlap. Beyond that, we formulate two constraints, such as boundary checking and mesh smoothing, for resizing.

- Boundary Constraints

Since the boundary of the visualization should fulfill the linear scaling in order to avoid the warp, R should obey the constraints on the boundary. We define the vertices set on the boundary as B, which includes four corner vertices and several vertices on four side edges of the boundary. The vertex set on four side edges could be defined as $\mathrm{B(left)}$, $\mathrm{B(top)}$, $\mathrm{B(right)}$ and $\mathrm{B(bottom)}$. Hence, the constraints for B can be defined respectively as:

$$\begin{cases} \mathbf{v}_{left,top} = (0,0)^T, \\ \mathbf{v}_{right,top} = (w,0)^T, \\ \mathbf{v}_{left,bottom} = (0,h)^T, \\ \mathbf{v}_{right,bottom} = (w,h)^T \end{cases} \tag{5.9}$$

$$\begin{cases} \mathbf{v}_i' = (0, v_{iy}), i \in \mathrm{B\,(left)}\,, \\ \mathbf{v}_i' = (v_{ix}, 0), i \in \mathrm{B\,(top)}\,, \\ \mathbf{v}_i' = (w, v_{iy}), i \in \mathrm{B\,(right)}\,, \\ \mathbf{v}_i' = (v_{ix}, h), i \in \mathrm{B\,(bottom)}\,. \end{cases}, \tag{5.10}$$

where w indicates the width of the canvas and h indicates the height of the canvas.

- Smoothing Constraint

It is necessary to further avoid distortion through smoothing constraint as shown in the works of Wang et al. [145] and Wu et al. [162]. We formulate a scaling energy function as follows to smooth out the scaling factor of each triangle:

$$\mathrm{E_s} = \sum_{t \in T} \sum_{n \in \mathrm{Adj(t)}} \frac{1}{2}(a_t + a_n)(\lambda_t s_t - \lambda_n s_n)^2 \tag{5.11}$$

where $\mathrm{Adj(t)}$ is a set of adjacent triangles of t, n indicates one adjacent triangle, a_t and a_n respectively represent the acreage of t and n. $\mathrm{E_s}$ is an energy function that represents the distortion scaling factor of the adjacent triangles. We can achieve better scaling factors by optimizing $\mathrm{E_s}$.

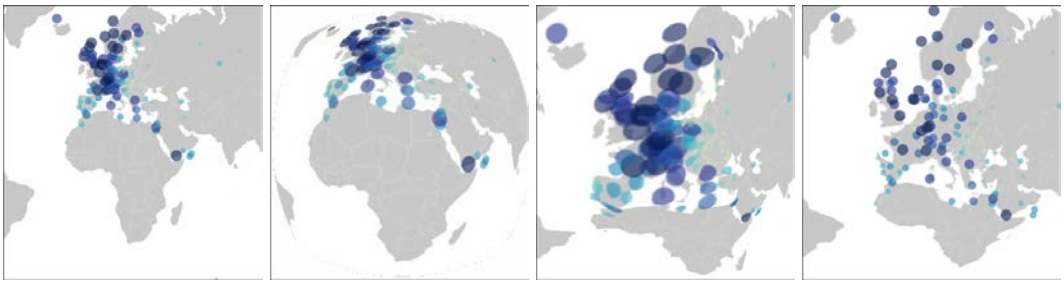
5.4.3 Vector adjustment

The proposed resizing model works well for the image resizing of the visualization. It can be easily extended to the vectorial resizing of the visualization with feature preservation. Vectorial resizing means that some vectorial elements such as points and edges can be bound to the vertices in the triangulated meshes. The positions of elements in a visualization will be adjusted through mesh resizing. We take the scatterplots on a world map as an example. We assume points in the scatterplots are the main regions and the background map is the context that also needs to be preserved. Each point will be bound to the vertex of the triangulated mesh. The adjustment of positions can be achieved by resizing the controlling mesh. We use the whole visualization image to generate the saliency map and respectively resize the main part and the context part. In the final step, adjusted points will replace the main part and combine with context part as the final result according to their adjusted positions. As shown in Fig. 5.11, in our experiments we adopt a hybrid method which takes advantage of both image resizing and vectorial resizing.

5.4.4 Content enhancement

Content enhancement is aimed at further enhancing the important regions while resizing the canvas. For example, there are many overlapping plots on the region of Europe; thus, it is difficult for the users to get integrated information as shown in Fig. 5.7. Fish-eye distortion [119] is a solution for solving this problem. However, there are two drawbacks of the fish-eye distortion method. First, manual selection of the significant region is tedious for the users. Second, the spherical representation of the fish-eys distortion is not suitable for enhancing irregular regions.

We present an approach that achieves better content enhancement for the information visualization. In addition, the proposed method has a potential for a resizing application that all content will be transformed from a large display to a small one. In our approach, we add an enhancing factor, called δ_k, to control the enhancement



(a) (b) (c) (d)

Figure 5.7: Results of three different strategies for content enhancement. (a) No enhancement. (b) Fisheye enhancement. (c) Content-aware enhancement without vectorial adjustment. (d) Content-aware enhancement with vectorial adjustment.

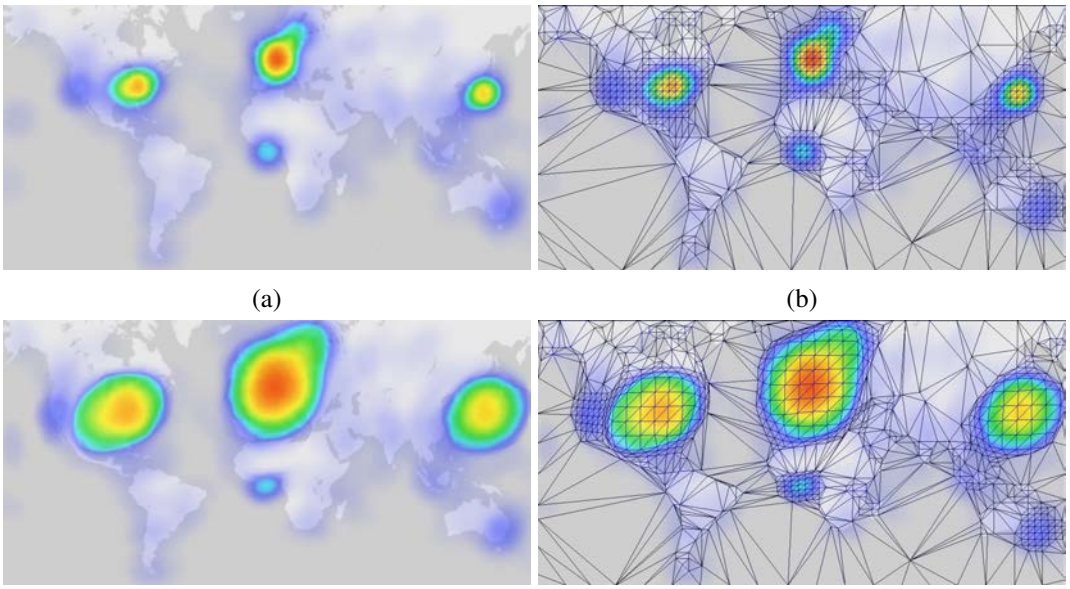
of the different layers in a visualization. A scaling factor s_t is formulated in the energy function of the resizing model. k denotes the layer of saliency. We formulate an enhancing energy function for content enhancement as shown in Eq. 5.12. The values of δ_k can be adjusted by the user to interactively enhance the desired regions. Figure 5.7 and Fig. 5.8 show two results of the content enhancement.

$$\mathrm{E(t)} = \sum_{\mathrm{\{m,n\} \in E(t)}} \left\| (\mathrm{v_m}' - \mathrm{v_n}') - \mathrm{s_t}\delta_\mathrm{k}\lambda_\mathrm{t}(\mathrm{v_m} - \mathrm{v_n}) \right\|^2. \tag{5.12}$$

5.5 Stream-Aware Resizing

The resizing becomes a challenge when the content dynamically changes in time. The resizing model for a static image or information visualization will cause waving because the temporal coherence is not considered. However, most of the monitoring data from physical devices are streams, so a content-aware resizing model should account for the stream nature. Wang et al. [143] presented a related solution for the resizing of a video stream through a video-based importance map detection and a video-based grid warp model. However, similar work has not been investigated from an information visualization perspective. Therefore, we propose a stream-aware resizing approach to deal with the resizing problem of dynamic information visualization.

Directly using a static resizing model to resize each frame of the streaming data



(a) (b)

(c) (d)

Figure 5.8: Result of the content enhancement without the canvas resizing. The content in this visualization is a heat-map of the people locations as described in Sec. 5.6.1. (a) Original visualization. (b) Triangulation of the original visualization. (c) Enhanced visualization without the canvas resizing. (d) Corresponding triangles of the enhanced visualization.

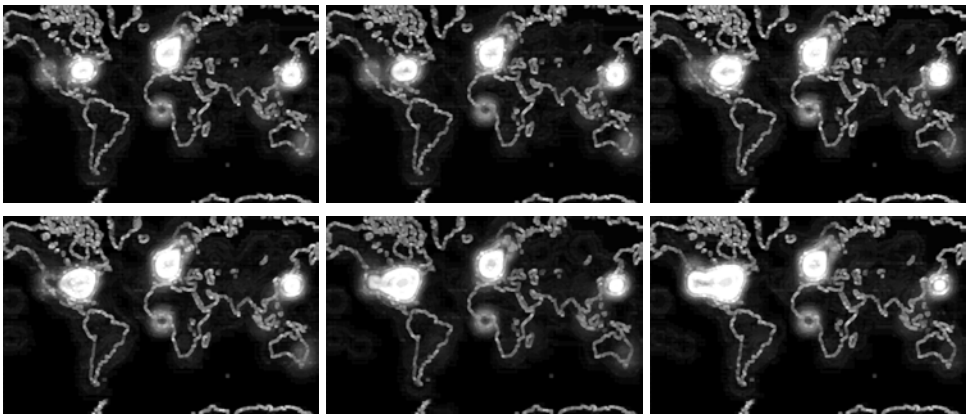
will cause abrupt changes in the region of importance that will make the human system unsuitable. We find that the abrupt changes are mainly caused by the variation of the VSM at each frame. Therefore, we consider generating a uniform VSM for a set of continuous frames with a similar nature.

First, we cluster the frames into k segments. Each segment includes a set of continuous frames that will use the same VSM in the stream-aware resizing model. We generate the VSM for a segment, $SVSM$, through the saliency maximization approach (SMA). The input of the SMA is the sequences of the VSM detected by using the VSM model presented before. The SMA can be formulated as

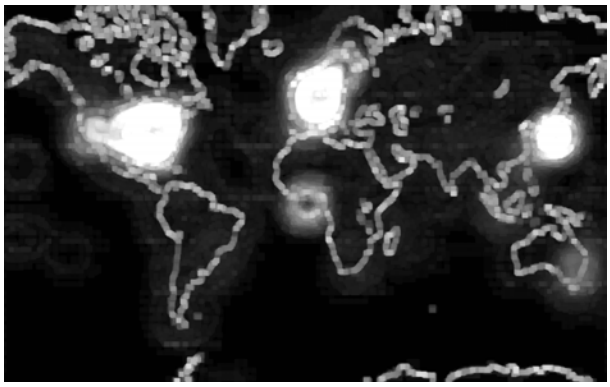
$$SVSM(x,y) = Max(VSM_1(x,y), VSM_2(x,y), ..., VSM_m(x,y)), \qquad (5.13)$$

where m is the frame number of a segment and x and y indicate the pixel position of the VSM.

Since we apply the same VSM for the frames of a segment in the resizing process,



(a) Six VSMs.



(b) SVSM.

Figure 5.9: An example of a SVSM that is generated by applying SMA on six VSMs.

the abrupt changes will not appear. Figure 5.9(b) shows an example of a SVSM that is generated by applying SMA on six VSMs. To avoid sudden changes between two segments, we can further apply linear blending on a $SVSM$, as shown in Eq. 5.14.

$$SVSM'_j = \frac{1}{4}(SVSM_{j-1} + SVSM_{j+1}) + \frac{1}{2}SVSM_j \tag{5.14}$$

5.6 Experiments and Results

All the experiments in this chapter were performed on a computer with the Windows 7 OS, an Intel i7 CPU 2.8 GHz and 8 GB RAM. We implemented the algorithm in C++ and used the CGAL [20] library to generate the triangulation. The Lapack++ library was employed to solve the large sparse linear system of equations.

We tested our method on several datasets and obtained better results than previous

methods. Since we use the adaptive method to generate triangles, better performance can be achieved than previous methods for visualization resizing. We show the performance of our method on Tab. 5.1 and evaluate our work through computing the remaining saliency (Fig 5.10) in the resized visualization.

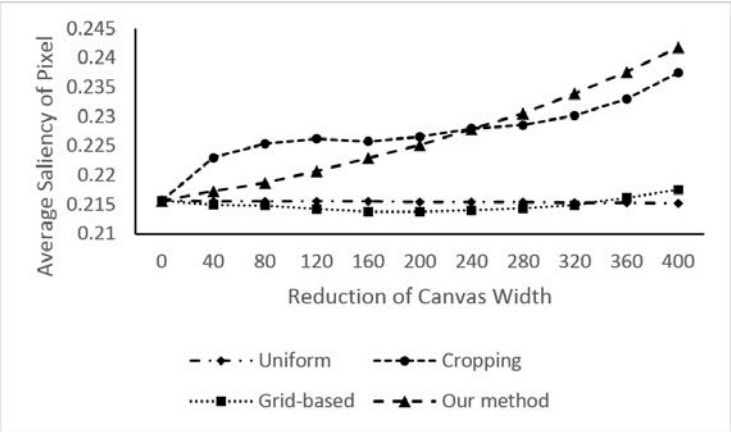


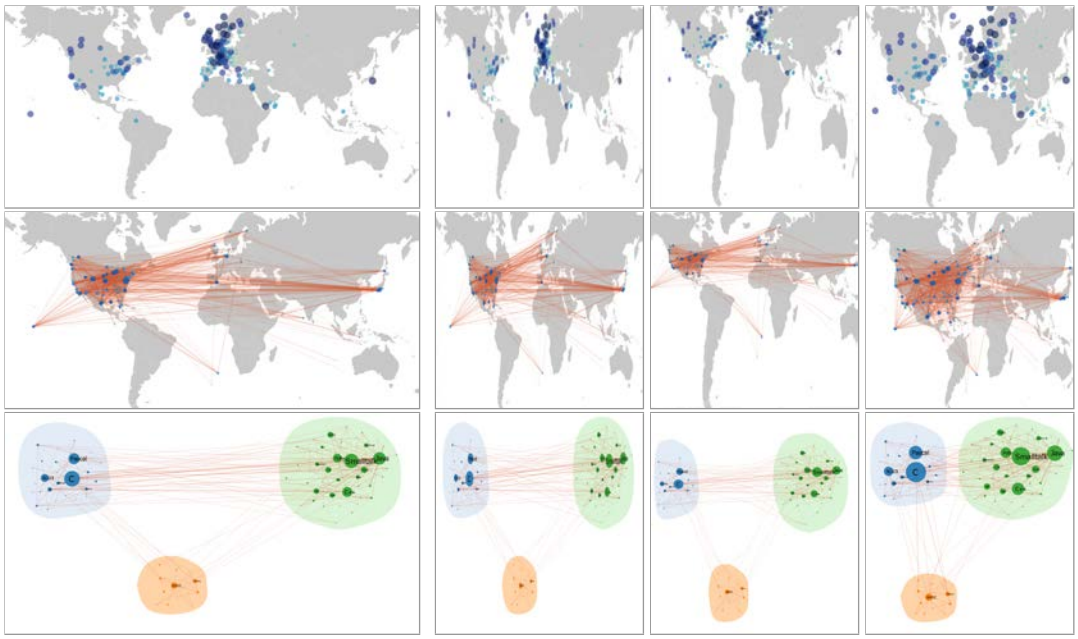
Figure 5.10: A comparison of the preservation of content measured by the average pixel saliency using four different approaches of resizing. The testing visualization is the bottom case, as shown in Fig. 5.15.

5.6.1 Results

We base our experiments on multi-layered information features for visualization. We analyze three cases of resizing using the method presented in this chapter. We compared our approach with the linear scaling and grid-based methods. Our results demonstrate that the adaptive content-aware method has better visual feature representation than the grid-based method for multi-layered visualization resizing. We show additional resizing results in Fig. 5.15.

Scatterplots and Geographical Maps (SGM)

In the first multi-layer visualization case, we select a dataset from numbeo.com that records the costs of living around the world. The scatterplots and geographical map (SGM) can be regarded as two layers of the visualization of this dataset. The scatterplots show the costs of living in different countries, and the geographical map



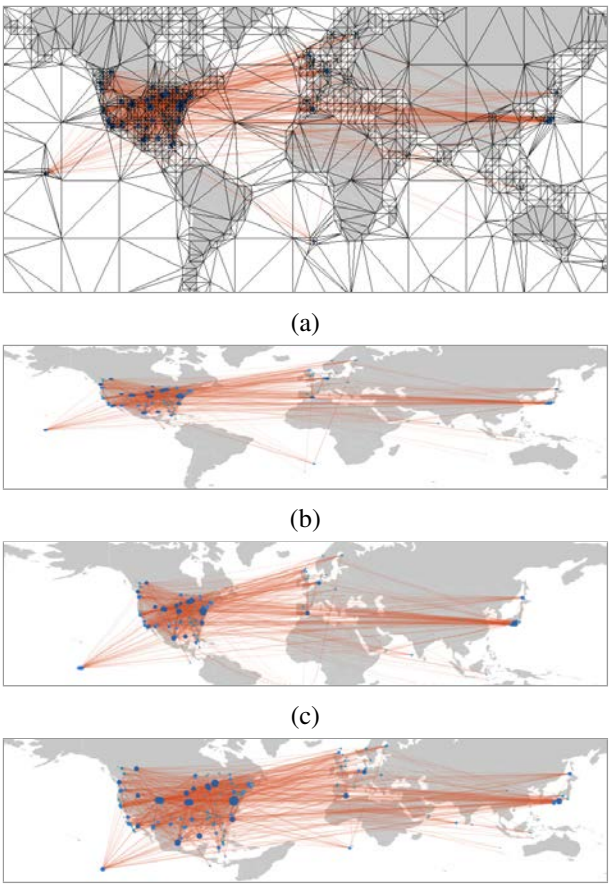
(a) (b) (c) (d)

Figure 5.11: Results of 3 different strategies for horizontal resizing of SGM 5.6.1, GGM 5.6.1 and GC 5.6.1. (a) Original visualization images. (b) Linear scaling. (c) Grid-based resizing [145]. (d) Our method.

demonstrates the distribution of the countries. The algorithm in [162] acts on just one layer in the visualization, such as the scatterplots, and we take another layer such as the geographical map into consideration. Since the geographical map occupies a large area, computing the cluster map in [162] requires more calculations. We can see that all the circles in Fig. 5.11(top) suffer less distortion after the resizing by using our method. We can also adjust the enhancing parameter to enhance important the regions in the visualization, as shown in Fig. 5.7(d).

Graph and Geographical Maps (GGM)

In the second case, we select a large dataset called Brightkite [30]. Each record in this dataset includes a geographical location. When the canvas is resized without the content-aware constraints, two main problems emerge. First, the re-layout algorithm such as the direct-force method is time-consuming for large graphs. Second, the important regions are not preserved. Although regular grid-based methods can preserve



(a) (b) (c) (d)

Figure 5.12: Results of 3 different strategies for the vertical resizing of GGM 5.6.1. (a) Original visualization with triangulations. (b) Linear scaling result. (c) Grid-based resizing result [145]. (d) Our method.

important regions, the high density of the grid will necessitate excessive calculation because the computation is related to the grid density. In our method, we bind each node with a nearby vertex in the controlling mesh and redraw the nodes after resizing the mesh to achieve a high performance of content-aware resizing. The resizing results of the GGM are shown in Fig 5.11(middle) and Fig 5.12.

Graph and Community (GC)

The third example demonstrates the advantages of our method in dealing with the SSDC problem, as discussed in Sec. 5.3.1. We select a graph-based dataset that presents the relationships between computer languages. This dataset can be considered as a social network of computer language. In the visualization, this dataset can

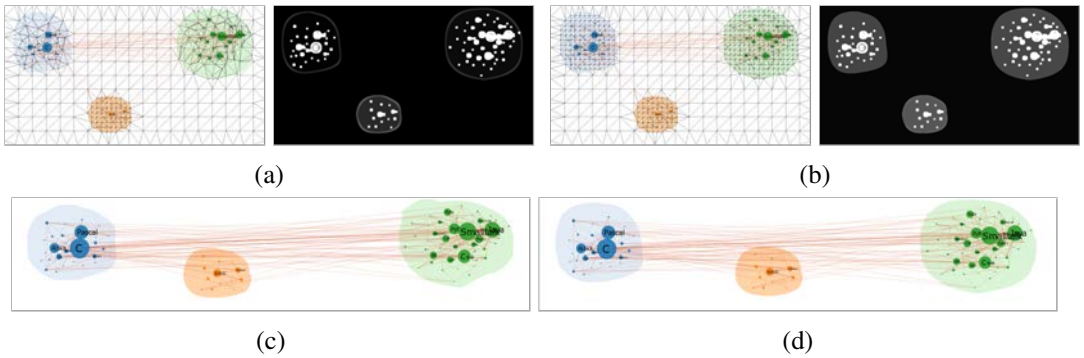


Figure 5.13: Results of 2 different strategies for vertical resizing of GC 5.6.1. (a) Triangulation and saliency map without SSDC fixing. (b) Triangulation and saliency map with SSDC fixing. (c) Result of vertical resizing of GC without SSDC fixing. (d) Result of vertical resizing of GC with SSDC fixing.

be represented as a graph and community. All the nodes in the graph are assigned with different colors according to their *Modularity Classes*, which were computed by a statistical method presented by Blondel et al. [19]. Nodes of similar color will build a community, which is presented as an ellipse-like shape in the visualization image. We draw the visualization image with the color palette from Tableau. Our result in Fig. 5.11(bottom) and Fig 5.13 shows that our method can achieve a better result and avoid the problem of SSDC.

Infographic (IG)

The fourth example demonstrates the advantages of our method for handmade multi-layer visualization. An infographic is a visual representation of information that utilizes text, lines, and graphics to improve the visual pattern discovery. Unlike splatterplot and graph, the layout of an infographic is usually created by a vector graphics tool such as Adobe Illustrator. We selected a vector graphic from Shutterstock.com and applied our method to resize the vector graphic. Our result in Fig. 5.14(d) shows that our method can obtain a better result than linear scaling for infographics.

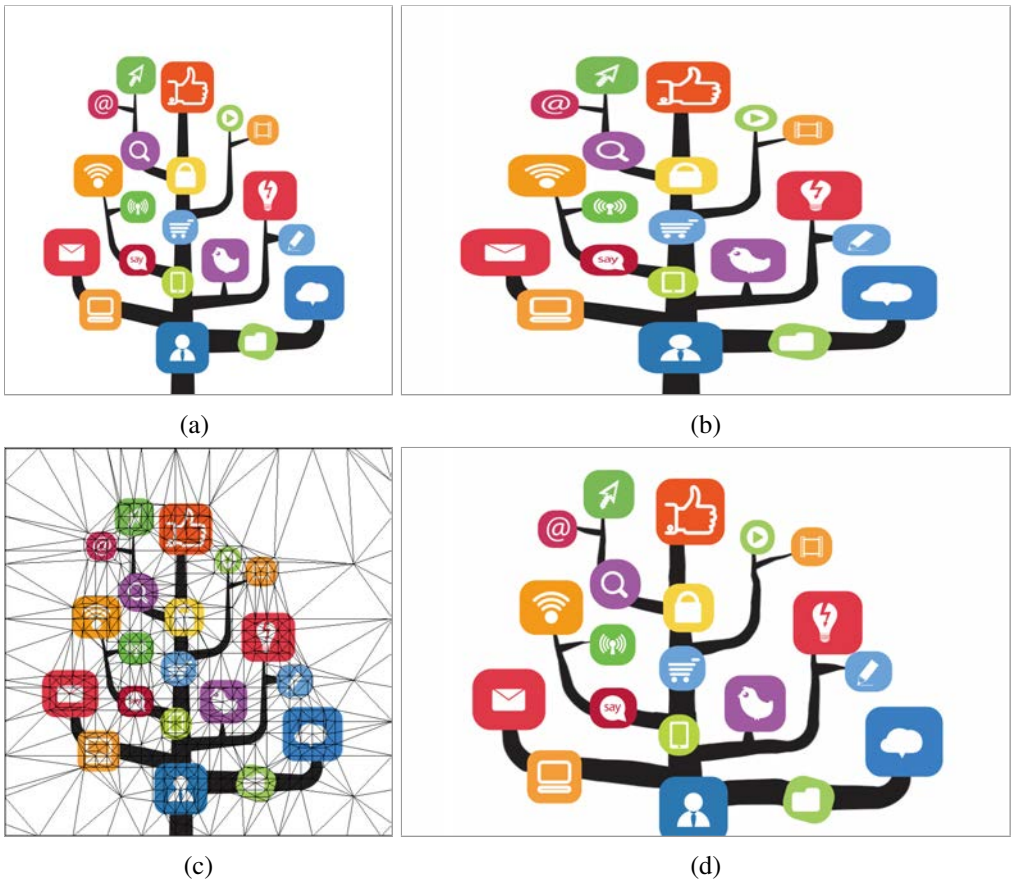
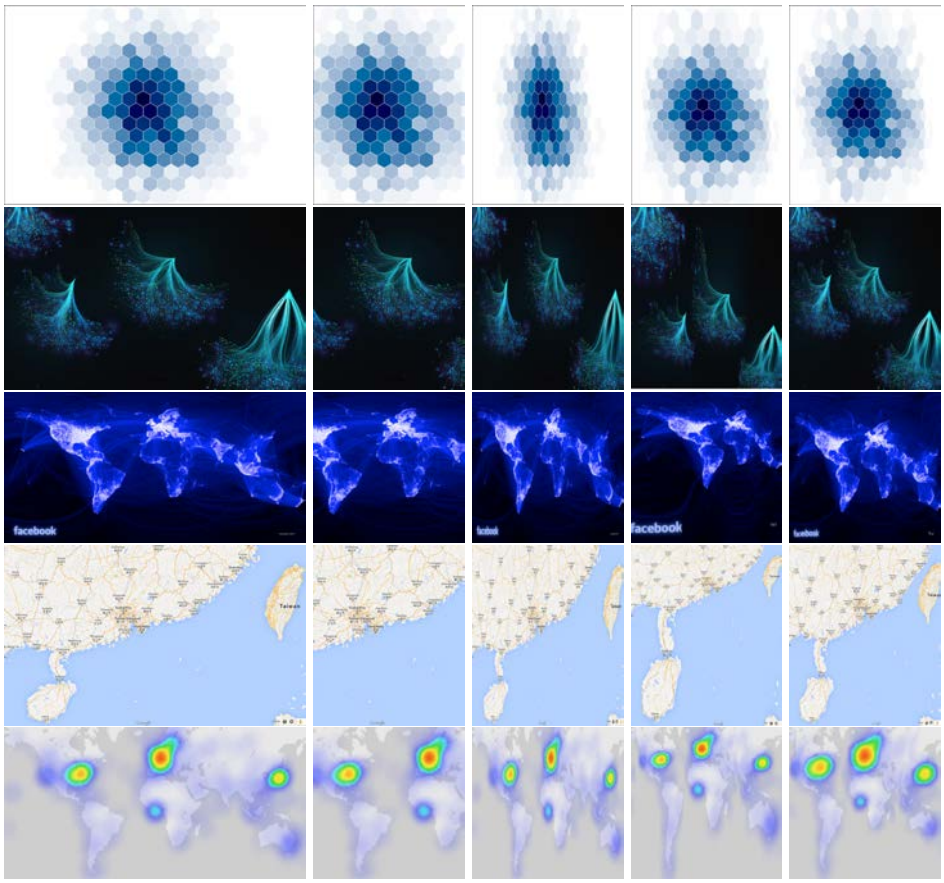


Figure 5.14: Results of 2 different approaches for the horizontal resizing of the visualization in IG 5.6.1. (a) Original visualization with triangulations. (b) Linear scaling result. (c) Original visualization with triangulations. (d) Our method.

Streaming heatmap

The fifth example demonstrates the performance of our stream-aware resizing model. We apply the model to a set of point frames on a geographical map. The stream-aware resizing model can be applied directly on the heatmap of points. We can first adjust the point layout through triangle deformation and then generate the heatmap on the updated point positions. Our result in Fig. 5.16(d) shows that our method can preserve the significant regions better than directly using a resizing model without stream-aware consideration.



(a) (b) (c) (d) (e)

Figure 5.15: Additional results of four different methods for reducing the width of the visualization. (a) Original visualization. (b) Cropping. (c) Linear resizing. (d) Grid-based resizing. (e) Ours. Figure 5.10 shows the saliency preservation of each method by using the bottom case in this figure.

5.6.2 Discussions

Performance

The performance of our method is better than that of previous resizing algorithms for visualizations, as shown in Tab 5.1. This is because we implement an adaptive method to generate the geometric mesh, which substantially improves the performance when a visualization image has more than half empty regions, which mean that the regions have zero saliency. Furthermore, we adopt a simple but efficient method to create the VSM, so that more time-consuming calculations, such as estimating the kernel density, can be avoided.

Table 5.1: Performance and evaluation of our experiments. The top data in each row indicate the results using a grid-based method [145], and the bottom data in each row address the results using the proposed method.

Cases	Original size	New size	Vertices	DOF	Time cost (ms)
SGM Sec. 5.6.1	876×374	385×374	741	4104	175
			324	586	19
GGM Sec. 5.6.1	876×414	438×414	1012	5670	263
			589	3242	64
GC Sec. 5.6.1	1500×800	750×800	2886	16644	4739
			853	4824	161
IG Sec. 5.6.1	678×680	1066×680	1849	10584	1337
			611	3498	58
Hex Points	750×495	375×495	950	3800	206
			408	2310	19
Graphs	600×299	300×299	465	2520	42
			405	2322	19
Social Network	600×363	300×363	589	2356	54
			280	1560	12
Heat Map	1126×563	663×563	1653	6612	1024
			479	2694	30
China Map	1320×791	660×791	2688	10752	3932
			393	2142	32

Evaluation

There are many methods to evaluate the effect of visualization resizing as shown in the cognitive experiments [116]. We mainly focus on evaluating three attributes: the count of vertices in the controlling mesh, degrees of freedom (DOF), and time cost. DOF indicates the number of variables that are free to change in the final computation. DOF is a statistical concept and is often used in evaluating the performance of retargeting, such as in [63]. Since our model has advantages in controlling the mesh reduction, DOF can be used in our evaluation. The results of the evaluation are shown in Tab. 5.1. From the evaluation, we find that the presented approach has better performance, especially for a large canvas size, and has fewer DOFs than the grid-based method.

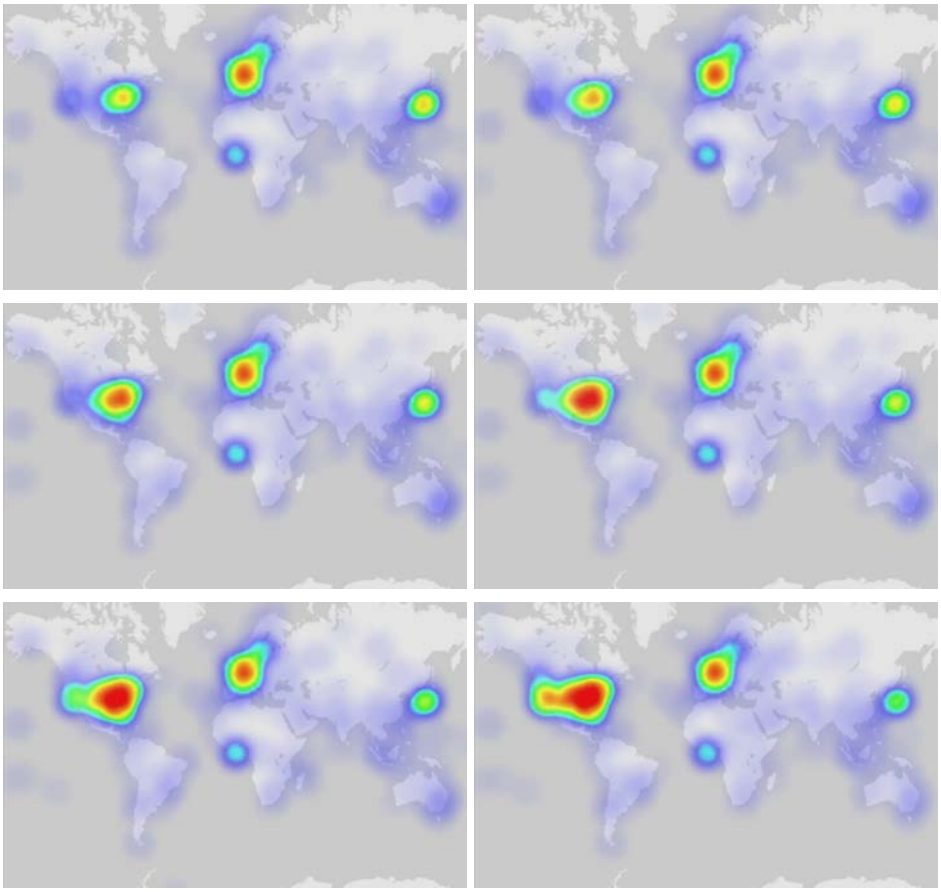
Table 5.2: Time cost of clutter map [114] and visual saliency map.

Cases	Canvas size	Method	Time cost (ms)
SGM Sec. 5.6.1	876×374	Clutter map [114]	3441
		Our method	472
GGM Sec. 5.6.1	876×414	Clutter map	3813
		Our method	525
GC Sec. 5.6.1	1500×800	Clutter map	12177
		Our method	1749
IG Sec. 5.6.1	678×680	Clutter map	4631
		Our method	653
Hex Points	750×495	Clutter map	3762
		Our method	534
Graphs	600×363	Clutter map	2370
		Our method	303
Social Network	600×299	Clutter map	1973
		Our method	254
Heat-map	1126×563	Clutter map	6477
		Our method	943
China Map	1320×791	Clutter map	10456
		Our method	1553

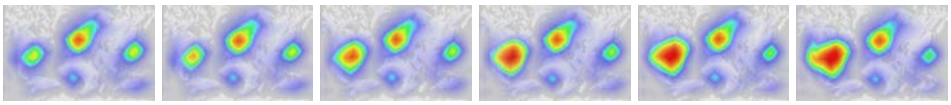
Although image energy preservation methods [9] are normally used in pixel-based retargeting approaches such as seam carving, they are also applicable to our method. The visual coherency can be evaluated quantitatively by calculating the percentage of energy preservation. We use the saliency of each pixel instead of the gradient to calculate the preserved energy because the presented method is based on a VSM. The average saliency of the pixels is shown in Fig. 5.10. The figure shows the saliency preservation abilities of the four strategies and demonstrates that our method can effectively preserve the saliency content in the visualization while resizing the canvas.

5.7 Conclusion

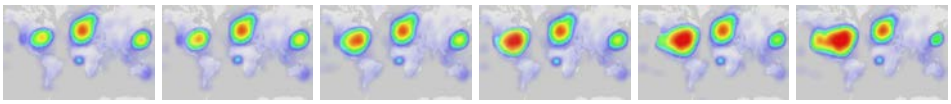
In this chapter, we present an adaptive triangle-based approach for the content-aware resizing of information visualizations, especially dynamic information visualizations. We propose a visual saliency detector that follows a set of criteria. The detected VSM is used not only to generate adaptive meshes but also to calculate the deformation factor of each triangle. A robust resizing energy function is defined to implement mesh resizing. We also extend the resizing model in a stream-aware form to achieve a smooth focus+context stream visualization. The experiments show that our method has the potential to be used effectively in the re-targeting of information visualizations.



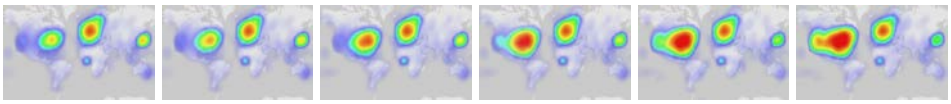
(a) 6 frames of time-varying heatmaps (1136×700).



(b) Resizing result using seam carving (400×250).



(c) Resizing result using our method without stream-aware consideration (400×250).



(d) Resizing result using our method with stream-aware consideration (400×250).

Figure 5.16: Comparison of the resizing results of the time-varying frames using different approaches.

CHAPTER 6

INTERACTIVE VISUAL QUERYING OF STREAMING DATA

6.1 Introduction

There is a growing need for interactive querying with the increase in spatio-temporal data applications. Basic requirements for simultaneous variation analysis and visualization are necessary due to the demand in spatio-temporal data analytics, particularly in dynamic data mining and prediction. We assume that spatio-temporal data are sequences of data that can be collected in real-time through physical sensors, such as vehicles, mobile phones, and climate monitors. Because the majority of spatio-temporal data include unorganized structures and a large variety of features, querying significant patterns and trends requires high-performance computing and accurate pattern matching.

A number of visual query systems have been presented before. Correll and Gleicher [32] introduced a complete sketch-based visual query framework for the understanding and exploration of time series data. However, they only focused the visual query on one-dimensional space. The query issues on two-dimensional space, such as a geographical map, are appearing in many real applications. Liu et al. [90] provided a fast approach to query big data in real-time on two-dimensional space through rectangle-based interactions. However, the work of Liu et al. [90] requires further adjustment to query a special pattern in the time series data. In addition, relational queries require a new visualization paradigm to make use of the native data properties and features. Simply listing the querying results as a histogram cannot fulfill the spatio-temporal querying requirements because of missing time sequence and feature details.

To address the above issues, we present a new saliency-based framework that provides a more intuitive querying interaction to better identify regions of interest. Our first contribution is a saliency map structure that helps in organizing time-varying spatial data. A pair of querying interactions is presented to query patterns on a sequence of saliency maps. We then describe methods to address the challenges of analyzing and visualizing query results, such as pattern flow and similarities. Experimental results obtained on two real datasets demonstrate that SalQuery provides an intuitive and effective interaction and visualization on large spatio-temporal datasets.

The remaining sections are organized as follows. Section 6.2 describes saliency map generation in detail. Section 6.3 shows the querying interactions. Section 6.4 describes the visualization designs for the query results. Section 6.5 addresses the implementation of our framework and presents the results obtained on two real datasets. Section 6.6 discusses the results and the performance. Section 6.7 draws the conclusion.

6.2 Saliency Definition and Generation

As shown in [60], Itti et al. define saliency as a set of points that can immediately attract a viewer's attention. Then, researchers such as Achanta et al. [1] and Goferman et al. [49] further define saliency as a set of regions, which consider the multiple visual features in images. Based on the theory of saliency, we assume that saliency indicates the most important parts of the spatio-temporal data. We use the KDE [125] to estimate the saliency map of the input data set. We assume that the input data are two-dimensional spatio-temporal data. The elements of the input are locations (points) and timestamps. Hence, spatio-temporal data in a period can be represented as a sequence of saliency maps. Beyond the saliency map, we present a new structure called saliency block (SB) to describe abundant features of the saliency regions and their relationships. The generation of saliency maps and blocks will be beneficial to the further visual querying method.

6.2.1 Saliency map

KDE provides a smoother result than using histograms, as discussed in [125]. Because traditional density computing methods such as k-means clustering require time-consuming calculations, kernel-based approximation has been devised to estimate the density of data. We adopt the Gaussian function $G(t) = \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}$ as a kernel because of its smoothness and parameterization. Hence, the KDE method is formulated as follows:

$$K(x,t) = \frac{1}{n_t} \sum_{j=1}^{n_t} \frac{1}{r_j} G(\frac{|x - x_j|}{r_j}), x_j \in S_t, x \in S_t \tag{6.1}$$

where n_t is the number of points in the data set S_t, t indicates the timestamps, x_j is one of the data points in S_t, and r_j is a kernel size parameter of x_j. r_j is defined according to the feature of r_j in a real application. Further discussion of KDE can follow a fundamental concept in statistics as shown in [125]. K(x) will be called n_t times to generate a density map of S_t. Although the calculation of KDE is still time-consuming, it is easy to be approximated through n_t times Gaussian circle rendering with alpha blending.

Figure 6.1 shows a simple example of density map generation. We assume that each saliency map has the same width and height. In our experiments, all of the saliency maps will be generated in advance and stored as gray-scale images. We define a saliency map in a timestamp as S_t. Figure 6.2(a-b) shows a saliency map generated from a frame of raw data.

- Connected Component Map

The generated saliency map can be considered to be an image. To query the salient regions, we present a binary saliency detection method to label the pixels in a saliency map. Each pixel will be labeled with two values, such as s_{ir} and s_{ic}. s_{ir} indicates "Is a pixel i in a saliency region?", and s_{ic} means "Which connected component does a pixel belong to?". First, a binary saliency map could be generated via a thresholding setting on a saliency map. If the pixel value in a saliency map is larger than δ, then

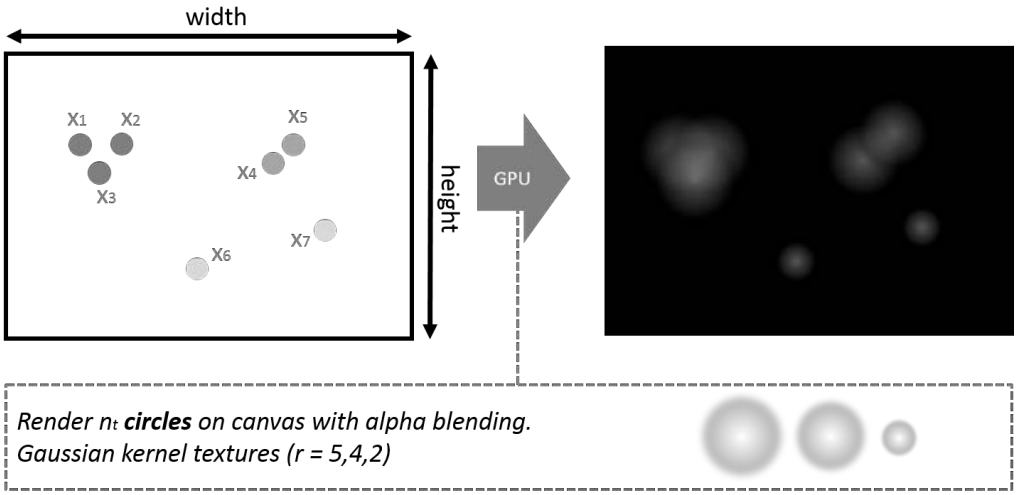


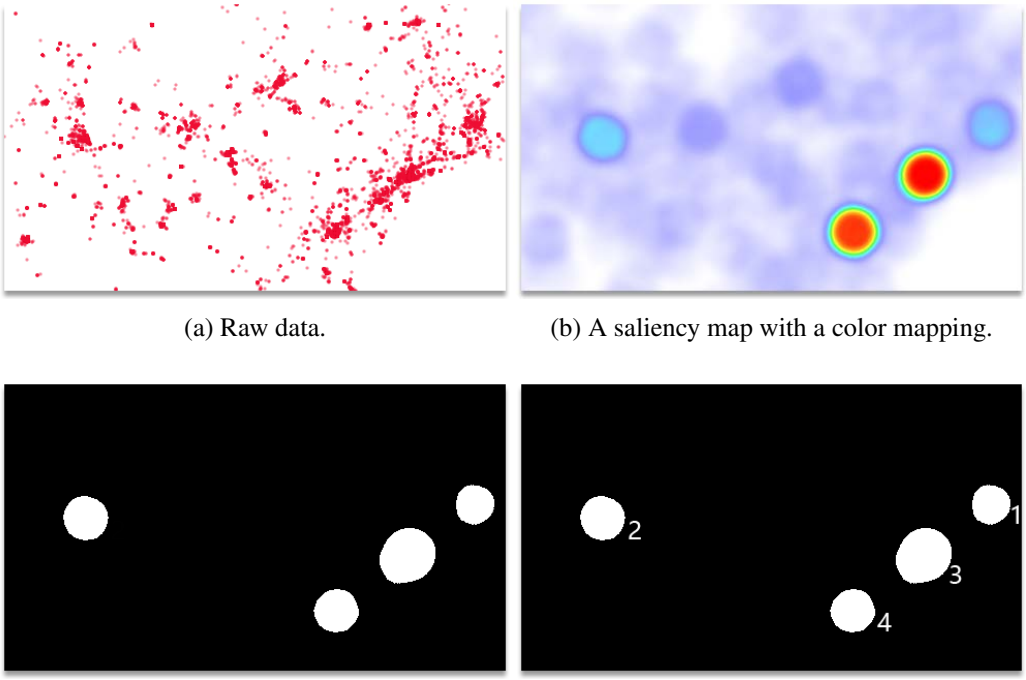
Figure 6.1: An example of kernel density estimation. The left canvas lists some raw points, whereas the right canvas shows the result of KDE. The different kernel sizes are listed at the bottom. This estimation process can be approximated through circle rendering.

s_{ir} is equal to 1. Otherwise, s_{ir} is equal to 0. We set δ as 0.5 in our experiments. Figure 6.2(c) shows an example of a binary saliency map generated through calculating s_{ir} of a saliency map.

Based on the binary saliency map, connected components can be found using sequential operations [113] that are effective for labeling the index of each region. There are two passes for the detection of connected components. The first pass is to scan the pixels in the binary saliency map from top to bottom and find the successive component for each row. Different components will be labeled with different component indices. Since it is unnecessary to traverse all pixels repeatedly, saliency region detection is not time-consuming. By using sequential operations, we can assign s_{ic} with different saliency component indices, as shown in Figure 6.2(d).

6.2.2 Saliency block

A saliency encoding method called saliency block is presented in this subsection to help to analyze the features of a saliency map and accelerate the querying process. We define a saliency block (SB) as a set of features of a saliency region. A number



(a) Raw data.

(b) A saliency map with a color mapping.

(c) A binary saliency map.

(d) A labeled connected component map.

Figure 6.2: The components of a saliency map.

of saliency blocks will be generated based on the processing of the saliency map. A saliency block includes many features, such as saliency boundary box, data number, key point, variation state, variation direction, last related saliency block index (LSBI), and next related saliency block index (NSBI). The features of a saliency block can be enriched according to different requirements from the real applications. We use $\boldsymbol{SB} = \{SB_i\}, i \in [1, sn]$ to indicate a set of saliency blocks in a saliency map with timestamp t, where sn indicates the saliency block number. Figure 6.3 shows a description of a saliency block.

- Boundary box (SB_{bb})

SB_{bb} is a rectangle that describes the boundary box of a saliency region. The rectangle can be calculated by finding left-top and right-bottom points.

- Boundary hull (SB_{bh})

A concave hull of the saliency region will be calculated according to the work presented in [98]. A concave hull can use fewer markers to approximate an accurate

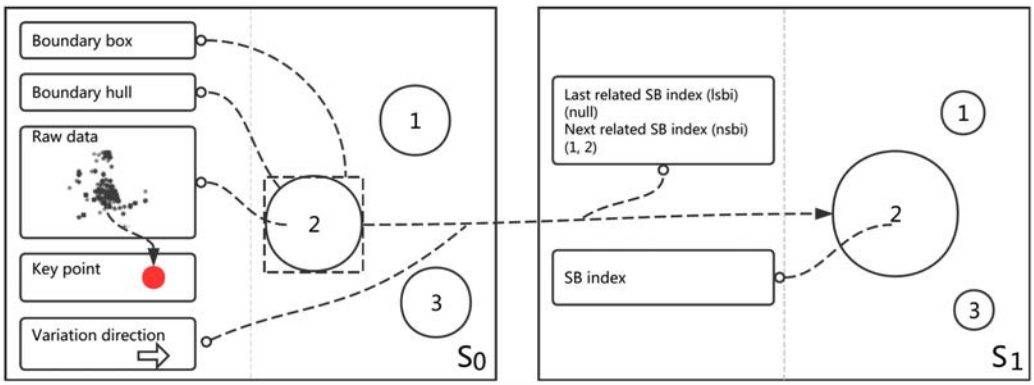


Figure 6.3: A description of a saliency block. S_0 and S_1 indicate two time-close saliency maps. Circles indicate the saliency regions.

region boundary.

- Data number (SB_{dn})

SB_{dn} is used to indicate the data number in the saliency region. This value is related to the raw data and will be visualized in the further data representation.

- Key point (SB_{kp})

SB_{kp} is used to indicate the most salient point in the saliency region. We use the steepest descent method to find the key point of a saliency region.

- Variation state (SB_{vs})

SB_{vs} outlines the variation state of the saliency region between two time steps. One of two states, namely, increment or decrement, will be assigned. If the data number is larger than the related saliency region at the last time step, then SB_{vs} will be set as increment. Otherwise, SB_{vs} will be set as decrement.

- Variation direction (SB_{vd})

SB_{vd} is addressed to describe the variation direction of a saliency region. To simplify the variation direction calculation, we divide the related boundary box into 9 average

sub-regions. Then, the variation states of these 9 sub-regions will be calculated. The surrounding 8 variation states can represent the variation direction of the saliency region. Finally, we will assign SB_{vd} with a two-dimensional vector.

- LSBI and NSBI (SB_{lsbi} and SB_{nsbi})

To link with the time-close saliency block, SB_{lsbi} and SB_{nsbi} are outlined. If a saliency block in the last time period saliency map is overlapping with the current saliency block, we will assign this SB to SB_{lsbi}. Similarly, an overlapping SB in the next time period saliency map will be assigned to SB_{nsbi}. If no overlapping SB can be found, SB_{lsbi} or SB_{nsbi} will be assigned with null. We assume that LSBI or NSBI can be greater than one.

6.3 The Querying Method

We address a pair of querying interactions as shown in Figure 6.4. Brush-based querying is provided to freely select multiple regions on the saliency map. Block-based querying is another querying interaction that allows users to select the most important regions on a saliency map and query similar saliency blocks in a period. Prior to querying, the user should select a timestamp to see the raw data or the related saliency map to start the querying. All of the querying inputs will be generated through the saliency map that is currently being viewed.

6.3.1 Brush-based querying

Traditional rectangle-based querying requires improvement to select accurate regions, particularly when the querying regions are along coastlines or rivers on a geographical map. Our brush-based querying (BRQ) is inspired by the attribute signatures [134], but we focus on time-based evolution visualization and similarity analysis. BRQ will generate a brushing trajectory that includes a set of points.

(a) Brush-based querying (BRQ) on a raw data map.

(b) Brush-based querying (BRQ) on a saliency map.

(c) Block-based querying (BLQ) on a saliency map.

Figure 6.4: Two querying interactions. BRQ interaction can be applied on a raw data map or a saliency map. BLQ can only be applied on a saliency map, and the selected saliency region will be highlighted with a red contour.

We define the points with inflections as special points, which will be detected through the second derivative of the brushing trajectory. If the second derivative of a point on the brushing trajectory is not close to 0, we define it as a special point. Saliency point detection has the advantage of improving performance. Because a drawing includes numerous points, which may lead to time-consuming variation calculations and comparisons, we reduce the complexity of a brushing trajectory and only store the special points. BRQ can be applied on a raw data map or a saliency map, as shown in Fig. 6.4(a-b). Red circles address the special points on a brushing trajectory. We define the saliency values on the red circle location as a spatial chain $SC_t = [s_{t1}, s_{t2}, ..., s_{tn}]$, where t indicates the timestamp of a saliency map. We define the currently viewed spatial chain as SC_{in}, which will be the input condition of the further matching.

- Spatial Chain Matching

Spatial chain matching is an operation that can find the similar SC_t that belong to a sequence of saliency maps compared with the input SC_{in}. It is beneficial to find matched results from massive historical data through a simple brush on the map. Figure 6.5 shows a description of the spatial chain matching. The numbers at the bottom indicate the indices of the detected special points. The green chain in Figure 6.5 outlines SC_{in}, which is the saliency values selected by a user through brush-based querying. The other three spatial chains are the candidates in other timestamps.

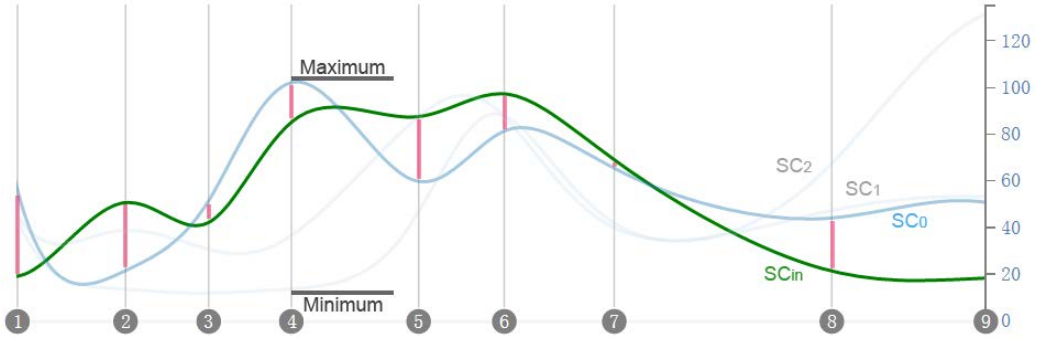


Figure 6.5: A description of the spatial chain matching.

The process of spatial chain matching includes three steps. First, peak values such as the maximum value and minimum value should be calculated in advance for each special point. Figure 6.5 shows an example of peak values at the fourth special point. Two high-dimensional vectors, SC_{max} and SC_{min}, are defined to store the peak values. Second, we calculate the similarities between the input and each spatial chain candidate. The spatial chain matching process can be formulated as follows:

$$SCM(SC_{in}, SC_t) = \frac{1}{k}\sum_{i=1}^{i\leq k} 1 - \frac{|SC_t(i) - SC_{in}(i)|}{|SC_{peak}(i) - SC_{in}(i)|} \tag{6.2}$$

$$SC_{peak}(i) = \begin{cases} SC_{\max}(i), SC_t(i) - SC_{in}(i) > 0 \\ SC_{\min}(i), SC_t(i) - SC_{in}(i) < 0 \end{cases} \tag{6.3}$$

where SCM defines the similarity function of spatial chains, $SC_{peak}(i)$ ensures that the range of the similarity is from 0.0 to 1.0, and k is the number of the special points. After calculating the similarities, matched high similarity timestamps will be visualized.

6.3.2 Block-based querying

In block-based querying (BLQ), we will check whether a touch occurs on the saliency regions. All of the touched saliency regions and their saliency blocks will be the querying input. If a user clicks on or touches a saliency regions, the entire region will be highlighted with a non-linear red contour, as shown in Figure 6.4(c). We adopt the concave hull method as mentioned in [98] to generate the contour. BLQ is

designed to interact with the saliency blocks. All of the block features are considered in the querying, such as saliency boundary box, data number, key point, variation state, variation direction, last related saliency block index (LSBI), and next related saliency block index (NSBI). The similarities of features will be calculated to find the most matched results through saliency block matching.

- Saliency Block Matching

Saliency block matching (SBM) is a hybrid matching method, and it is more complex than a simple image comparison as mentioned in [78]. As previously mentioned, the related saliency block will be an input to query the matching data from the candidates. Because a sub-saliency map in a block contains less obvious feature points, image-based matching such as SIFT matching [92] may not be a suitable solution. Perceptual hashing [65] is another method to match the saliency, but it requires a fixed region size. Based on the saliency block structure, we present the SBM method for evaluating the similarities of saliency blocks. SBM can be formulated as follows:

$$SBM(SB_{in}, SB_t) = \sum_{j=1}^{j \leq m} \lambda_j sim_j(SB_{inj}, SB_{tj}), sbm_j \in \boldsymbol{sbm}, \lambda_j \in \boldsymbol{\lambda} \tag{6.4}$$

$$\boldsymbol{sbm} = \{sbm_{bb}, sbm_{dn}, sbm_{kp}, sbm_{vs}, sbm_{vd}, sbm_{lsbi}, sbm_{nsbi}\} \tag{6.5}$$

where $m = 7$ means the feature number of the saliency block, j indicates the feature index, λ_j means the weight of each feature, and $\boldsymbol{sbm}$ defines the sub-matching function set. In our experiment, we define $\boldsymbol{\lambda}$ as $\{0.5, 0.1, 0.15, 0.075, 0.075, 0.05, 0.05\}$. The value of $\boldsymbol{\lambda}$ can be adjusted according to different requirements, but the sum of λ_j should be 1.0 to ensure that $SBM \in [0.0, 1.0]$.

The first step of SBM is to calculate the similarity of two boundary boxes sbm_{bb} that belong to two saliency blocks. sbm_{bb} is related to the crossing area between two boxes; hence, we define it as $sbm_{bb}(x, y) = \frac{Area_{cross}}{Area_x + Area_y - Area_{cross}}$. sbm_{dn} indicates the similarity of the raw data number. It can be calculated as $sim_{dn}(x, y) = 1.0 -$

$\frac{|x-y|}{x}$. sbm_{kp} outlines the similarity of the key points. We define it as $sim_{kp}(x,y) = 1.0 - \frac{|x-y|}{255}$.

sbm_{vs} and sbm_{vd} outline the trend similarity of two saliency blocks. Trend-based querying has not been discussed in the prior querying work such as [90]. However, it is useful for spatio-temporal data querying because the variation feature provides potential information. Various studies such as feature tracking [157, 28] in scientific visualization have discussed the usage of the variation. Because the variation state only has two values, we define the similarity of two variation states as $sbm_{vs}(x,y) = \begin{cases} 0, x \neq y \\ 1, x = y \end{cases}$. The similarity of two variation directions can be formulated as $sbm_{vd}(x,y) = \cos(\frac{x \cdot y}{|x||y|})$ using a cosine similarity calculation.

sbm_{lsbi} and sbm_{nsbi} are defined as $sbm_{sbi}(x,y) = \begin{cases} 0, x \neq y \\ 1, x = y \end{cases}$ where x and y mean the numbers of linked blocks. Finally, we can calculate the similarity of two saliency blocks using Eq. 6.4.

6.4 Visualization Design

Query result visualization is an important part of SalQuery. We design two representations to visualize the query results. The first one provides an overview of the flow information related to the selected regions. The second one shows the candidates with high similarities.

6.4.1 Flow-oriented representation

Flow-oriented representation (FOR) is a flow-based design to show the saliency evolution in a period. We design two types of FORs according to our two interactions. $Flow_{sal}$ is designed to visualize the saliency flow. $Flow_{sal}$ includes four parts, as shown in Figure 6.9(a). The left side indicates the timestamps of the candidates. The right side is the axis of the saliency values. The non-linear arranging special points

with indices are listed at the bottom. The distance scale of two special points is according to their distance on the map. All of the spatial chains in different timestamps are rendered as curves with blue color and red color, where red color indicates the input saliency chain. The range of the time span can be adjusted by the user. $Flow_{sal}$ can be used to compare the data saliency in different regions and different times.

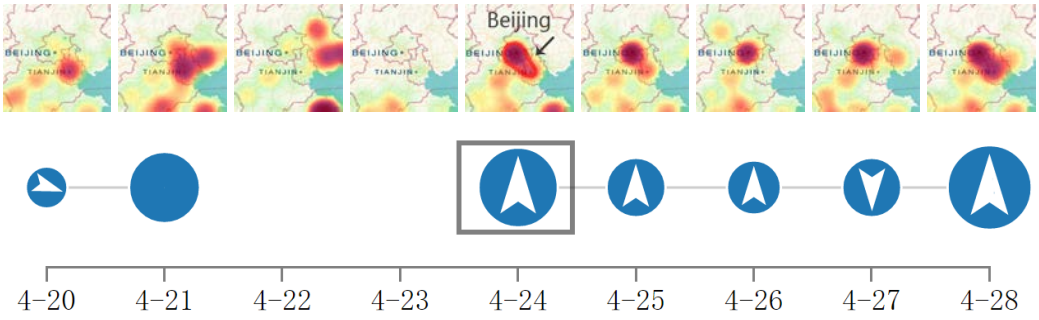


Figure 6.6: An example of $Flow_{block}$ and corresponding saliency snapshots. (top) Saliency air pollution snapshots in Beijing from April 20th, 2016, to April 28th, 2016. (bottom) A flow representation that visualizes the variation of the air pollution in Beijing.

For the block-based querying interaction, we design $Flow_{block}$ as shown in Figure 6.6 to obtain further insights of the saliency regions. In the representation of $Flow_{block}$, the main features of the saliency blocks will be visualized through a bubble-flow. Each bubble indicates a saliency block on the saliency map. A bubble will be connected with a neighbor bubble according to the saliency block linking state. The size of a bubble corresponds to the area of the saliency block. The variation direction of the saliency region has not been considered in traditional time-line visualization. Hence, we enhance the time-line-style visualization [68] via adding a directional icon in the center of a bubble. No bubble along the y-axis indicates that there is no salient region at the corresponding timestamps. For example, no bubble on April 23 in Figure 6.6 means no saliency on the selected region. The bubble with no further linking node shows that the salient region is going to disappear. A related example is on April 21 in Figure 6.6. If a saliency block is divided into two small saliency blocks in the next timestamps, two linking bubbles will be generated. In summary, $Flow_{block}$ can not only show the overview of a sequence of saliency maps

but also visualize the hidden details, such as variation state, direction, appearing, disappearing, combination, and division.

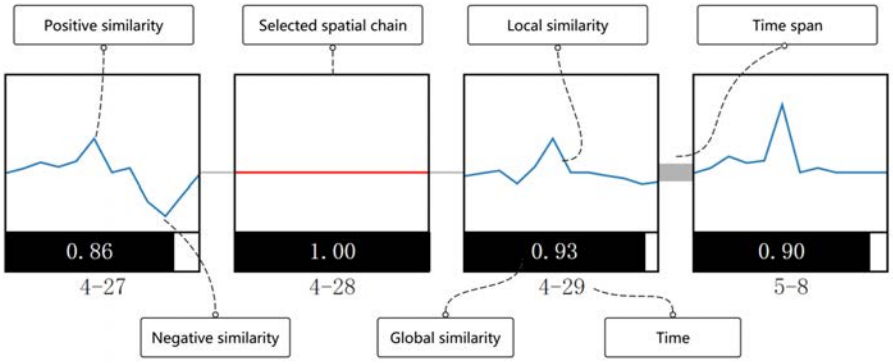


Figure 6.7: A description of the Sim_{curve} design.

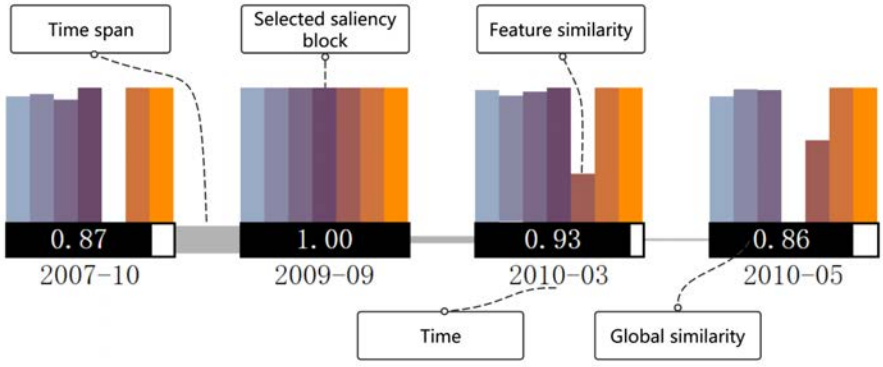
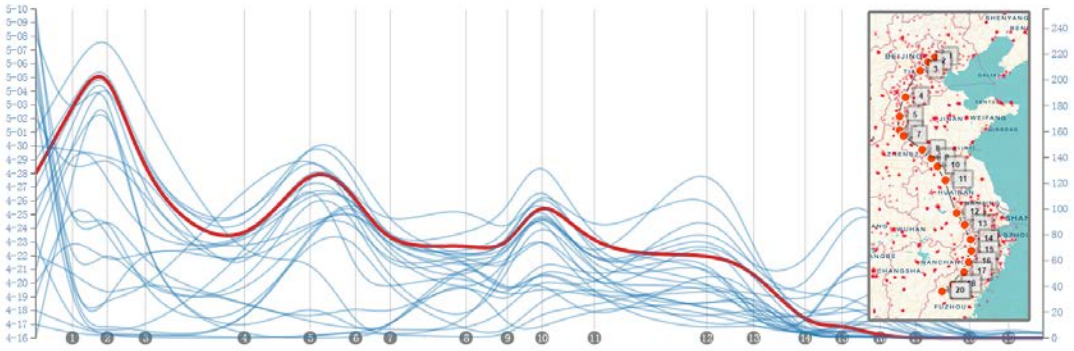


Figure 6.8: A description of the Sim_{block} design.

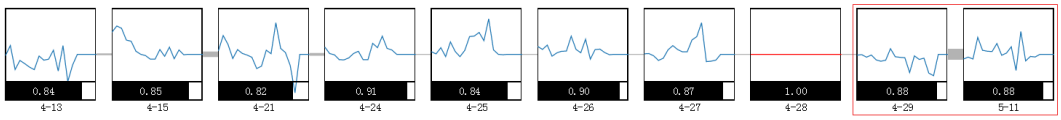
6.4.2 Similarity-oriented representation

We find that bar chart visualization, as shown in VisQuery [78], is limited to the query result visualization. A bar chart can only present simple information, such as a date and a value. We design a new similarity-based visualization called similarity-oriented representation (SOR) that provides abundant information for further analysis and decision.

Similarly, we design two types of SORs according to our two interactions. We first design a small-size curve Sim_{curve} as shown in Figure 6.7 to visualize the similarity of the special chain in a timestamp. The similarities of all special points are



(a) The $Flow_{sal}$ representation shows the air pollution values on 20 regions in 25 days.



(b) The Sim_{curve} representation shows the most similar air pollution distribution in 25 days.

Figure 6.9: Brush-based querying of air pollution from north to south regions in China.

listed on this curve. We define the similarity of a special point as a local similarity and the entire special chain similarity as a global similarity. There are two types of saliency similarities: positive and negative. In the final visualization, we only show the information in a timestamp with higher global similarities. We use different widths of the linking line, as shown in Figure 6.7, to indicate the time span.

For the block-based querying interaction, the main task is to visualize many saliency regions with multiple features. Therefore, we designed Sim_{bar} as shown in Figure 6.8 to visualize groups of feature similarities through multiple bars. Seven bars indicate seven feature similarities between a saliency block and a selected saliency block. The connection style of Sim_{bar} is similar to that of Sim_{curve}.

6.5 Experiments and Discussion

The experiments are implemented on a MacBook Pro with an Intel Core i7 CPU and 16 GB RAM. The raw spatio-temporal data and saliency blocks are managed through *MongoDB*, which is a structured storage database. We adopt D3 [21] to provide interactions and visualize the query results. All saliency maps are visualized through color mapping on the geographical map. The color mapping themes are shown in the

(a) 3378 air quality monitoring stations in China and other countries.

(b) An air pollution saliency map on May 1st, 2016, in China.

Figure 6.10: A visualization example of the air pollution data.

right parts of Figure 6.10(b) and Figure 6.11(b).

6.5.1 Air Pollution (AP)

We found that the air pollution influences the lives of individuals. To help users query the air pollution distribution, we apply our framework to a real air pollution (AP) dataset, as shown in Figure 6.10. This dataset provides the air quality index (AQI), such as PM2.5, in China. They were collected from aqicn.org over 32 days from April 12th, 2016, to May 13th, 2016. AQI data were monitored at 3378 monitoring stations. Each record includes an AQI value and corresponding monitoring location. A high AQI means a bad air quality. Thirty-two saliency maps were generated in advance through kernel density estimation. SalQuery provides an interactive querying to find the date with a similar AQI distribution on a geographical map compared the selected day.

We switch the viewing date on April 28th and use brush-based querying interaction to query the air pollution data from north to south regions in China, as shown in

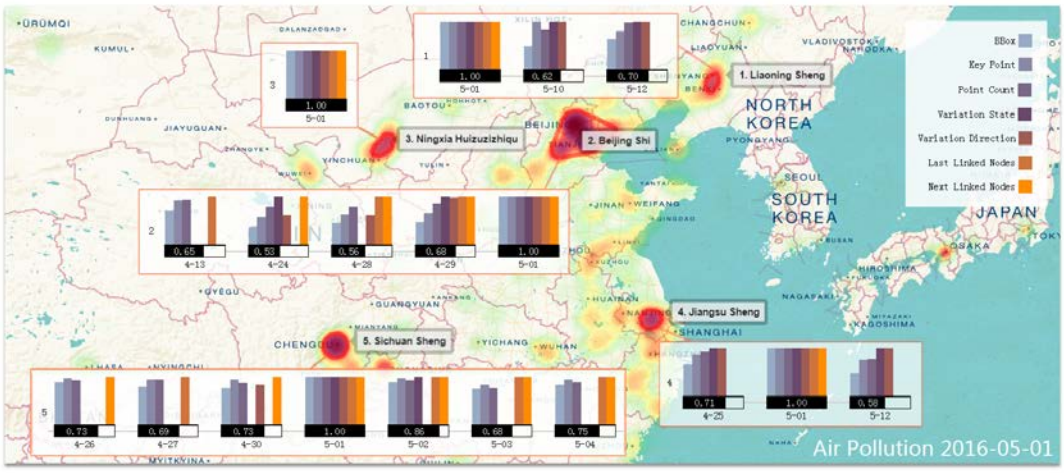
(a) 71831 photo locations on December 2012 in U.S.A.

(b) A photo location saliency on December 2012 in U.S.A.

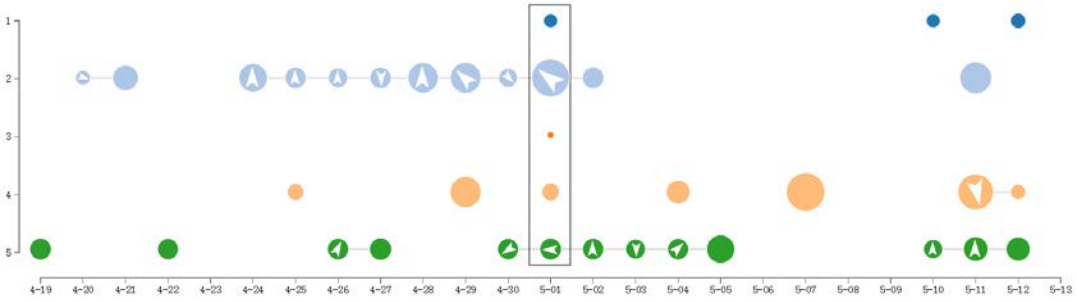
Figure 6.11: A visualization example of the photo location data.

Figure 6.9. Nearly 20 regions (20 special points) have been automatically selected. The $Flow_{sal}$ representation shown in Figure 6.9(a) shows the air pollution values in 20 regions in 25 days. The red curve indicates the input of the spatial chain on April 28th. From the $Flow_{sal}$, we can find that the air quality becomes increasingly better from north to south regions. The air quality of some regions, such as region 2 in Figure 6.9, was varying in a very large range over 23 days. Moreover, the Sim_{curve} representation presented in Figure 6.9(b) shows the most similar air pollution distribution over 25 days. Although the similarity of April 29th is equal to that of May 11th, their local similarities of special points are different. Most of the local similarities on April 29th are negative, which means that the air pollution is less than the selected date. By contrast, most of the local similarities on May 11th are positive, which means the air pollution is more than the selected date.

Another querying task is based on saliency regions of air pollution. On the May 1st saliency map, we select five saliency regions through touching. The evolution flow of the related saliency blocks is generated through the $Flow_{block}$ representation,



(a) The Sim_{bar} representation shows the most similar air pollution distribution dates and the related feature similarities.



(b) The $Flow_{block}$ representation shows the air pollution evolutions on five saliency regions.

Figure 6.12: Block-based querying of air pollutions on saliency regions in China.

as shown in Figure 6.12(b). Although the air quality of Ningxia (index is 3) on May 1st is not good, there is less air pollution on other days. For the region of Jiangsu (index is 4), the bad air quality occurrence is not continuous. The Sim_{bar} visualization shown in Figure 6.12(a) is a supplement of $Flow_{block}$. Sim_{bar} of the five regions show the most similar date and the similarities of seven features.

6.5.2 Flickr Photo (FP)

To further evaluate SalQuery, we applied it to a large-scale dataset called YFCC100m [130]. YFCC100m is an official dataset released by Yahoo! Flickr, which contains nearly fifty millions geotagged photos. We filter the YFCC100m dataset and only use the photos with locations in the U.S.A from Jan. 1st, 2007, to Feb. 28, 2014. Filtered YFCC100m can be considered to be a large-scale spatio-temporal dataset. The total

number of photo records that we used is 17 million. We aggregate the photo locations in a month into a saliency map. Hence, we generated 86 saliency maps according to the photo distributions among 86 months in the U.S.A. Figure 6.11 shows an example of the photo distribution in a month, which includes 71831 photo locations.

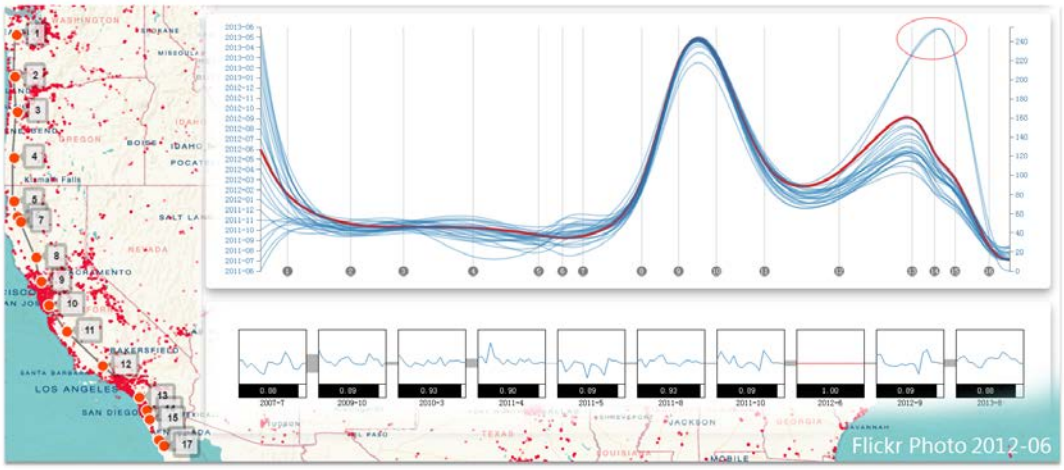
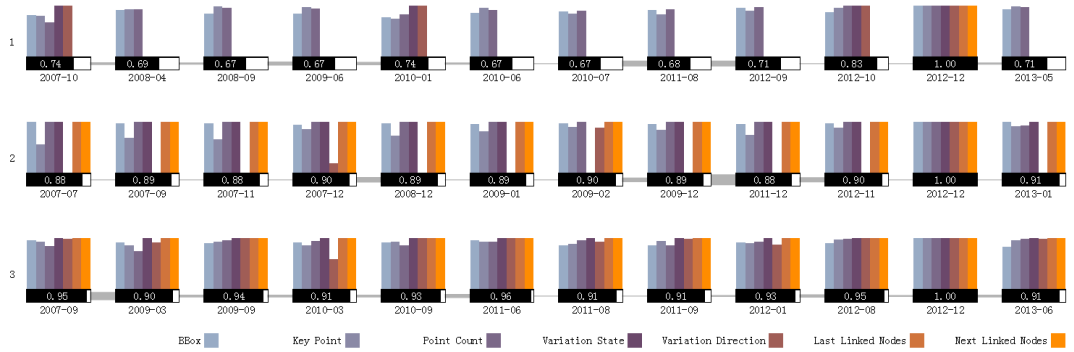


Figure 6.13: Brush-based querying of the photo distribution along a coastline in the western U.S.A.

It is easy to use SalQuery to query the photo distributions along coastlines or other special sequences of regions. Figure 6.13 shows photo distributions of region 17 over 25 months. We can find that most of the dates have similar photo distributions. Special distributions only occur on regions 13, 14 and 15, which are highlighted using a red ellipse.

Because SOR omits fewer similarity dates, it can show longer period information than FOR. As shown in the bottom of Figure 6.13, the dates with similar photo distributions have been listed. If the saliency regions were selected, SalQuery can show the evolution of them, as shown in Figure 6.14(a). We can find that the selected saliency regions keep stable photo distributions over two years. Sim_{bar} in Figure 6.14(b) overcomes the space limitation problem of FOR. We can observe time-line-based similar results in the entire period of the data. Space limitation problem means the maximum period of FOR is limited in a fixed value such as 25 because the screen display is limited.

(a) The $Flow_{block}$ representation shows the photo distribution evolutions on three saliency regions.



(b) The Sim_{bar} representation shows the dates with the most similar photo distribution and the related feature similarities.

Figure 6.14: Block-based querying of the photo distribution on saliency map in the eastern U.S.A.

6.6 Performance

SalQuery can support an interactive ratio querying over the spatio-temporal data. We evaluate the time cost of our brush-based querying method. We adopt different scales of the special points to evaluate the performance, namely, 10, 20, 40, 80, 160 and 320.

The resolution of the querying space is 1240×620 on the display. The datasets for measurement are AP and FP, which have been presented in Sec. 6.5. As shown in Figure 6.15, the brush-based querying can be finished in a nearly interactive time. Moreover, we can observe from Fig. 6.15 that when the count of special points is increasing, the time-cost of brush-based querying is still stable. The dataset of FP is large-scale, but the performance on it is similar to the dataset of AP. Therefore, we can conclude that brush-based querying is suitable for a large-scale spatio-temporal dataset.

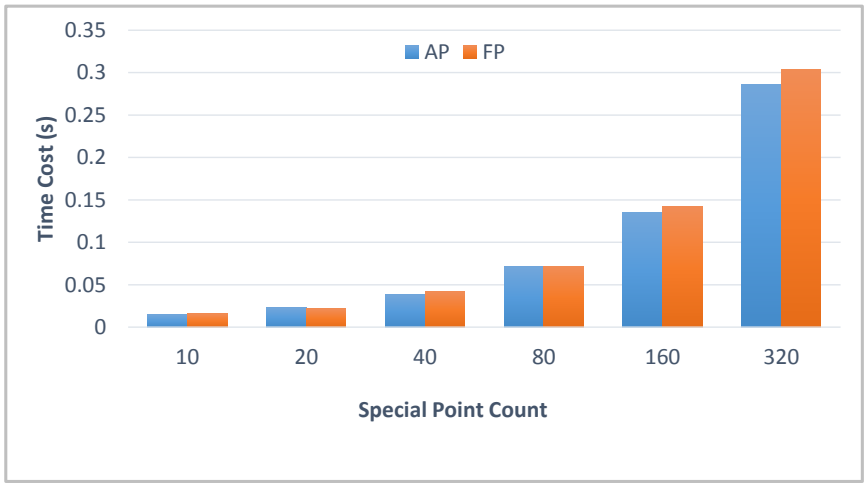


Figure 6.15: Performance of the brush-based querying.

6.7 Conclusion

We outlined a new framework with two interactions for interactively querying spatio-temporal data. A semi-automatic extraction of density features was addressed to aggregate the spatio-temporal data into saliency maps. Based on the saliency maps, we introduced two querying interactions. We also designed a pair of representations to visualize and explore the query results. Experimental results demonstrate that SalQuery can offer an intuitive and effective query on large-scale spatio-temporal datasets.

CHAPTER 7

CONCLUSION

This chapter concludes the dissertation. In the first section, a brief summary of the ideas and methodologies presented in the prior chapters are addressed. The limitations of our methods are then presented. In the last section, we propose future work.

7.1 Summing Up

Large-scale streaming data commonly exists in various real applications. For example, social network applications such as Twitter and Flickr receive numerous user locations every second, and air quality monitoring systems generate a large amount of air pollution data every hour. Visualizing the data structures and exploring the potential patterns is a significant task for interpreting large-scale time-series data.

Various approaches have been presented to solve large-scale information visualization problems. Visual clustering methods such as KDE and edge bundling are frequently used to reduce the visual overlapping among high-density data. Timeline techniques such as CloudLines and the Sankey flow diagram are proposed to solve the visualization problems of streaming data. Although these approaches have been applied to visualize data streams, there are still many critical challenges that must be investigated and addressed. For example, the sudden change between two stream frames is an unsolved problem; however, finding expected patterns in the data stream requires a feasible morphing technique. To help users comprehend the data patterns, we proposed a series of techniques for visualizing large-scale dynamic data through streaming. A brief introduction of the stream-based information visualization problems and our motivation for solving these problems were presented in Chapter 1.

Chapter 2 reviewed the prior research works on dynamic information visualization, especially on the methodologies of large-scale visualization frameworks, visual clustering, dynamic data representation, content-aware resizing, and visual querying. It was found that few visualization frameworks directly support dynamic information visualization. The stream processing frameworks such as Spark [163] do not address streaming data from a visualization perspective. This section categorized the visual clustering models into simplification, projection, bundling, and statistics. Various representation approaches of dynamic data were discussed in this chapter to find the unresolved problems. Various representation approaches were categorized as the interpolation method, variation feature tracking, and timeline visualization. Finally, the content-aware resizing of information visualization and the visual querying of the streaming data were discussed.

Chapter 3 presented a novel morphing framework, StreamMap, to smooth a pair of stream frames. StreamMap is based on a kernel density estimation method and a diffusion model. For the stream clustering, a similarity binning method and a super kernel density estimation model were presented to adaptively cluster dynamic high-density data. For the stream frame morphing, we proposed a robust morphing model to implement smooth morphing among different streams. The morphing model could also be used to represent the frame variation to find significant variation patterns. StreamMap has been demonstrated on three large-scale time-series datasets and evaluated using two quantifiable measurements.

Chapter 4 described a Module Graph approach to deal with large-graph data visualization. Module Graph either reduces the number of visualizing nodes or maintains the graph structure. By combining it with the idea of the super-graph, the SMG approach was presented to address the challenging issue of the streaming large-scale graph visualization. The Module Graph approach has been applied to various large-scale graph datasets and achieved positive results in either visual effectiveness or time performance. In addition, it was demonstrated that Module Graph could be used to represent spatial graph data with a simple modification of the module model.

Chapter 5 outlined a content-aware resizing model to make the streaming data visualization suitable for screen resolution variation via adaptive triangle implementation and energy optimization of the mesh deformation. We first introduced a multi-layer resizing framework to deal with the re-targeting problem related to the normal cases of information visualization, and we then presented a stream-aware approach to resize the dynamic visualization content with major content preservation. The visual effectiveness and time performance have been evaluated on various datasets.

Chapter 6 presented a visual querying method, SalQuery, for rapidly finding desired patterns among time-varying spatial data. Querying streaming data is becoming a dominant problem in big data analytics. We presented a new visual framework that provides a more intuitive querying interaction for streaming data by combining visual selections on patterns with image processing techniques and two querying interactions. Our experiments showed that this framework can provide an effective interactions for querying the desired patterns from spatio-temporal data.

7.2 Limitations

Although we have demonstrated the effectiveness of the presented methods for streaming data visualization, there are still some limitations.

First, StreamMap fails to address the following issues: (1) Although we adopted an adaptive bandwidth selection KDE method (SKDE) to estimate the density of the data, the accuracies of the density contour and peak still require improvement. In addition, current SKDE method cannot deal with different forms of data using the same parameters. Manual adjustment of the parameter for one frame in a dataset is required. (2) The accuracy of the morphing operation requires further improvement, particularly for handling a "moving" morphing pattern. (3) Current trend representations also suffer from over-plotting, which might affect the density pattern finding. Potential improvements with more sophisticated representations might be achieved through flow-visualization methods, such as OLIC [148] and IBFV [138]. (4) For

data sets without a "flow" nature, such as web portal logins and urban noise, smooth morphing works well; however, the trend representation will likely not present valid states.

Second, the proposed Module Graph method has not dealt with special graph data such as directed graphs, ego networks, fuzzy networks, and graphs with uncertainty. For directed graphs, the traditional community detection method requires modification to detect the high-accuracy communities. The graph pattern in an ego network is less than that in a general network, so the pattern design and selection should be further modified. The overlapping communities in a fuzzy network also hinder the wide usage of Module Graph. A graph with uncertainty is challenging for both the information visualization researcher and statistician; a combination of a graph visualization and a statistical method is a potential solution. In addition, our presented SMG approach is still at the beginning stage of the smooth dynamic graph visualization because the result of the super-graph based community detection of each graph frame is not highly accurate. An efficient community detection method in dynamic social networks such as that of Nguyen et al. [100] presents a potential direction for the SMG extension.

Third, since the content-aware resizing of the dynamic information visualization is very challenging, the stream-aware resizing model has not been completely performed and evaluated, especially when the layer number of the information visualization is large. In addition, if the visualization does not have a clear multi-layer designation, automatically classifying features of visual representations is required. The work of cartoon and texture decomposition [167] is a potential direction to abstract the structure of multiple layers in information visualization. Furthermore, our proposed saliency detection method is not of high accuracy, so a deep learning saliency map [81] can be adopted to guide the mesh deformation to further reduce the distortion.

Fourth, the current version of the visual querying approach has limitations in addressing the different levels of detail of the streaming data because saliency maps

at different levels of detail can exhibit high variations when using the current density estimation methods. In addition, the brush-based interaction has a limitation in result representation that the query result should be visualized on another panel. A new visual design that displays the query result directly on the geographical map will be a suitable solution. Bar diagrams and arrows can be visualized directly on the brushing path to make the trend representation more intuitive.

7.3 Areas of Further Research

This dissertation proposes a number of methods to deal with the problems of stream visualization. The current works of visual clustering, smooth morphing, resizing, and visual querying can be further investigated and extended in a number of directions.

1. The extension of the usage area of the visual clustering method on streaming data. Information visualization researchers have proposed many approaches to visualizing edge clouds, e.g., edge bundling [53] and smooth time-varying graph visualization [58]. In the edge bundling method, adjacent edges are visualized as a bundled group to reduce visual clutter. However, few methods have been presented with the specific purpose of visualizing data in related research areas such as image processing and computer vision. For example, the feature matching lines of two videos require a dynamic visual clustering approach. A comprehensive solution for visualizing video features and their corresponding pair matchings has not been investigated from an information visualization perspective. Therefore, we believe that a new visualization system for understanding and exploring video feature matchings to satisfy the requirements of domain researchers would be a breakthrough work.

2. The extension of the data class in the dynamic visualization system from single to multiple. The current visual clustering methods, such as SKDE and Module Graph, can be extended for dealing with multi-class streaming data. Multi-class dynamic data exists in various real applications such as air pollution visualization systems and Buckets [25]. Air pollution normally contains multiple pollutants such as $PM2.5$,

$PM10$, NO_2, and CO_2. The web system of Buckets shows a multi-class dataset related to the shooting indexes of NBA players. A visual analysis of the dynamic multi-class data will make the related application more convincing. Certainly, multi-class data visualization is still very challenging.

3. The extension of the smooth morphing model. The first potential significant extension is based on the deep learning (DL) method. Compared with the traditional computer vision methods, such as optical flow, the DL method has a higher morphing accuracy and a higher extension probability. For examples, Liao et al. [83] shows various convincing examples to demonstrate the effectiveness of DL in content transformation. Berthelot et al. [17] perform a smooth face morphing experiment to demonstrate the effectiveness of Generative Adversarial Networks (GAN). The two natures of the streaming data are similar to the training data of DL. The saliency map of the stream frame is similar to the image and the long period frames are large-scale, so it is possible to adopt the DL method to implement a better information morphing. The second potential extension is applying the morphing model on dynamic network data with uncertainty to visualize the network variation. Schulz et al. [120] visualized the probability distributions of a large-scale graph and extended their work to a spatial space. Since the relative distributions in the spatial space are fixed, it is possible to apply a smooth morphing model on sequences of probability distribution frames to visualize the graph variation. The third potential extension is proposing a coincident approach for three parts of StreamMap to speed up it for real-time processing on large-scale datasets. The peak-based clustering method [112] is a candidate.

4. The improvement of the stream-aware resizing framework without mesh warping. Although the solution of the video resizing issue is mature, similar approaches applying dynamic information visualization are still very challenging because the data structure type is more abundant than the image. Even if the mesh-based resizing model can reduce the content distortion rate while resizing, the layout deformation still precludes effective pattern finding. A new approach without content deformation, such as the work of Photo Collage [168, 87], will be a potential direction for the

improvement of the current content-aware visualization model. In addition, a learning based color mapping is a potential direction to retarget the content of the stream visualization as shown in the work of Poco et al. [107].

5. The improvement of the visual querying approach with higher accuracy and abundant interaction. In the future, a more flexible framework for querying different levels of detail will be considered. Moreover, forecast-based visualization via deep learning is a potential direction for the information visualization of streams. Wang et al. [141] propose a deep learning method to predict the traffic speed. Similarly, the stream data can be applied as a matrix and input into the deep learning network to predict further stream frames. In addition, with the increasing popularity of applications based on CNNs [67], a visual analysis tool for exploring and explaining convolutional stream features is expected to be beneficial to the deep learning and data mining communities. The sketch-based querying interaction [32] is another direction to improve SalQuery by converting the sketch from a one-dimensional space to a two-dimensional space.

REFERENCES

[1] Radhakrishna Achanta, Sheila Hemami, Francisco Estrada, and Sabine Susstrunk. Frequency-Tuned Salient Region Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR2009)*, pages 1597–1604, 2009.

[2] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[3] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, volume 27, pages 94–105, 1998.

[4] Alooma. https://www.alooma.com/live.

[5] Gennady Andrienko, Natalia Andrienko, Christophe Hurter, Salvatore Rinzivillo, and Stefan Wrobel. Scalable Analysis of Movement Data for Extracting and Exploring Significant Places. *IEEE Transactions on Visualization and Computer Graphics*, 19(7):1078–1094, 2013.

[6] Gennady Andrienko, Natalia Andrienko, Salvatore Rinzivillo, Mirco Nanni, Dino Pedreschi, and Fosca Giannotti. Interactive visual clustering of large collections of trajectories. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST 2009)*, pages 3–10. IEEE, 2009.

[7] Natalia Andrienko and Gennady Andrienko. Visual Analytics of Movement: An Overview of Methods, Tools and Procedures. *Information Visualization*, 12(1):3–24, 2013.

[8] Arvind Arasu, Brian Babcock, Shivnath Babu, John Cieslewicz, Mayur Datar, Keith Ito, Rajeev Motwani, Utkarsh Srivastava, and Jennifer Widom. Stream: The Stanford Data Stream Management System. *Book Chapter*, 2004.

[9] Shai Avidan and Ariel Shamir. Seam Carving for Content-Aware Image Resizing. *ACM Transactions Graphics*, 26(3):10:1–10:10, 2007.

[10] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and Issues in Data Stream Systems. In *Proceedings of the 21 ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 1–16, 2002.

[11] Benjamin Bach, Nathalie Henry Riche, Christophe Hurter, Kim Marriott, and Tim Dwyer. Towards Unambiguous Edge Bundling: Investigating Confluent Drawings for Network Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):541–550, 2017.

[12] Benjamin Bach, Conglei Shi, Nicolas Heulot, Tara Madhyastha, Tom Grabowski, and Pierre Dragicevic. Time Curves: Folding Time to Visualize Patterns of Temporal Evolution in Data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):559–568, 2016.

[13] George Baciu, Chenhui Li, Yunzhe Wang, and Xiujun Zhang. Cloudets: Cloud-based Cognition for Large Streaming Data. In *2015 IEEE 14th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC)*, pages 333–338, 2015.

[14] George Baciu, Chenhui Li, Yunzhe Wang, and Xiujun Zhang. Cloudet: A Cloud-Driven Visual Cognition of Large Streaming Data. *International Journal of Cognitive Informatics and Natural Intelligence*, 10(1):12–31, 2016.

[15] Stefano Baldassi, Nicola Megna, and David C Burr. Visual Clutter Causes High-magnitude Errors. *PLoS Biol*, 4(3):e56, 2006.

[16] Sebastiano Battiato, Giovanni Maria Farinella, Giovanni Puglisi, and Daniele Ravi. Saliency-Based Selection of Gradient Vector Flow Paths for Content Aware Image Resizing. *IEEE Transactions on Image Processing*, 23(5):2081–2095, 2014.

[17] David Berthelot, Tom Schumm, and Luke Metz. Began: Boundary equilibrium generative adversarial networks. *arXiv preprint arXiv:1703.10717*, 2017.

[18] Jacques Bertin. *Semiology of Graphics*. University of Wisconsin Press, 1983.

[19] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast Unfolding of Communities in Large Networks. *J. Stat. Mechtheory. E.*, 2008(10):P10008, 2008.

[20] Jean-Daniel Boissonnat, Olivier Devillers, Sylvain Pion, Monique Teillaud, and Mariette Yvinec. Triangulations in CGAL. *Computer Geometry*, 22(1-3):5–19, 2002.

[21] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[22] Joachim Böttger, Alexander Schäfer, Gabriele Lohmann, Arno Villringer, and Daniel S Margulies. Three-Dimensional Mean-Shift Edge Bundling for the Visualization of Functional Connectivity in the Brain. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):471–480, 2014.

[23] Matthew Brehmer, Bongshin Lee, Benjamin Bach, Nathalie Henry Riche, and Tamara Munzner. Timelines Revisited: A Design Space and Considerations for Expressive Storytelling. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):449–458, 2016.

[24] Gennady Andrienko Natalia Andrienko Tobias Schreck Bremm, Sebastian and Tatiana von Landesberger. Interactive Analysis of Object Group Changes over Time. In *EuroVA 2011: International Workshop on Visual Analytics*. The Eurographics Association, 2011.

[25] Buckets. http://buckets.peterbeshai.com. 2017.

[26] Nan Cao, Chaoguang Lin, Qiuhan Zhu, Yu-Ru Lin, Xian Teng, and Xidao Wen. Voila: Visual Anomaly Detection and Monitoring with Streaming Spatiotemporal Data. *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[27] Sangwon Chae, Aditi Majumder, and M Gopi. HD-GraphViz: Highly Distributed Graph Visualization on Tiled Displays. In *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing*, ICVGIP '12, pages 43:1–43:8, 2012.

[28] Yu-Hsuan Chan, Carlos D Correa, and Kwan-Liu Ma. Flow-based Scatterplots for Sensitivity Analysis. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology*, pages 43–50, 2010.

[29] Haidong Chen, Wei Chen, Honghui Mei, Zhiqi Liu, Kun Zhou, Weifeng Chen, Wentao Gu, and Kwan-Liu Ma. Visual Abstraction and Exploration of Multi-class Scatterplots. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1683–1692, 2014.

[30] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and Mobility: User Movement in Location-Based Social Networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090. ACM, 2011.

[31] Pio Claudio and Sung-Eui Yoon. Metro Transit-Centric Visualization for City Tour Planning. *Computer Graphics Forum*, 33(3):271–280, 2014.

[32] Michael Correll and Michael Gleicher. The Semantics of Sketch: Flexibility in Visual Query Systems for Time Series Data. In *2016 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 131–140, 2016.

[33] Michael Correll and Jeffrey Heer. Surprise! Bayesian Weighting for De-Biasing Thematic Maps. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):651–660, 2017.

[34] Joseph Cottam, Andrew Lumsdaine, and Peng Wang. Overplotting: Unified Solutions under Abstract Rendering. In *IEEE International Conference on Big Data*, pages 9–16, 2013.

[35] Joseph A Cottam, Andrew Lumsdaine, and Chris Weaver. Watch This: A Taxonomy for Dynamic Data Visualization. In *IEEE Conference on Visual Analytics Science and Technology (VAST 2012)*, pages 193–202, 2012.

[36] Stephan Diehl and Carsten Görg. Graphs, They are Changing. In *International Symposium on Graph Drawing*, pages 23–31. Springer, 2002.

[37] Sergey N. Dorogovtsev, Alexander V. Goltsev, and Jose F. Mendes. k-Core Organization of Complex Networks. *Physical Review Letters*, 96:040601, 2006.

[38] Fan Du, Nan Cao, Jian Zhao, and Yu-Ru Lin. Trajectory Bundling for Animated Transitions. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 289–298. ACM, 2015.

[39] Cody Dunne and Ben Shneiderman. Motif Simplification: Improving Network Visualization Readability with Fan, Connector, and Clique Glyphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 3247–3256. ACM, 2013.

[40] Tim Dwyer, Christopher Mears, Kerri Morgan, Todd Niven, Kim Marriott, and Mark Wallace. Improved optimal and approximate power graph compression for clearer visualisation of dense graphs. In *2014 IEEE Pacific Visualization Symposium (PacificVis)*, pages 105–112. IEEE, 2014.

[41] Ulrich Engelke, Hantao Liu, Junle Wang, Patrick Le Callet, Ingrid Heynderickx, Hans-jürgen Zepernick, Senior Member, and Anthony Maeder. Comparative Study of Fixation Density Maps. *IEEE Transactions on Image Processing*, 22(3):1121–1133, 2013.

[42] Ozan Ersoy, Christophe Hurter, Fernando Paulovich, Gabriel Cantareiro, and Alex Telea. Skeleton-Based Edge Bundling for Graph Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2364–2373, 2011.

[43] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD*, volume 96, pages 226–231, 1996.

[44] Kun-Chuan Feng, Chaoli Wang, Han-Wei Shen, and Tong-Yee Lee. Coherent Time-Varying Graph Drawing with Multifocus+Context Interaction. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1330–1342, 2012.

[45] Nivan Ferreira, Jorge Poco, Huy T. Vo, Juliana Freire, and Cláudio T. Silva. Visual exploration of big spatio-temporal urban data: A study of new york city taxi trips. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2149–2158, 2013.

[46] Martin Fink, Jan-Henrik Haunert, Joachim Spoerhase, and Alexander Wolff. Selecting the Aspect Ratio of a Scatter Plot Based on Its Delaunay Triangulation. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2326–2335, 2013.

[47] Simone Frintrop, Thomas Werner, and Martn Garca Germn. Traditional Saliency Reloaded: A Good Old Model in New Shape. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR2015)*, 2015.

[48] Ran Gal, Olga Sorkine, and Daniel Cohen-Or. Feature-Aware Texturing. In *Conference of the European Association for Computer Graphics*, pages 297–303, Nicosia, Cyprus, 2006. Eurographics Association.

[49] Stas Goferman, Lihi Zelnik-Manor, and Ayellet Tal. Context-Aware Saliency Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(10):1915–26, October 2012.

[50] Google. Color palette. `https://www.google.com/design/spec/style/color.html`.

[51] Sebastian Grottel, Guido Reina, Jadran Vrabec, and Thomas Ertl. Visual Verification and Analysis of Cluster Detection for Molecular Dynamics. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1624–1631, 2007.

[52] Jan-Henrik Haunert and Leon Sering. Drawing Road Networks with Focus Regions. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2555–62, 2011.

[53] D Holten. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[54] Danny Holten and Jarke J Van Wijk. Force-directed edge bundling for graph visualization. In *Computer Graphics Forum*, volume 28, pages 983–990, 2009.

[55] Berthold K Horn and Brian G Schunck. Determining Optical Flow. In *1981 Technical Symposium East*, pages 319–331. International Society for Optics and Photonics, 1981.

[56] Duen Horng, Polo Chau, Aniket Kittur, Jason I Hong, and Christos Faloutsos. Apolo : Making Sense of Large Network Data by Combining Rich User In-

teraction and Machine Learning. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (SIGCHI2011)*, pages 167–176, 2011.

[57] Yidan Hu, Guojun Dai, Jin Fan, Yifan Wu, and Hua Zhang. BlueAer: A Fine-Grained Urban PM2.5 3D Monitoring System Using Mobile Sensing. In *The 35th Annual IEEE International Conference on Computer Communications (INFOCOM 2016)*, pages 1–9. IEEE, 2016.

[58] Christophe Hurter, Ozan Ersoy, and Alex Telea. Smooth Bundling of Large Streaming and Sequence Graphs. pages 41–48, 2013.

[59] Christophe Hurter, Ozan Ersoy, and Alexandru Telea. Graph Bundling by Kernel Density Estimation. In *Computer Graphics Forum*, volume 31, pages 865–874, 2012.

[60] Laurent Itti, Christof Koch, and Ernst Niebur. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[61] Heike Jänicke and Min Chen. A Salience-based Quality Metric for Visualization. *Computer Graphics Forum*, 29(3):1183–1192, 2010.

[62] Yong Jin, Ligang Liu, and Qingbiao Wu. Nonhomogeneous Scaling Optimization for Realtime Image Resizing. *The Visual Computer*, 26(6-8):769–778, 2010.

[63] Peter Kaufmann, Oliver Wang, Alexander Sorkine-Hornung, Olga Sorkine-Hornung, Aljoscha Smolic, and Markus Gross. Finite Element Image Warping. *Computer Graphics Forum*, 32(2-1):31–39, 2013.

[64] Daniel A Keim, Christian Panse, Mike Sips, and Stephen C North. Visual Data Mining in Large Geospatial Point Sets. *IEEE Computer Graphics and Applications*, 24(5):36–44, 2004.

[65] Fouad Khelifi and Jianmin Jiang. Perceptual image hashing based on virtual watermark detection. *IEEE Transactions on Image Processing*, 19(4):981–994, 2010.

[66] Hye-Rin Kim, Min-Joon Yoo, Henry Kang, and In-Kwon Lee. Perceptually-based Color Assignment. *Computer Graphics Forum*, 33(7):309–318, October 2014.

[67] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[68] Milos Krstajic, Enrico Bertini, and Daniel Keim. CloudLines: Compact Display of Event Episodes in Multiple Time-series. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2432–2439, 2011.

[69] Milos Krstajic and Daniel A. Keim. Visualization of Streaming Data: Observing Change and Context in Information Visualization Techniques. *Proceedings of the IEEE International Conference on Big Data*, pages 41–47, 2013.

[70] Oh-Hyun Kwon, Chris Muelder, Kyungwon Lee, and Kwan-Liu Ma. A Study of Layout, Rendering, and Interaction Methods for Immersive Graph Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(7):1802–1815, 2016.

[71] Ove Daae Lampe and Helwig Hauser. Interactive Visualization of Streaming Data with Kernel Density Estimation. In *2011 IEEE Pacific Visualization Symposium (PacificVis)*, pages 171–178, 2011.

[72] Leaflet. http://leafletjs.com. 2013.

[73] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[74] Chenhui Li and George Baciu. Valid: A web framework for visual analytics of large streaming data. In *IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom2014)*, pages 686–692, Sept 2014.

[75] Chenhui Li, George Baciu, and Yu Han. Interactive visualization of high density streaming points with heat-map. In *2014 International Conference on Smart Computing*, pages 145–149, 2014.

[76] Chenhui Li, George Baciu, and Yu Han. StreamMap: Smooth Dynamic Visualization of High-Density Streaming Points. *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[77] Chenhui Li, George Baciu, and Yunzhe Wang. Modulgraph: Modularity-based visualization of massive graphs. In *SIGGRAPH Asia 2015 Visualization in High Performance Computing*, SA '15, pages 11:1–11:4, New York, NY, USA, 2015. ACM.

[78] Chenhui Li, George Baciu, and Yunzhe Wang. VisQuery: Visual Querying of Streaming Data via Pattern Matching. In *2016 Digital Media Industry Academic Forum (DMIAF)*, pages 161–165, 2016.

[79] Chenhui Li, George Baciu, and Yunzhe Wang. Module-Based Visualization of Large-Scale Graph Network Data. *Journal of Visualization*, pages 205–215, 2017.

[80] Chenhui Li, George Baciu, Yunzhe Wang, and Xiujun Zhang. Fast Content-Aware Resizing of Multi-layer Information Visualization via Adaptive Triangulation. *Journal of Visual Languages and Computing*, 2017.

[81] Guanbin Li and Yizhou Yu. Visual Saliency Based on Multiscale Deep Features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5455–5463, 2015.

[82] Jing Liao, Rodolfo S Lima, Diego Nehab, Hugues Hoppe, and Pedro V Sander. Semi-Automated Video Morphing. 33(4):51–60, 2014.

[83] Jing Liao, Yuan Yao, Yuan Yuan, Gang Hua, and Sing Bing Kang. Visual Attribute Transfer through Deep Image Analogy. *arXiv preprint arXiv:1705.01088*, 2017.

[84] Sharon Lin, Julie Fortuna, Chinmay Kulkarni, Maureen Stone, and Jeffrey Heer. Selecting Semantically-Resonant Colors for Data Visualization. *Computer Graphics Forum*, 32:401–410, 2013.

[85] Shih-Syun Lin, Chao-Hung Lin, Yan-Jhang Hu, and Tong-Yee Lee. Drawing Road Networks with Mental Maps. *IEEE Transactions on Visualization and Computer Graphics*, 20(9):1241–1252, 2014.

[86] Lauro Lins, James T Klosowski, and Carlos Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2456–2465, 2013.

[87] Lingjie Liu, Hongjie Zhang, Guangmei Jing, Yanwen Guo, Zhonggui Chen, and Wenping Wang. Correlation-Preserving Photo Collage. *IEEE Transactions on Visualization and Computer Graphics*, PP(99):1–13, 2017.

[88] Shixia Liu, Yingcai Wu, Enxun Wei, Mengchen Liu, and Yang Liu. StoryFlow: Tracking the evolution of stories. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of IEEE InfoVis 2013*, 19(12):2436–2445, 2013.

[89] Xiaotong Liu, Han-Wei Shen, and Yifan Hu. Supporting Multifaceted Viewing of Word Clouds with Focus+Context Display. *Information Visualization*, page 1473871614534095, 2014.

[90] Zhicheng Liu, Biye Jiang, and Jeffrey Heer. imMens: Real-time Visual Querying of Big Data. *Computer Graphics Forum*, 32(3):421–430, 2013.

[91] Dionysios Logothetis and Ken Yocum. Wide-Scale Data Stream Management. In *Usenix Annual Technical Conference*, pages 405–418, 2008.

[92] David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[93] Yafeng Lu, Michael Steptoe, Sarah Burke, Hong Wang, Jiun-Yi Tsai, Hasan Davulcu, Douglas Montgomery, Steven R Corman, and Ross Maciejewski. Exploring evolving media discourse through event cueing. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):220–229, 2016.

[94] James MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[95] Dhruv Mahajan, Fu-Chung Huang, Wojciech Matusik, Ravi Ramamoorthi, and Peter Belhumeur. Moving Gradients: a Path-based Method for Plausible Image Interpolation. *ACM Transactions Graphics*, 28(3):1–11, 2009.

[96] Adrian Mayorga and Michael Gleicher. Splatterplots: Overcoming Overdraw in Scatter plots. *IEEE Transactions on Visualization and Computer Graphics*, 19(9):1526–38, September 2013.

[97] Vladimir Molchanov, Alexey Fofonov, and Lars Linsen. Continuous Representation of Projected Attribute Spaces of Multifields over Any Spatial Sampling. *Computer Graphics Forum*, 32:301–310, 2013.

[98] Adriano Moreira and Maribel Yasmina Santos. Concave hull: A k-nearest Neighbours Approach for the Computation of the Region Occupied by a Set of Points, 2007.

[99] Mark E J Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[100] Nam P Nguyen, Thang N Dinh, Yilin Shen, and My T Thai. Dynamic Social Community Detection and its Applications. *PloS One*, 9(4):e91431, 2014.

[101] Philippe Noriega, Benedicte Bascle, and Olivier Bernier. Local Kernel Color Histograms for Background Subtraction. In *Proceedings of the International Conference on Computer Vision Theory and Applications*, pages 213–219, 2006.

[102] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*, volume 153. Springer Science & Business Media, 2006.

[103] Sedat Ozer, Jishang Wei, Deborah Silver, Kwan-Liu Ma, and Patrick Martin. Group dynamics in scientific visualization. In *Proceedings of the IEEE Symposium on Large Data Analysis and Visualization*, pages 97–104, Oct 2012.

[104] Gregorio Palmas, Myroslav Bachynskyi, Antti Oulasvirta, Hans Peter Seidel, and Tino Weinkauf. An Edge-bundling Layout for Interactive Parallel Coordinates. In *IEEE Pacific Visualization Symposium (PacificVis2014)*, pages 57–64, 2014.

[105] Daniele Panozzo, Ofir Weber, and Olga Sorkine. Robust Image Retargeting via Axis-Aligned Deformation. *Computer Graphics Forum*, 31(2-1):229–236, 2012.

[106] Ken Perlin. An Image Synthesizer. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '85)*, pages 287–296, New York, NY, USA, 1985. ACM.

[107] Jorge Poco, Angela Mayhua, and Jeffrey Heer. Extracting and Retargeting Color Mappings from Bitmap Images of Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 2017.

[108] Yael Pritch, Eitam Kav-Venaki, and Shmuel Peleg. Shift-Map Image Editing. In *IEEE International Conference on Computer Vision (CVPR2009)*, pages 151–158, Sept 2009.

[109] Yao Qin, Huchuan Lu, Yiqun Xu, and He Wang. Saliency Detection via Cellular Automata. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR2015)*, 2015.

[110] Xiaofeng Ren and Jitendra Malik. Learning a Classification Model for Segmentation. In *Proceedings of the Ninth IEEE International Conference on Computer Vision*, pages 10–17, 2003.

[111] George Robertson, Roland Fernandez, Danyel Fisher, Bongshin Lee, and John Stasko. Effectiveness of Animation in Trend Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1325–1332, 2008.

[112] Alex Rodriguez and Alessandro Laio. Clustering by Fast Search and Find of Density Peaks. *Science*, 344(6191):1492–1496, 2014.

[113] Azriel Rosenfeld and John L Pfaltz. Sequential Operations in Digital Picture Processing. *Journal of ACM*, 13(4):471–494, 1966.

[114] Ruth Rosenholtz, Yuanzhen Li, Jonathan Mansfield, and Zhenlan Jin. Feature Congestion: A Measure of Display Clutter. In *Proc. SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 761–770, 2005.

[115] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[116] Michael Rubinstein, Diego Gutierrez, Olga Sorkine, and Ariel Shamir. A Comparative Study of Image Retargeting. *ACM Transactions Graphics*, 29(6):1–10, 2010.

[117] Michael Rubinstein, Ariel Shamir, and Shai Avidan. Improved Seam Carving for Video Retargeting. *ACM Transactions Graphics*, 27(3):1–9, 2008.

[118] Ravi Samtaney, Deborah Silver, Norman Zabusky, and Jim Cao. Visualizing Features and Tracking Their Evolution. *Computer*, 27(7):20–27, 1994.

[119] Manojit Sarkar and Marc H Brown. Graphical fisheye views of graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 83–91. ACM, 1992.

[120] Christoph Schulz, Arlind Nocaj, Jochen Goertler, Oliver Deussen, Ulrik Brandes, and Daniel Weiskopf. Probabilistic Graph Layout for Uncertain Network Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):531–540, 2017.

[121] Bremm Sebastian, Gennady Andrienko, Natalia Andrienko, Tobias Schreck, and Tatiana von Landesberger. Interactive Analysis of Object Group Changes over Time. In *Proc. International Workshop on Visual Analytics (EuroVA 2011)*, pages 41–44, 2011.

[122] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Guality Mesh Generator and Delaunay Triangulator. pages 203–222, 1996.

[123] Lei Shi, Nan Cao, Shixia Liu, Weihong Qian, Li Tan, Guodong Wang, Jimeng Sun, and Ching-Yung Lin. Himap: Adaptive visualization of large-scale online social networks. In *2009 IEEE Pacific Visualization Symposium (PacificVis)*, pages 41–48, 2009.

[124] Abhinav Shrivastava, Tomasz Malisiewicz, Abhinav Gupta, and Alexei A Efros. Data-driven Visual Similarity for Cross-domain Image Matching. In *ACM Transactions on Graphics*, volume 30, page 154, 2011.

[125] Bernard W Silverman. *Density Estimation for Statistics and Data Analysis*, volume 26. CRC press, 1986.

[126] Maoyuan Sun, Peng Mi, Chris North, and Naren Ramakrishnan. Biset: Semantic edge bundling with biclusters for sensemaking. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):310–319, 2016.

[127] Yuzuru Tanahashi and Kwan-Liu Ma. Design Considerations for Optimizing Storyline Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2679–2688, 2012.

[128] Alexandru Telea and Ozan Ersoy. Image-based edge bundles: Simplified visualization of large graphs. 29(3):843–852, 2010.

[129] J P Thirion. Image Matching as a Diffusion Process: an Analogy with Maxwell's Demons. *Medical Image Analysis*, 2(3):243–260, 1998.

[130] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: The New Data in Multimedia Research. *Communications of the ACM*, 59(2):64–73, 2016.

[131] Yuanyuan Tian, Richard A Hankins, and Jignesh M Patel. Efficient Aggregation for Graph Summarization. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 567–580, 2008.

[132] Ying Tu and Han-Wei Shen. Balloon Focus: A Seamless Multi-Focus+Context Method for Treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1157–64, 2008.

[133] Cagatay Turkay, Július Parulek, Nathalie Reuter, and Helwig Hauser. Interactive Visual Analysis of Temporal Cluster Structures. In *Computer Graphics Forum*, volume 30, pages 711–720, 2011.

[134] Cagatay Turkay, Aidan Slingsby, Helwig Hauser, Jo Wood, and Jason Dykes. Attribute Signatures: Dynamic Visual Summaries for Analyzing Multivariate Geographical Data. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):2033–2042, 2014.

[135] Barbara Tversky, Julie Bauer Morrison, and Mireille Betrancourt. Animation: Can It Facilitate? *International Journal of Human-Computer Studies*, 57(4):247–262, 2002.

[136] S. van den Elzen, D. Holten, J. Blaas, and J.J. van Wijk. Reducing snapshots to points: A visual analytics approach to dynamic network exploration. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):1–10, Jan 2016.

[137] Matthew van der Zwan, Valeriu Codreanu, and Alexandru Telea. Cubu: universal real-time bundling for large graphs. *IEEE Transactions on Visualization and Computer Graphics*, 22(12):2550–2563, 2016.

[138] Jarke J Van Wijk. Image Based Flow Visualization. *ACM Transactions on Graphics*, 21(3):745–754, 2002.

[139] Corinna Vehlow, Fabian Beck, Patrick Auwärter, and Daniel Weiskopf. Visualizing the Evolution of Communities in Dynamic Graphs. In *Computer Graphics Forum*, volume 34, pages 277–288, 2015.

[140] Tatiana Von Landesberger, Sebastian Bremm, Natalia Andrienko, Gennady Andrienko, and Maria Tekusova. Visual analytics methods for categoric spatio-temporal data. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 183–192, 2012.

[141] Jingyuan Wang, Qian Gu, Junjie Wu, Guannan Liu, and Zhang Xiong. Traffic Speed Prediction and Congestion Source Exploration: A Deep Learning Method. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 499–508. IEEE, 2016.

[142] Yu-Shuen Wang and Ming-Te Chi. Focus+Context Metro Maps. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2528–2535, 2011.

[143] Yu-Shuen Wang, Hongbo Fu, Olga Sorkine, Tong-Yee Lee, and Hans-Peter Seidel. Motion-Aware Temporal Coherence for Video Resizing. *ACM Transactions Graphics*, 28(5):1–10, 2009.

[144] Yu-Shuen Wang, Hui-Chih Lin, Olga Sorkine, and Tong-Yee Lee. Motion-Based Video Retargeting with Optimized Crop-and-Warp. In *ACM Transactions on Graphics*, volume 29, page 90, 2010.

[145] Yu-Shuen Wang, Chiew-Lan Tai, Olga Sorkine, and Tong-Yee Lee. Optimized Scale-and-Stretch for Image Resizing. *ACM Transactions Graphics*, 27(5):1–8, 2008.

[146] Yunzhe Wang, George Baciu, and Chenhui Li. Smooth animation of structure evolution in time-varying graphs with pattern matching. In *SIGGRAPH Asia Symposium On Visualization*, 2017.

[147] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[148] Rainer Wegenkittl, Eduard Groller, and Werner Purgathofer. Animating Flow Fields: Rendering of Oriented Line Integral Convolution. In *Computer Animation'97*, pages 15–21, 1997.

[149] Stephen Welstead. *Fractal and Wavelet Image Compression Techniques*. SPIE Optical Engineering Press, 1999.

[150] Stephen Welstead and Jorge Nocedal. *Numerical Optimization*. Springer Science, 2006.

[151] Sebastian Wernicke. Efficient detection of network motifs. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(4):347–359, 2006.

[152] Hadley Wickham. Asa 2009 data expo. *Journal of Computational and Graphical Statistics*, 20(2), 2011.

[153] Hadley Wickham. Bin-summarise-smooth: a Framework for Visualising Large Data. *Tech. rep.,had.co.nz*, 2013.

[154] Niels Willems, Huub van de Wetering, and Jarke J. van Wijk. Visualization of Vessel Movements. In *Proceedings of the 11th Eurographics / IEEE - VGTC Conference on Visualization*, pages 959–966, 2009.

[155] Pak Chung Wong, Harlan Foote, Dan Adams, Wendy Cowley, and Jim Thomas. Dynamic Visualization of Transient Data Streams. In *IEEE Symposium on Information Visualization (InfoVis 2003)*, pages 97–104, 2003.

[156] Krist Wongsuphasawat and Ben Shneiderman. Finding comparable temporal categorical records: A similarity measure with an interactive visualization. In *IEEE Symposium on Visual Analytics Science and Technology (VAST2009)*, pages 27–34. IEEE, 2009.

[157] Jonathan Woodring and Han-Wei Shen. Multiscale Time Activity Data Exploration via Temporal Clustering Visualization Spreadsheet. *IEEE Transactions on Visualization and Computer Graphics*, 15(1):123–137, 2009.

[158] Hsiang-Yun Wu, Shigeo Takahashi, Daichi Hirono, Masatoshi Arikawa, Chun-Cheng Lin, and Hsu-Chun Yen. Spatially Efficient Design of Annotated Metro Maps. *Computer Graphics Forum*, 32(3pt3):261–270, 2013.

[159] Huisi Wu, Yu-Shuen Wang, Kun-Chuan Feng, Tien-Tsin Wong, Tong-Yee Lee, and Pheng-Ann Heng. Resizing by Symmetry-Summarization. *ACM Transactions Graphics*, 29(6):1–10, 2010.

[160] Yanhong Wu, Naveen Pitipornvivat, Jian Zhao, Sixiao Yang, Guowei Huang, and Huamin Qu. egoslider: Visual analysis of egocentric network evolution. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):260–269, 2016.

[161] Yanhong Wu, Wenbin Wu, Sixiao Yang, Youliang Yan, and Huamin Qu. Interactive visual summary of major communities in a large network. In *IEEE Pacific Visualization Symposium (PacificVis2015)*, pages 47–54, April 2015.

[162] Yingcai Wu, Xiaotong Liu, Shixia Liu, and Kwan-Liu Ma. ViSizer: A Visualization Resizing Framework. *IEEE Transactions on Visualization and Computer Graphics*, 19(2):278–290, 2012.

[163] Reynold S Xin, Joseph E Gonzalez, Michael J Franklin, Ion Stoica, and Eecs AMPLab. GraphX: A Resilient Distributed Graph System on Spark. *First International Workshop on Graph Data Management Experiences and Systems*, pages 2:1–2:6, 2013.

[164] Junming Xu. *Topological Structure and Analysis of Interconnection Networks*, volume 7. Springer Science & Business Media, 2013.

[165] Li Xu, Qiong Yan, Yang Xia, and Jiaya Jia. Structure Extraction from Texture via Relative Total Variation. *ACM Transactions on Graphics*, 31(6):1–10, 2012.

[166] Qiong Yan, Li Xu, Jianping Shi, and Jiaya Jia. Hierarchical Saliency Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR2013)*, pages 1155–1162, 2013.

[167] Han Yu, Xu Chen, Baciu George, Li Min, and Islam Md. Robiul. Cartoon and Texture Decomposition-Based Color Transfer for Fabric Images. *IEEE Transactions on Multimedia*, 19(1):80–92, 2017.

[168] Zongqiao Yu, Lin Lu, Yanwen Guo, Rongfei Fan, Mingming Liu, and Wenping Wang. Content-Aware Photo Collage using Circle Packing. *IEEE Transactions on Visualization and Computer Graphics*, 20(2):182–195, 2014.

[169] Xiaoru Yuan, Donghao Ren, Zuchao Wang, and Cong Guo. Dimension Projection Matrix/Tree: Interactive Subspace Visual Exploration and Analysis of High Dimensional Data. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2625–2633, 2013.

[170] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster Computing with Working Sets. In *Proceedings of*

the 2nd USENIX Conference on Hot Topics in Cloud Computing, pages 1–7, 2010.

[171] Guo-Xin Zhang, Ming-Ming Cheng, Shi-Min Hu, and Ralph R Martin. A Shape-Preserving Approach to Image Resizing. *Computer Graphics Forum*, 28(7):1897–1906, 2009.

[172] Xiao Zhang, Travis Martin, and M. E. J. Newman. Identification of Core-periphery Structure in Networks. *Physical Review E*, 91(3):1–10, 2015.

[173] Xiujun Zhang, Chen Xu, Min Li, and Robert K.F. Teng. Study of Visual Saliency Detection via Nonlocal Anisotropic Diffusion Equation. *Pattern Recognition*, 48(4):1315 – 1327, 2015.

[174] Yu Zheng, Furui Liu, and Hsun-Ping Hsieh. U-Air: When Urban Air Quality Inference Meets Big Data. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1436–1444. ACM, 2013.

[175] Yu Zheng, Tong Liu, Yilun Wang, Yanmin Zhu, Yanchi Liu, and Eric Chang. Diagnosing New York City's Noises with Ubiquitous Data. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '14, pages 715–725, New York, NY, USA, 2014.

[176] Hong Zhou, Panpan Xu, Xiaoru Yuan, and Huamin Qu. Edge bundling in information visualization. *Tsinghua Science and Technology*, 18(2):145–156, 2013.

[177] Daniel Zielasko, Benjamin Weyers, Bernd Hentschel, and Torsten W Kuhlen. Interactive 3D Force-Directed Edge Bundling. 35(3):51–60, 2016.

[178] Michael Zinsmaier, Ulrik Brandes, Oliver Deussen, and Hendrik Strobelt. Interactive Level-of-Detail Rendering of Large Graphs. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2486–2495, 2012.