StreamMap: Smooth Dynamic Visualization of High-Density Streaming Points

Chenhui Li, George Baciu, *Member, IEEE,* and Yu Han

Abstract—Interactive visualization of streaming points for real-time scatterplots and linear blending of correlation patterns is increasingly becoming the dominant mode of visual analytics for both big data and streaming data from active sensors and broadcasting media. To better visualize and interact with inter-stream patterns, it is generally necessary to smooth out gaps or distortions in the streaming data. Previous approaches either animate the points directly or present a sampled static heatmap. We propose a new approach, called *StreamMap*, to smoothly blend high-density streaming points and create a visual flow that emphasizes the density pattern distributions. In essence, we present three new contributions for the visualization of high-density streaming points. The first contribution is a density-based method called *super kernel density estimation* that aggregates streaming points using an adaptive kernel to solve the overlapping problem. The second contribution is a robust *density morphing algorithm* that generates several smooth intermediate frames for a given pair of frames. The third contribution is a trend representation design that can help convey the flow directions of the streaming points. The experimental results on three datasets demonstrate the effectiveness of StreamMap when dynamic visualization and visual analysis of trend patterns on streaming points are required.

Index Terms—Information visualization, trend visualization, streaming data, density map, time-varying, scatterplots

1 Introduction

Research on streaming data visualization is becoming particularly important 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) 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. 2(a). The advantages of smooth dynamic point 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 scatterplots. This advantage was reported by Tversky et al. [1]. Second, as the size of data exponentially increases, it becomes increasingly more difficult to show sufficient information in a single image frame. Although some techniques such as binning and summarization [2] perform re-sampling and data reduction for the target display, the number of pixels for a static image will always remain finite, whereas the points from data streams can easily exceed the display capacity. Third, morphing operations, as performed in our method, produce intermediate patterns compared to static visualizations. Moreover, these additional patterns can include data trends, as presented in the work of Thirion [3].

Histogram and *kernel density estimation* (KDE) have been used to overcome the overlapping problem [4]. However, the histogram has some disadvantages, as mentioned in the works by Noriega et al. [4] and Lampe and Hauser [5]; it is less smooth and constrains bin selection. KDE requires manual bandwidth adjustment to estimate the density of streaming points. 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. 2(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 be-

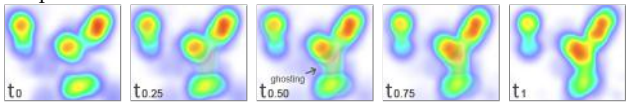
- *C. Li and G. Baciu are with the GAMA Lab, Dept. Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong. E-mail: dawn.chli@gmail.com, csgeorge@comp.polyu.edu.hk. G. Baciu is the corresponding author.*
- *Y. Han is with the College of Mathematics and Computational Science, Shenzhen University, China. E-mail: hany@szu.edu.cn.*

tween frames is necessary to produce patterns that are easier to observe and evaluate in data streaming visualization.

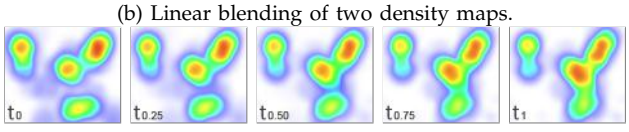
Fig. 1. High-density streaming points at different time steps.

To address the above issues, we propose a novel framework to represent streaming points, called *StreamMap* (Fig. 3). StreamMap offers a smooth dynamic visualization in large regions of point clouds, as shown in Fig. 2(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.

(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.

Fig. 2. 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.

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. [6] and Andrienko et al. [7]. 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. 5.1) include explicit continuous point distributions; therefore, we adopt artificial data to evaluate the effectiveness of the morphing method. Because a crowd of people (Sec. 5.2) includes heterogeneous point densities, it was used to demonstrate the density estimation method. Air pollution (Sec. 5.3) is more unusual because the point positions are fixed at different time steps.

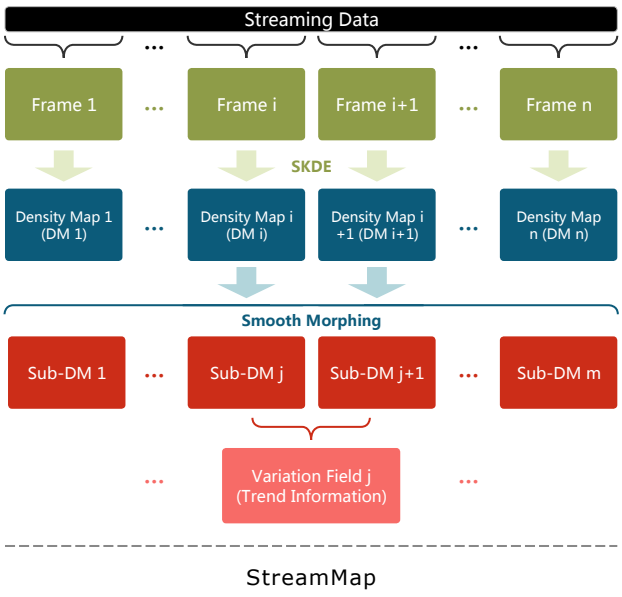


Fig. 3. The definition of StreamMap. We assume that points in a stream are similar to moving darts and that the frame is the collection of darts on a projected 2D plane representation, as shown at the top of the figure. Streaming data are organized as a density map through data aggregation and density estimation. StreamMap's diffusion model supports the composition of sub-DMs between two density maps.

Our visualization experiments on the three cases show that StreamMap can effectively overcome the overlapping problem, avoid abrupt changes in the data streams, and help to reveal the key patterns needed to comprehend the information contained in point streams. In addition, our approach can also be used to prioritize the trends at various resolutions in streaming data. We summarize our major contributions as follows:

(1) We propose a new framework for visualizing and exploring point data streams;

(2) We present an adaptive kernel density estimation method to aggregate high-density points in a period as a density map with high accuracy;
(3) We propose a novel algorithm for smoothing by morphing the estimated density maps;
(4) We provide a visualization approach to show the variation trends in streaming data.

2 Related Work

Our work is related to three types of techniques, namely, overcoming the overdraw problem, dynamic streaming point visualization, and feature tracking and representation.

2.1 Overcoming the Overdraw Problem

Many methods have been proposed to overcome the overdraw problem in high-density point visualizations. These methods can be categorized into three main types: sampling, projection, and statistics.

The sampling methods merge points according to their position distribution, a time step, or a feature channel. The binning and summarization [2] method samples data to reduce the size of the target display. Chen et al. [8] proposed a visual abstraction and exploration system that samples the original points through blue noise calculation, which can represent multi-class point distributions. Cottam et al. [9] proposed a simple aggregation process that enables the concise expression of alpha composition. When data points are obtained from moving objects such as taxis, flights or animals, the sampling approach presented by Andrienko et al. [10, 11] can be used to explore the locations of significant changes.

Compared with aggregation, the projection method converts data from the original space into a different space. Keim et al. [12] 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 [13] considered readers' impressions by adjusting the aspect ratio. In addition, for high-dimensional data visualizations, a projection technique proposed by Molchanov et al. [14] represented the point data in 5D attribute space. The projection matrix and tree methods were proposed in the work of Yuan et al. [15] to provide insights into high-dimensional data.

In addition to the aggregation and projection methods, statistical methods such as kernel density estimation (KDE) [16] have frequently been adopted to overcome the overlapping problem in high-density point visualizations. Lampe and his colleagues [5] visualized large-scale traffic data on a map using KDE. Similarly, the KDE technique was adopted by Willems et al. [6] to address the problems involved in visualizing moving objects. Because point data may belong to different groups, Mayorga and Gleicher [17] presented a splatter plot to visualize group contours using an extended KDE method.

2.2 Dynamic Streaming Point Visualization

A considerable amount of work has been performed to achieve dynamic effects from streaming point data. Krstajic and Keim [18] addressed the challenges involved in measuring changes and maintaining context in dynamic information visualization. Assuming that each frame is a scatterplot representation and that each point's position in each frame is known, then position interpolation is a suitable method to overcome ghosting. Position interpolation, as explained in the works of Robertson et al. [19] and Du et al. [20], involves dynamically computing the movement trajectory for each point. 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 numbers of animated points increase.

Generating density maps frame by frame is a good solution candidate to overcome visual confusion because KDE normalizes all the points in a frame into a structured density map. However, visual ghosting is still a problem, e.g., Fig. 2(b), if we adopt linear interpolation between two density maps when the frame displacements are large.

A nonlinear retargeting algorithm such as optical flow [21] could be used to establish accurate correspondences between two density maps to avoid visual ghosting. However, the time-consuming iteration required to morph density maps and the misconvergence problem make optical flow impractical for meeting the requirements of dynamic visualizations. Mahajan et al. [22] described a path-based moving gradient approach that can handle complex non-rigid morphing. Unfortunately, the moving gradient [22] method requires further improvement when the content exhibits large-scale changes between frames.

The choice of bandwidth selection for KDE is extremely important in dynamic visualization because it is related to the accuracy of density estimation. For streaming data visualizations, adaptive bandwidth estimation was adopted by Lampe et al. [5] to estimate the density of points with respect to the level of detail in the data. However, their method did not include adapting the bandwidth of KDE for each region at one level, which may lead to inaccurate density maps. Therefore, for streaming points, the existing animation-based morphing techniques require more improvement to achieve better dynamic effects.

2.3 Feature Tracking and Representation

Our work is also related to scientific visualization studies, such as those in the field of feature tracking. Extracting features such as trend information

from streaming data can be of great benefit to users. Woodring et al. [23] used the wavelet transform to change point sets into curve sets along a time axis to track time-varying trends. Flow-based scatterplots [24] were presented to highlight variations in flow data. Moreover, Samtaney et al. [25] 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. [25], Ozer and his colleagues [26] tracked clusters of features from time-varying 3D flow fields to improve the performance.

In addition, Grottel et al. [27] addressed flow groups to study molecular dynamics by visualizing cluster evolution over time. Moreover, cluster structural variations were visualized by Turkay et al. [28] through an interactive cluster viewing design. However, little work has been performed on tracking dynamic features in streaming point data.

3 Definitions

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.

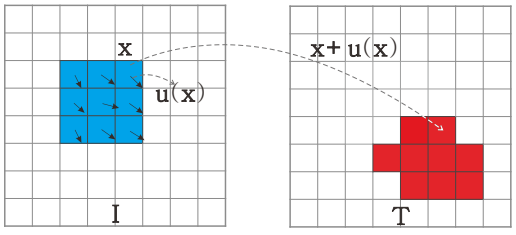


Fig. 4. An example of the density diffusion between two density maps. $\mathbf{x}$ means a pixel in a density map, and $\mathbf{u}$ indicates the transformation value. I and T are a pair of inputs in the morphing model.

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. 4. The input 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 $\mathbf{u}(\mathbf{x})$,

$$I_{\mathbf{u}}(\mathbf{x}) = I(\mathbf{x} + \mathbf{u}(\mathbf{x})), \mathbf{x} \in \mathbf{\Omega}, \quad (1)$$

where $\mathbf{x}$ denotes 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.

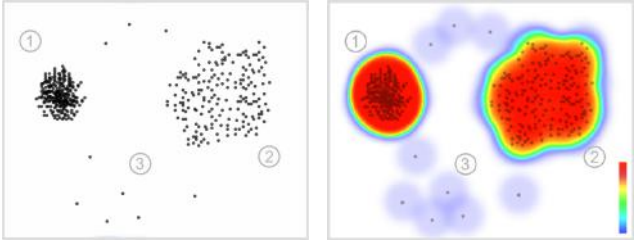
4 StreamMap

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 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.

4.1 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.



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

Fig. 5. KDE result with fixed bandwidth. The number texts on the figure indicate three different density points.

4.1.1 Adaptive Kernel Density Estimation

We improve the prior KDE approach [5] 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 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:

$$K(\mathrm{x}) = \frac{1}{\mathrm{n}} \sum_{\mathrm{j}=1}^{\mathrm{n}} \frac{1}{\mathrm{h_j}} \mathrm{G}(\frac{|\mathrm{x}-\mathrm{x_j}|}{\mathrm{h_j}}), \mathrm{x_j} \in \mathrm{F}, \mathrm{x} \in \mathrm{F}, \quad (2)$$

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. 5(b) as there are some independent points in spatial space. In addition, the KDE result in Fig. 5(b) shows a poor visual result because medium-density points at set-2 were estimated with high density that is similar with 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.

4.1.2 Superpoint Generation

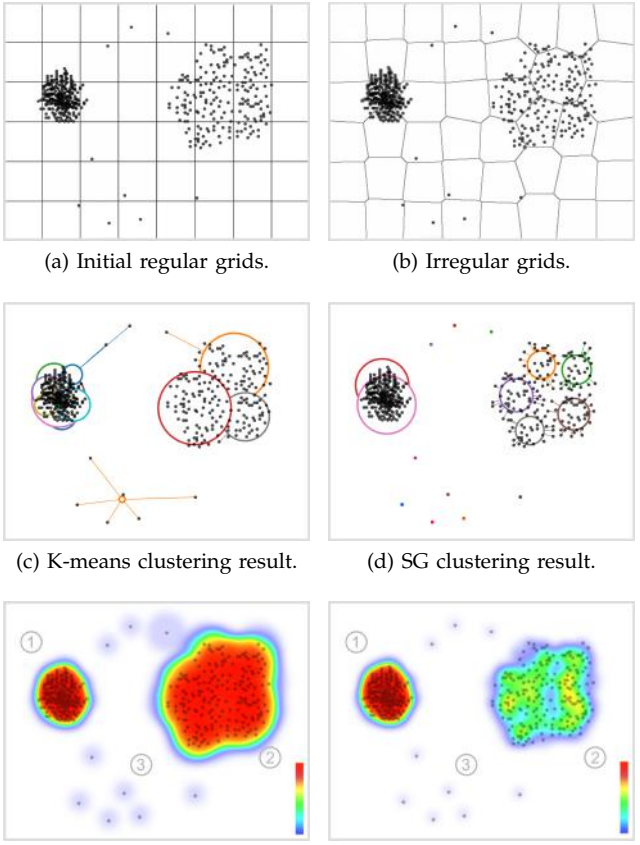
The adaptive bandwidth setting is achieved through *superpoint generation* (SG), which clusters the points into nearly uniform-area superpoints. SG is inspired from the superpixel algorithms [29] and [30] used in the image processing field. Superpixel, which was first presented by Ren and Malik [29], 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 [30] is an improvement of the work of Ren and Malik [29], 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.

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 $k = \frac{\sqrt{\mathrm{dwidth}\cdot\mathrm{dheight}}}{8}$. 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:

$$sh_i = \frac{n_i}{\sum_{j=1}^{n_i} \|\mathrm{p_{ij}} - \mathrm{c_i}\|}, i \in [1, k], \quad (3)$$

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. 2) can be achieved after calculating sh_i.



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

(c) K-means clustering result. (d) SG clustering result.

(e) Adaptive KDE result using k-mean method. (f) Adaptive KDE result using SG method (SKDE).

Fig. 6. A comparison of the k-means and the SG clustering methods.

Finally, a grayscale density map $\boldsymbol{D}$ for each frame can be generated using Eq. 2. SG is an improvement of k-means [31] that exhibits improved performance because the required distance calculations are reduced by limiting the search region. Figure 6(a) presents

the initial regular grids of SG, and Fig. 6(b) shows an example of the irregular grids in the SG process. Figure 6(c, d) shows a comparison of the k-means and SG clustering methods. Each circle in Fig. 6(c, d) 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. 6(f), the SG method achieves a better density map than the k-means method. Set-2 in Fig. 6(f) are estimated with a medium density that is more accurate than the one in Fig. 6(e).

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

4.2.1 Diffusion Model

Inspired by the Demons diffusion model [3], we formulate the morphing operation between a pair of density maps as the following optimization problem:

$$\begin{cases} {\boldsymbol{\delta}_{\mathbf{u}}}^{(n+1)} = \arg\min_{\boldsymbol{\delta}_{\mathbf{u}}} \{ \underbrace{E_d(I, T, \boldsymbol{\delta}_{\mathbf{u}})}_{data} + \underbrace{\lambda E_r(\boldsymbol{\delta}_{\mathbf{u}})}_{\text{regularization}} \} \\ \mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + {\boldsymbol{\delta}_{\mathbf{u}}}^{(n+1)} \end{cases}, \tag{4}$$

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:

$$E_d = \int_{\Omega} \left\| I_{\mathbf{u}^{(n)}} + (\nabla I_{\mathbf{u}^{(n)}})^{\mathrm{T}} \boldsymbol{\delta}_{\mathbf{u}} - T \right\|^2 \mathbf{dx}, \tag{5}$$

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:

$$E_r = \int_{\Omega} \|\boldsymbol{\delta}_{\mathbf{u}}\|^2 \mathbf{dx}. \tag{6}$$

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 ${\boldsymbol{\delta}_{\mathbf{u}x}}^{(n+1)}$ 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. 7.

$$\begin{aligned} &E'(\boldsymbol{\delta}_{\mathbf{u}}) = \mathbf{0} \Rightarrow \\ &(I_{\mathbf{u}^{(n)}} - T)\nabla I_{\mathbf{u}^{(n)}} + (\nabla I_{\mathbf{u}^n} \cdot \delta_{\mathbf{u}})\nabla I_{\mathbf{u}^n} + \lambda \boldsymbol{\delta}_{\mathbf{u}} = \mathbf{0} \end{aligned} \tag{7}$$

$${\boldsymbol{\delta}_{\mathbf{u}x}}^{(n+1)} = \frac{T - I_{\mathbf{u}^{(n)}}}{(\partial_x I_{\mathbf{u}^{(n)}})^2 + \lambda} \partial_x I_{\mathbf{u}^{(n)}} \tag{8}$$

$${\boldsymbol{\delta}_{\mathbf{u}y}}^{(n+1)} = \frac{T - I_{\mathbf{u}^{(n)}}}{(\partial_y I_{\mathbf{u}^{(n)}})^2 + \lambda} \partial_y I_{\mathbf{u}^{(n)}} \tag{9}$$

From Eq. 7, we arrive at the solution as shown in Eq. 8 and Eq. 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. 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:

$$\begin{cases} g_x(i,j) = \frac{1}{2}(D(i-1,j) - D(i+1,j)), \\ g_y(i,j) = \frac{1}{2}(D(i,j-1) - D(i,j+1)). \end{cases}, \tag{10}$$

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

4.2.2 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. 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. 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. 7(c), means that I and T overlap. When $I \cap T = \emptyset$, as shown in Fig. 7(d-f), this diffusion model is inefficient. Finally, for the pattern in Fig. 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. 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.

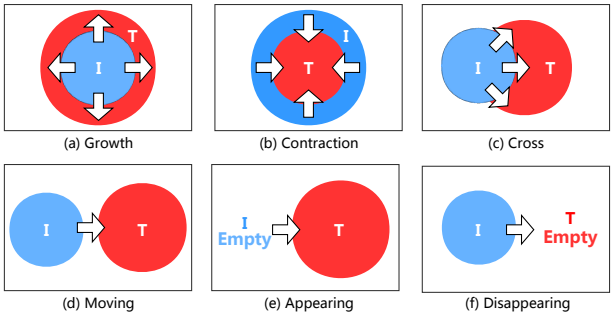
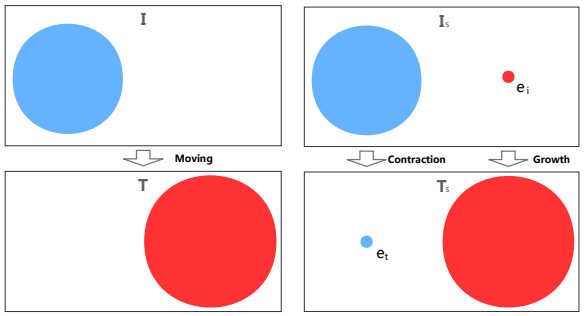


Fig. 7. Six morphing patterns. The blue region indicates the area in the original density map. The red region shows the area in the target density map. A complete morphing procedure for two density maps normally includes various morphing patterns.

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. 7 can be divided into two patterns that match the top patterns in Fig. 7. For example, if we add two helper seeds called e_i and e_t to the moving pattern region (Fig. 8a), we obtain two sub-patterns as shown in Fig. 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.



(a) Original I and T. (b) New I_s and T_s with helper seeds.

Fig. 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.

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, particularly in the peak-value pixels. We adopt the *steepest descent* (SD) method [32] (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:

$$\begin{cases} I_s = I + E(SD(Sr(T, \beta))) \\ T_s = T + E(SD(Sr(I, \beta))) \end{cases} \tag{11}$$

We define the density maps with helper seeds as I_s and T_s. Fig. 9 shows a result of peak pixel detection on a density map using the steepest descent method [32]. Fig. 10 shows six morphing results using our seed-based morphing model (I_s and T_s are inputs).

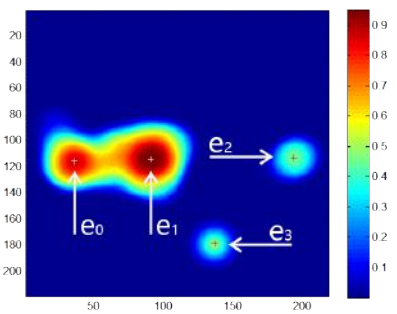
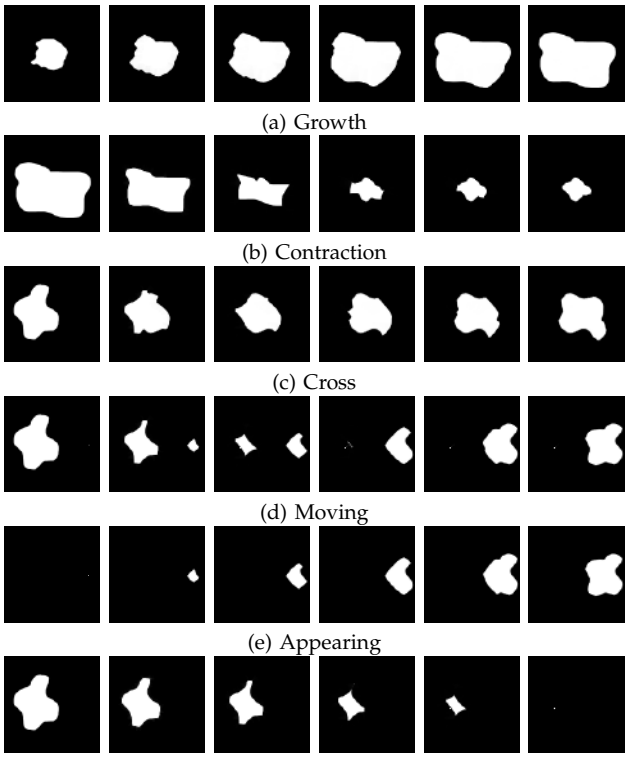


Fig. 9. A result of peak pixel detection on a density map using the steepest descent method [32]. e_0, e_1, e_2, and e_3 are the detected peak pixels.

4.2.3 *Overcoming Density Nonconformity*

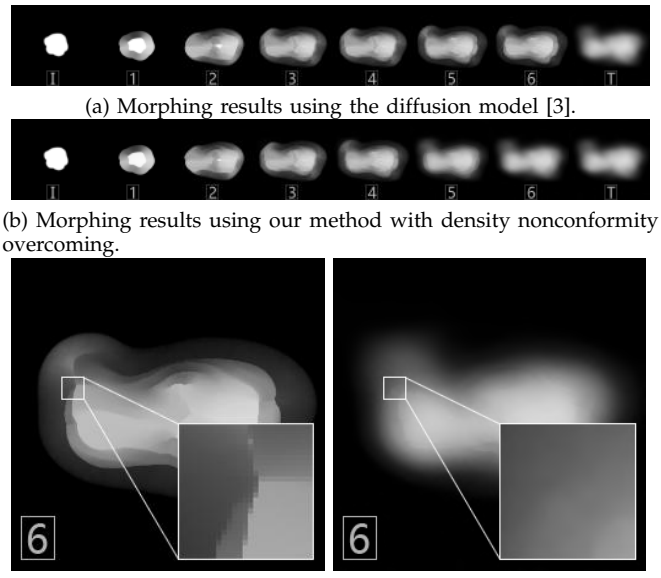
By adding the helper seeds, the diffusion model may still suffer from the *density nonconformity* (DN) problem, as shown in Fig. 11(a). DN means that the final morphing sequence will not match the target density map (T) using only limited calculation iterations. As shown in Fig. 11(a), the DN problem makes it difficult to obtain satisfactory morphing results. Figure 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 11(a) shows a pair of density maps that will not fulfill the AC. The morphing result generated using the diffusion model is consequently



(a) Growth
(b) Contraction
(c) Cross
(d) Moving
(e) Appearing
(f) Disappearing

Fig. 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.



(a) Morphing results using the diffusion model [3].
(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.

Fig. 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.

distorted, as shown in Fig. 11(a). Hence, an improved method is required to overcome the DN problem.

In linear blending, although distortion and ghosting appear frequently in the morphing 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:

$$\begin{cases} S_{i+1} = (1-(\frac{i}{\tau})^{\psi})S_{i\mathbf{u}_i} + (\frac{i}{\tau})^{\psi}T, S_0 = I_s \\ \mathbf{u}_{i+1} = \mathbf{u}_i + \frac{T_s - S_{i\mathbf{u}_i}}{(\nabla S_{i\mathbf{u}_i})^2+\lambda}\nabla S_{i\mathbf{u}_i}, \end{cases} \quad (12)$$

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. 11(b). We find that the sixth sequence of the morphing result, as shown in Fig. 11(c), right, is better than using the diffusion model without overcoming DN as shown in Fig. 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.

4.3 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 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. 12(a), is similar to wind and current visualization. We define the basic visual element of TR as a trend

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

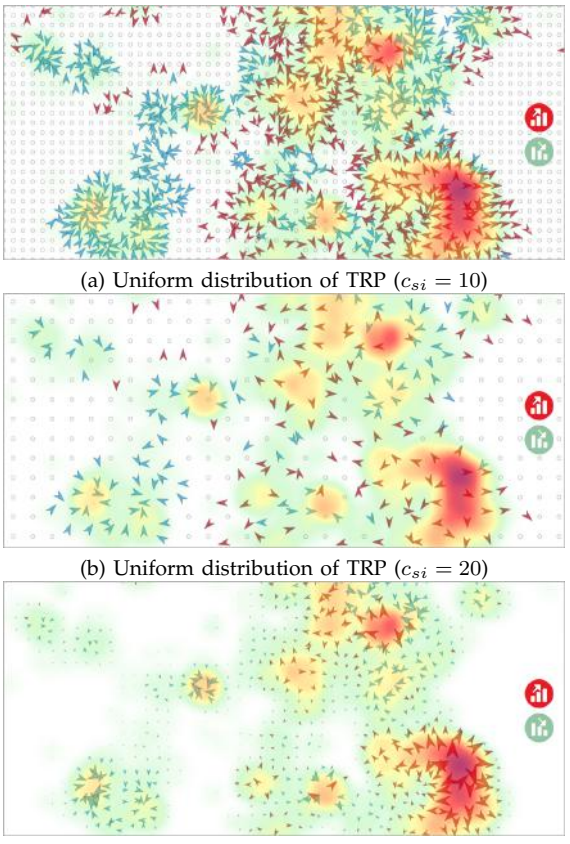
representation particle (TRP). Each TRP is represented by an arrow with a special size and direction on a density map. We define c_{si} as a sampling interval pixel number that defines the distance between neighboring TRPs. Fig. 12(a) and Fig. 12(b) show two TR results with different c_{si} values. In addition, as shown in Fig. 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 a 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 12(a) shows an example of a TRP distribution with $c_{si} = 10$.
(3) We define $\lambda(\|\boldsymbol{u}_{id}\|)$ to determine whether a TRP, generated in the second step, will be shown on the density map.

$$\lambda(\|\boldsymbol{u}_{id}\|) = \begin{cases} 0, \|\boldsymbol{u}_{id}\| < \theta \\ 1, \|\boldsymbol{u}_{id}\| \geq \theta \end{cases} \quad (13)$$

In Eq. 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.



(a) Uniform distribution of TRP ($c_{si} = 10$)
(b) Uniform distribution of TRP ($c_{si} = 20$)
(c) Non-linear distribution of TRP

Fig. 12. Trend direction representation.

Following the rules presented above, we visualize the NTR as shown in Fig. 12(c), which enhances the salient region and improves the performance.

5 Use Cases

This section demonstrates usages for StreamMap. We show how to apply our approach to three point stream datasets. The first dataset is an artificial dataset, whereas the other two are real-world datasets. Table 1 presents a summary of the experimental datasets. All visualizations are implemented using JavaScript and the D3 [33] and Leaflet [34] libraries. We 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.

5.1 Artificial Data (AD)

We use Perlin noise (PN) [35] to artificially generate a variety of time-varying point sets to test and evaluate

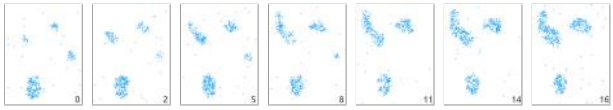
TABLE 1
Summary of datasets used in our experiments.

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

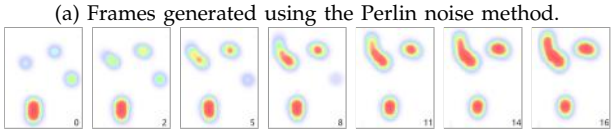
the StreamMap method. PN is frequently used to create natural object surfaces in the field of computer graphics; however, we use it here to make the two-dimensional testing points more realistic than random points.

Figure 13 shows an example of the morphing result (some of the morphing sequences are selected). Figure 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. 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. 13(c). The results are smooth and contain less artificial ghosting compared with the result using the linear blending method, as shown in Fig. 13(d). In addition, our morphing result is similar to the benchmarks, as shown in Fig. 13(b). We highlight a smooth morphing density map with corresponding frame in Fig. 13(e), which demonstrates that the morphing results closely match the point sets. In Fig. 13(f), we highlight the artifacts generated using the linear blending method.

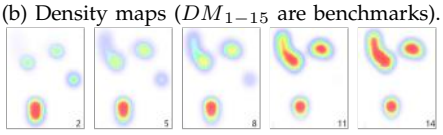
The similarities between benchmarks and sub-DMs will be calculated to evaluate the morphing effectiveness as presented in Sec. 6. When sub-DMs are similar to benchmarks, we consider that the morphing model has achieved a good result.



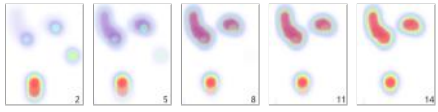
(a) Frames generated using the Perlin noise method.



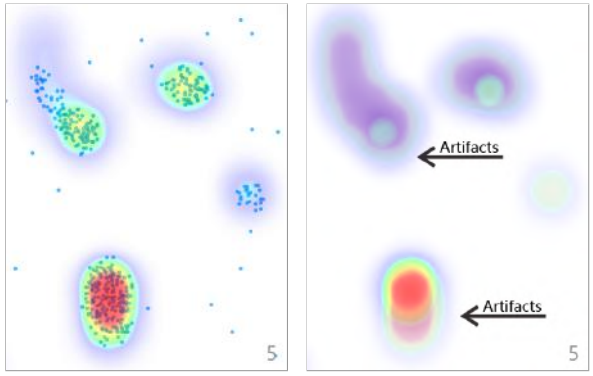
(b) Density maps (DM_{1-15} are benchmarks).



(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).



(e) Fifth in-between DM merged with $frame_5$ (ourblending).

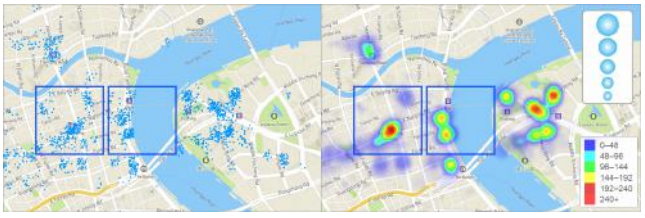
(f) Fifth in-between DM (linear method).

Fig. 13. Estimated density maps and the morphing result according to the artificial data (AD).

5.2 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 14(b) shows a density map generated using the SKDE method with adaptive kernel sizes. Figure 14(c-d) shows the results of the KDE method with coincident kernel sizes. Figure 14(c) (KDE with a small kernel size) shows unclear salient regions, whereas the result will appear over-estimated, as shown in Fig. 14(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. 14(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 15 shows smooth morphing results on 25 July 2015 at East Nanjing Subway Station (the left rectangle region in Fig. 14a). In Fig. 15, we observe that people are moving toward



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

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

Fig. 14. Estimated density maps according to the collected PC data between 1:00 PM and 2:00 PM on 25 July 2015.

Fig. 15. 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.

the southwest exit 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.

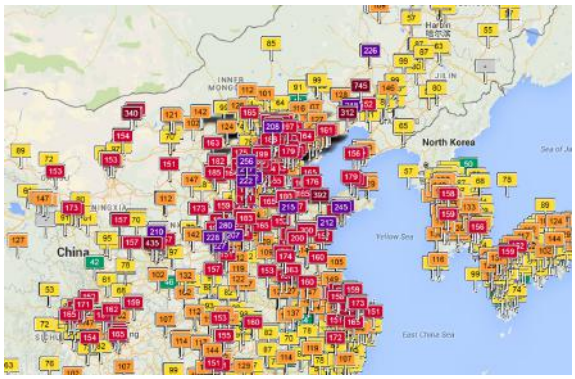
Figure 15 also shows the crowd variations of people at the Bund (the right rectangle region in Fig. 14a), which is a famous attraction in Shanghai. There are two crowds of people at the Bund; one is stable and the other is decreasing from 13:00 to 14:00. From 21:00 to 22:00, the upper people crowd at 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 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.

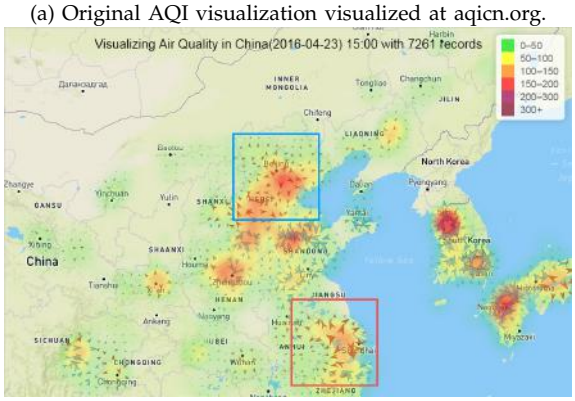
5.3 Air Pollution (AP)

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. 16(a). In addition, dynamic representation of AQI data is a difficult challenge.

Using the StreamMap method, we can generate smooth air pollution diffusion animations to aid viewers in understanding the distribution and variation



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

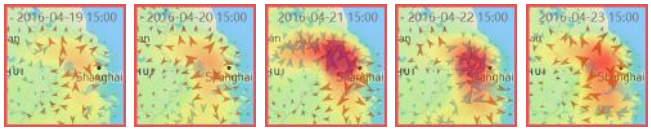


(b) Improved AQI visualization with density map and trend representation.

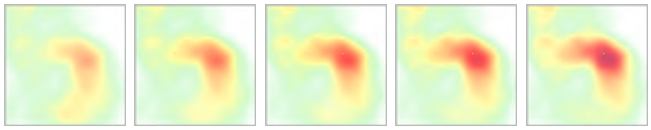
Fig. 16. Comparison of two air quality visualization methods.

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 16(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.



(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.

Fig. 17. Air pollution visualization using StreamMap.

As shown in Fig. 16(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 16(b) shows the air pollution density map estimated from Fig. 16(a) through SKDE. Figure 16(b) also shows the air pollution diffusion trend. From the blue rectangle in Fig. 16(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 pollution trend in Shanghai. The pollution from Shanghai may affect Anhui Province to the west of Shanghai. Fig. 17(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. 17(b) shows a variety of smooth air pollution sub-DMs. Both smooth sub-DMs and trend representations can help users notice an obvious distribution and a trend of the air pollution data.

6 Evaluation

Fig. 18 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 random 2D points over the fixed area. The generation of these random 2D points was discussed in Section 5.1. 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. 18, 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.

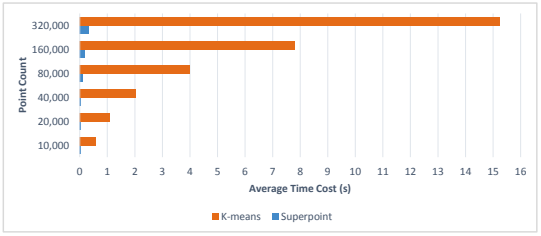


Fig. 18. Time cost of the SKDE method with different data sizes (the dataset used is AD as shown in Sec. 5.1).

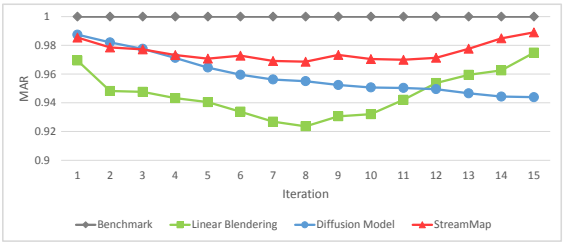
We also computed the structural similarity (SSIM) to evaluate the morphing effectiveness. SSIM was used as a structure similarity measurement between two images, as discussed in the work of Wang et al. [36]. SSIM is more consistent with the visual perception of a human than peak signal-to-noise ratio (PSNR) [37]. More similar density maps achieve higher SSIM scores. We selected 170 continuous frames from the AD dataset in Sec. 5.1 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 19(a) shows the MAR results obtained using three different methods: linear blending, diffusion model [3], and our method. As these results show, our smooth morphing results closely match the benchmarks.

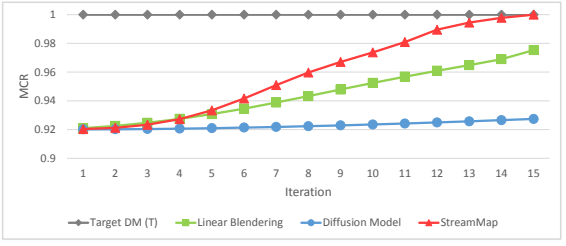
Another measurement is called morphing completion rate (MCR), which defines the SSIM of an in-between density map and a target density map (T). MCR is used to evaluate whether a morphing operation will be performed in finite iterations. If an in-between density map is similar to a target density map (T), then the MCR will be close to 1. Figure 19(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 [3] suffers mis-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.

Fig. 19. Comparisons of the morphing effectiveness using two measurements.

7 Conclusions

This paper 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 propose 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 visualizations.

StreamMap still has some limitations. First, although we adopted an adaptive bandwidth selection KDE method (SKDE) to estimate the density of high-density points, the accuracies of the density contour and peak still require improvement. Second, the accuracy of the morphing operation requires further improvement, particularly for handling the "moving" morphing pattern. Third, current trend representations also suffer from over-plotting, which might affect the density pattern finding. Potential improvements with more sophisticated representation might be achieved through flow-visualization methods, such as OLIC [38] and IBFV [39]. Fourth, for the data sets without a "flow" nature, such as the web portal logins and the urban noise, the smooth morphing work well; however, the trend representation will likely not present valid states.

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. [40] and Turkay et al. [41]. Many other applications can easily be configured to work with StreamMap, such as crowding effects in congested public urban transportation systems. Andrienko et al. [7] 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. [7] to address streaming trajectory data in the future.

Acknowledgments

We would like to acknowledge the partial support from IGRF PolyU 152142/15E, Project 4-ZZFF from the Department of Computing, Hong Kong Polytechnic University, NSFC under Grant 61402290 and Grant 61472257, and Grant 2014KQNCX134. In addition, we thank three anonymous reviewers for their constructive comments that helped us substantially improve the manuscript.

References

[1] B. Tversky, J. B. Morrison, and M. Betrancourt, "Animation: Can It Facilitate?" *International Journal of Human-Computer Studies*, vol. 57, no. 4, pp. 247–262, 2002.

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

[3] J. P. Thirion, "Image Matching as a Diffusion Process: an Analogy with Maxwell's Demons," *Med. Image Anal.*, vol. 2, no. 3, pp. 243–260, 1998.

[4] P. Noriega, B. Bascle, and O. Bernier, "Local Kernel Color Histograms for Background Subtraction," in *Proc. International Conference on Computer Vision Theory and Applications*, 2006, pp. 213–219.

[5] O. D. Lampe and H. Hauser, "Interactive Visualization of Streaming Data with Kernel Density Estimation," in *Proc. IEEE Pacific Visualization Symposium (PacificVis)*, 2011, pp. 171–178.

[6] N. Willems, H. van de Wetering, and J. J. van Wijk, "Visualization of Vessel Movements," in *Comput. Graph. Forum*, vol. 28, no. 3, 2009, pp. 959–966.

[7] G. Andrienko, N. Andrienko, S. Rinzivillo, M. Nanni, D. Pedreschi, and F. Giannotti, "Interactive Visual Clustering of Large Collections of Trajectories," in *Proc. IEEE Symposium on Visual Analytics Science and Technology (VAST 2009)*. IEEE, 2009, pp. 3–10.

[8] H. Chen, W. Chen, H. Mei, Z. Liu, K. Zhou, W. Chen, W. Gu, and K.-L. Ma, "Visual Abstraction and Exploration of Multi-class Scatterplots," *IEEE Trans. Vis. Comput. Graphics*, vol. 20, no. 12, pp. 1683–1692, 2014.

[9] J. Cottam, A. Lumsdaine, and P. Wang, "Overplotting: Unified Solutions under Abstract Rendering," in *Proc. IEEE International Conference on Big Data*, 2013, pp. 9–16.

[10] N. Andrienko and G. Andrienko, "Visual Analytics of Movement: An Overview of Methods, Tools and Procedures," *Information Visualization*, vol. 12, no. 1, pp. 3–24, 2013.

[11] G. Andrienko, N. Andrienko, C. Hurter, S. Rinzivillo, and S. Wrobel, "Scalable Analysis of Movement Data for Extracting and Exploring Significant Places." *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 7, pp. 1078–1094, 2013.

[12] D. A. Keim, C. Panse, M. Sips, and S. C. North, "Visual Data Mining in Large Geospatial Point Sets," *IEEE Comput. Graph. Appl.*, vol. 24, no. 5, pp. 36–44, 2004.

[13] M. Fink, J.-H. Haunert, J. Spoerhase, and A. Wolff, "Selecting the Aspect Ratio of a Scatter Plot Based on Its Delaunay Triangulation," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 12, pp. 2326–2335, 2013.

[14] V. Molchanov, A. Fofonov, and L. Linsen, "Continuous Representation of Projected Attribute Spaces of Multifields over Any Spatial Sampling," *Comput. Graph. Forum*, vol. 32, pp. 301–310, 2013.

[15] X. Yuan, D. Ren, Z. Wang, and C. Guo, "Dimension Projection Matrix/Tree: Interactive Subspace Visual Exploration and Analysis of High Dimensional Data," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 12, pp. 2625–2633, 2013.

[16] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. CRC press, 1986, vol. 26.

[17] A. Mayorga and M. Gleicher, "Splatterplots: Overcoming Overdraw in Scatter plots," *IEEE Trans. Vis. Comput. Graphics*, vol. 19, no. 9, pp. 1526–38, 2013.

[18] M. Krstajic and D. a. Keim, "Visualization of Streaming Data: Observing Change and Context in Information Visualization Techniques," *Proc. IEEE International Conference on Big Data*, pp. 41–47, 2013.

[19] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko, "Effectiveness of Animation in Trend Visualization," *IEEE Trans. Vis. Comput. Graphics*, vol. 14, no. 6, pp. 1325–1332, 2008.

[20] F. Du, N. Cao, J. Zhao, and Y.-R. Lin, "Trajectory Bundling for Animated Transitions," in *Proc. 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015, pp. 289–298.

[21] B. K. Horn and B. G. Schunck, "Determining Optical Flow," in *1981 Technical Symposium East*. International Society for Optics and Photonics, 1981, pp. 319–331.

[22] D. Mahajan, F.-C. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur, "Moving Gradients: a Path-based Method for Plausible Image Interpolation," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–11, 2009.

[23] J. Woodring and H.-W. Shen, "Multiscale Time Activity Data Exploration via Temporal Clustering Visualization Spreadsheet," *IEEE Trans. Vis. Comput. Graphics*, vol. 15, no. 1, pp. 123–137, 2009.

[24] Y.-H. Chan, C. D. Correa, and K.-L. Ma, "Flow-based Scatterplots for Sensitivity Analysis," in *Proc. IEEE Symposium on Visual Analytics Science and Technology*, 2010, pp. 43–50.

[25] R. Samtaney, D. Silver, N. Zabusky, and J. Cao, "Visualizing Features and Tracking Their Evolution," *Computer*, vol. 27, no. 7, pp. 20–27, 1994.

[26] S. Ozer, J. Wei, D. Silver, K.-L. Ma, and P. Martin, "Group dynamics in scientific visualization," in *Proc. IEEE Symposium on Large Data Analysis and Visualization*, Oct 2012, pp. 97–104.

[27] S. Grottel, G. Reina, J. Vrabec, and T. Ertl, "Visual Verification and Analysis of Cluster Detection for Molecular Dynamics," *IEEE Trans. Vis. Comput. Graphics*, vol. 13, no. 6, pp. 1624–1631, 2007.

[28] C. Turkay, J. Parulek, N. Reuter, and H. Hauser, "Interactive Visual Analysis of Temporal Cluster Structures," in *Comput. Graph. Forum*, vol. 30, no. 3, 2011, pp. 711–720.

[29] X. Ren and J. Malik, "Learning a Classification Model for Segmentation," in *Proc. Ninth IEEE International Conference on Computer Vision*, 2003, pp. 10–17.

[30] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, 2012.

[31] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in *Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967, pp. 281–297.

[32] S. Welstead and J. Nocedal, *Numerical Optimization*. Springer Science, 2006.

[33] M. Bostock, V. Ogievetsky, and J. Heer, "D3: Data-Driven Documents," *IEEE Trans. Vis. Comput. Graphics*, vol. 17, no. 12, pp. 2301–2309, 2011.

[34] Leaflet, "http://leafletjs.com."

[35] K. Perlin, "An Image Synthesizer," in *Proc. 12th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '85. New York, NY, USA: ACM, 1985, pp. 287–296.

[36] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image Quality Assessment: from Error Visibility to Structural Similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, 2004.

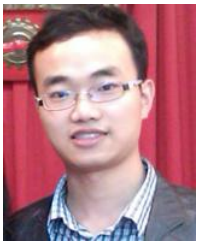
[37] S. Welstead, *Fractal and Wavelet Image Compression Techniques*. SPIE Optical Engineering Press, 1999.

[38] R. Wegenkittl, E. Groller, and W. Purgathofer, "Animating Flow Fields: Rendering of Oriented Line Integral Convolution," in *Computer Animation'97*, 1997, pp. 15–21.

[39] J. J. Van Wijk, "Image Based Flow Visualization," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 745–754, 2002.

[40] B. Sebastian, G. Andrienko, N. Andrienko, T. Schreck, and T. v. Landesberger, "Interactive Analysis of Object Group Changes over Time," in *Proc. International Workshop on Visual Analytics (EuroVA 2011)*, 2011, pp. 41–44.

[41] T. Von Landesberger, S. Bremm, N. Andrienko, G. Andrienko, and M. Tekusova, "Visual Analytics Methods for Categoric Spatio-temporal Data," in *Proc. IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2012, pp. 183–192.



Chenhui Li received his M. Eng. degree in computer science from East China Normal University, Shanghai, China, in 2011. He is working toward his Ph.D. degree in the Department of Computing, Hong Kong Polytechnic University, Hong Kong. His main research interests include information visualization and visual analytics.

George Baciu (M'90) received his Ph.D. degree in systems design engineering from the University of Waterloo, Waterloo, Canada. He is a Professor with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. He has published extensively and has served as a Chair in international conference committees, such as Pacific Graphics, Eurographics, Computer Graphics International, and CAD/Graphics.

Yu Han received his Ph.D. degree from Xidian University, Xian, China, in 2013. He is a lecturer in the College of Mathematics and Computational Science, Shenzhen University, China. His current research interests include variational and partial differential equations methods in image segmentation.