DDLVis: Real-time Visual Query of Spatiotemporal Data Distribution via Density Dictionary Learning

Chenhui Li, George Baciu, Yunzhe Wang, Junjie Chen, and Changbo Wang

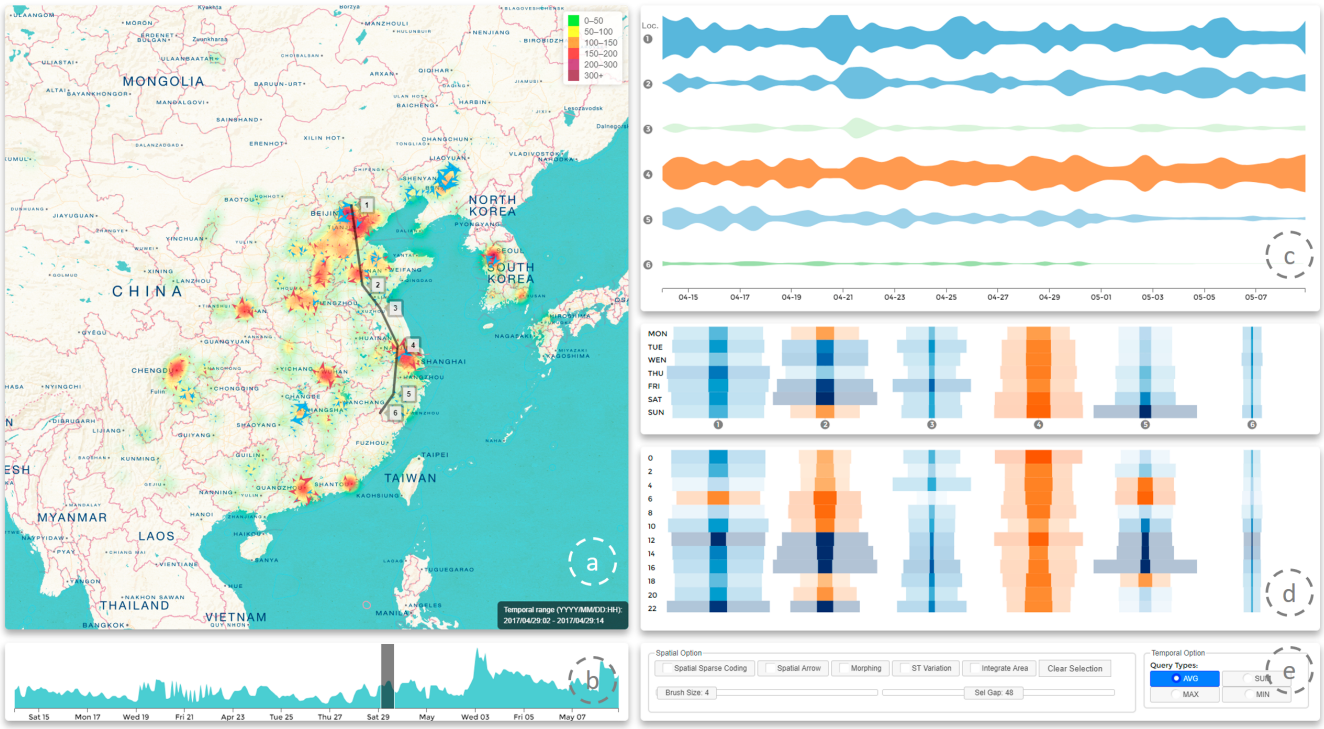


Fig. 1. DDLVis system interface. a) A spatial query is applied on a geographical map view; b) a time period is selected as query input; c) a stream view shows the temporal data of the selected key points; d) a statistical view provides the exploration of the temporal data according to day and hour; e) the option panels support parameter setting and query type switching. The color meanings in panels c) and d) are described in Sect. 6.5.

Abstract—Visual query of spatiotemporal data is becoming an increasingly important function in visual analytics applications. Various works have been presented for querying large spatiotemporal data in real time. However, the real-time query of spatiotemporal data distribution is still an open challenge. As spatiotemporal data become larger, methods of aggregation, storage and querying become critical. We propose a new visual query system that creates a low-memory storage component and provides real-time visual interactions of spatiotemporal data. We first present a peak-based kernel density estimation method to produce the data distribution for the spatiotemporal data. Then a novel density dictionary learning approach is proposed to compress temporal density maps and accelerate the query calculation. Moreover, various intuitive query interactions are presented to interactively gain patterns. The experimental results obtained on three datasets demonstrate that the presented system offers an effective query for visual analytics of spatiotemporal data.

Index Terms—Visual query, information visualization, spatiotemporal data, data compression, interaction, density map

1 Introduction

Spatiotemporal data are increasingly produced in real-time by physical sensors, such as vehicles, mobile phones, and climate monitors. Visual queries of the spatiotemporal patterns are beneficial to many domains, such as geographical research, urban environment estimation, and air pollution analysis. Some relevant visual query works have been previously proposed. Nanocubes [27] and imMens [28] provided a fast approach to query big data in real time in two-dimensional space. The further improvement includes the works of Hashedcubes [38], which offered a memory-efficient data structure for querying and exploring big data. However, most of the existing methods assume the query operation is applied on the raw dataset. When the data are sparse or dense in a two-dimensional space, the distribution representation approach, such as kernel density estimation (KDE) [24], is usually used to prioritize and represent the information in a dataset before pursuing higher levels of visual exploration. Many visualization works, such as Splatterplots [35], Hotspots [30], and StreamMap [26], show the benefits of the KDE-based approach. KDE-based approaches can overcome the visual clutter problem and help the user to comprehend significant patterns of the majority of spatiotemporal data when they include unorganized structures

- *Chenhui Li, Junjie Chen, and Changbo Wang are with School of Computer Science and Technology, East China Normal University. Changbo Wang is the corresponding author. E-mail: chli@cs.ecnu.edu.cn, cbwang@cs.ecnu.edu.cn. George Baciu is with The Hong Kong Polytechnic University. Yunzhe Wang is with Suzhou University of Science and Technology.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

and features.

To our best knowledge, less attention has been paid to the visual query of density-estimated spatiotemporal data in the information visualization field. Most visual query work focuses on the most common type of data, which is sparse points. To simplify the definition, the spatio-temporal data discussed later does not include line-based and region-based data. Density-estimated point data in two-dimensional space are normally generated via KDE. For example, the spatiotemporal air pollution data are normally incomplete because we cannot distribute infinite sensors in the world to detect the wide scope of the data. KDE is an alternative approach to produce a wide range of information about air pollution distribution. However, the reduction of the temporal density map size is challenging in a real-time visual query of spatiotemporal data distribution. For example, if the spatiotemporal data for distribution querying includes $10,000$ frames, one of which is estimated as a density map with 1024^2 pixels through KDE, then a distribution query operation, such as a mean query of the whole data period, on $10,000$ density maps requires approximately 10 GB of memory. Such memory consumption is unacceptable for a real-time visual query system, particularly when the system is running on a memory-limited mobile device. Besides, some works of Kim et al. [20] and Li et al. [26] present approaches to visualize the variational field between two spatial frames. However, while some researchers, such as Correll and Gleicher [8], have presented a visual query framework for understanding and exploring variation information in one-dimensional space, the query of variation in two-dimensional space is still a challenge.

To address these issues, we present a visual query framework, DDLVis, to fulfill the requirements of a real-time visual query of the spatiotemporal data distribution. Our method is proposed for storing, visualizing, and querying the density maps effectively. DDLVis aids users in obtaining insights regarding the distribution variation and evolution according to the selection. First, we propose a peak-based kernel density estimation (PKDE) method that transforms the raw spatiotemporal data into the sequential high-accuracy fixed-size density maps in an efficient way. The PKDE method ensures the performance of density map generation and requires a limited parameter setting for the temporal data. Second, we present a novel dictionary learning method to encode the temporal density maps to small-size sparse code before pursuing higher levels of the visual query. We consider the efficiency of the data storage and data processing to support a real-time visual query. Third, a peak-based variation generation model is applied on temporal density maps to provide an efficient processing approach to allow the user to comprehend the variation patterns of interest. We also present helpful interactions to simplify the query operation from two perspectives: spatial and temporal perspectives.

Our experiments show that our approach can be applied to the spatiotemporal datasets effectively and help to query and comprehend the temporal and spatial patterns. The main contributions of our work are as follows:

(1) We propose a variable kernel density estimation approach to present the spatial data in a structured format, which can enrich the visibility and reduce the preprocessing complexity of the query.
(2) We address a novel sparse coding approach through dictionary learning aimed at a low-memory storage of the density map and a fast process and query of the temporal density maps.
(3) We provide a technically efficient processing approach to generate and query the variation of the temporal density maps.

2 Related Work

For large-scale spatiotemporal data analysis, a convincing visual query framework is essential to rapidly find useful information. We review related methods on visual query approaches that include the following content: visual encoding of spatiotemporal data, visual query system, and dictionary learning.

2.1 Visual Query System

Early query processes are normally graphical interfaces of the traditional structured query language (SQL), as shown in the work of Derthick et al. [9]. Johansson et al. [17] presented an alpha blending method to address the querying of high-dimensional data. Recently, researchers have begun to focus more on visual big data queries due to the demand for data analysis. Ferreira et al. [13] constructed a spatiotemporal system that supports the interactive visualization of data patterns and potential details. Turkay et al. [51] visualized the variations in time and scale according to a brushing path on the geographical map. Slingsby and Van [49] proposed a similar brush-based interaction. There are many alternative methods are available for visualizing and querying the large-scale spatiotemporal data. Liu et al. [28] provided an interactive querying method on a large-scale dataset. Their system can support a rectangle-based interaction. Lins et al. [27] further presented nanocubes to support very large geographical data querying and browsing and demonstrated their system on a large Twitter location dataset. Correll and Gleicher [8] introduced a complete sketch-based visual query framework for understanding and exploring time-series data in one-dimensional space. A real-time visual exploratory approach, called TopKube [37], was proposed to find top-ranked objects in the spatiotemporal data. Then Pahins et al. [39] outlined Quantile Datacube Structure (QDS) to explore data patterns via order statistics. Later, a distribution-aware approach, RSATree [36], was proposed to support the visual query of large-scale tabular data. For the efficient selection for visual exploration, Guo et al. [14] formalized a spatial object selection problem and presented a new approach to select representative objects from the region of interest. However, the distribution variation and the temporal characteristics have not been considered in the prior visual query systems.

As the amount of data and time period become larger and larger, the previous method is somewhat challenging. On the one hand, the previous visual query methods need to design different data structures for different data density; on the other hand, the previous methods require a larger data storage space. A visual system for querying spatiotemporal data distribution is lacking. Compared with the prior methods, DDLVis aims a fast query approach for spatiotemporal data distribution in an intuitive way.

2.2 Visual Encoding of Spatiotemporal Data

Numerous approaches are presented for the visual encoding of the spatiotemporal data. CloudLines [23] is a timeline tool for visualizing event episodes in temporal data. Jänicke et al. [16] describe an interactive system to query and explore point-based data based on space and time via a dynamic Delaunay triangulation approach. Apart from the direct visualization timeline, the similarity measurement of time-series records has been discussed in [56], which is beneficial to the compatible visualization of continuous data. Visualizing patterns through timeline deformation is put forward by Bach et al. [2]. Their work overcame the space limitation problem of the prior timeline visualization while preserving the time information. Another work, called event cueing [29], studied the spatiotemporal distribution of evolving media discourse. Event cueing utilizes the benefit of the timeline and allows users to explore underlying spatial patterns. Brehmer et al. [5] surveyed several timelines and designed a hybrid timeline representation that combines different timeline representations in a three-dimensional design space. Data abstraction is another type of visual encoding approach to preserve users' important features. Shurkhovetskyy et al. [47] presented a comprehensive classification of data abstraction methods for temporal data visualization. Data abstraction methods also include the works of STULL [53] and Phoenixmap [59]. STULL presents a new pyramid-based sampling approach to reduce the data size for a geovisualization. Phoenixmap is a new visual design that deals with the multiple spatial distributions through enclosed outlines.

Another frequently used visual encoding method for spatiotemporal data distribution is the KDE (kernel density estimation), as described in the work of Silverman [48]. Many KDE-based studies have been presented to overcome the overdrawing problem and present the distribution. Scheepens et al. [46] offered a composite approach to present multiple density maps. They improve the density map visualization through contour enhancement. Hurter et al. [15] provided a KDE-based visual clustering approach to address visual clutter in complex graph drawing. An extension of KDE [35] has been presented to further overcome the overdrawing problem in visualizing high-dimensional data. Maciejewski et al. [30, 31] presented Hotspot to abstract the spatiotemporal data. Hotspot took advantage of both KDE and timeline to visualize the variation evolution. Willems et al. [55] adopted KDE to address the problems involved in visualizing moving objects. Perrot et al. [42] outlined a GPU-based approach to estimate the point density. For the analysis of spatiotemporal data without trajectory information, Kim et al. [20] and Li et al. [26] provided trend visualization models to present the spatial variations on a two-dimensional map.

The KDE approach can ignore the impact of data scale since it can map large-scale spatial data of a certain time period to the normalized density map. Although the KDE approach provides a good visual representation for a large-scale dataset, visual query over temporal density maps is challenging

in terms of efficiency when the system memory is limited. As far as we know, no previous studies have implemented a visual query system on temporal density maps. DDLVis enriches visibility and reduces the preprocessing complexity of the query.

2.3 Dictionary Learning

The dictionary learning approach [1] aims at learning the most pristine features behind the raw data and representing data via sparse codes. The learned dictionary contains prototype atoms that can describe the raw data via linear combinations of the prototype atoms. Dictionary learning has been adopted in many visual media applications, such as signal compression [6], classification [58], feature extraction [7], image super-resolution [57], and compressive sensing [10]. Signal compression is an intuitive application of dictionary learning because only a dictionary and a sparse code are required to be stored. Signal compression through dictionary learning is similar to the usage of JPG2000 [34], which compresses the natural image with a high compression ratio through wavelet transforms. The difference is that JPG2000 standard code requires a complete wavelet dictionary. The calculation of the wavelet transform method is time-consuming.

K singular value decomposition (K-SVD) [1]is an effective method to learn the image sparse representation from many samples. Further improvements of K-SVD have been discussed in the work of Elad et al. [11]. With the rise of deep learning research, CNNs (convolutional neural networks [40] are also used in the dictionary learning approach. Since dictionary learning can represent as much knowledge as possible with few resources, it has the potential to be used in large-scale processing and analysis of spatiotemporal data.

Therefore, because of the demand of large-scale spatiotemporal data query and visualization, and considering the shortcomings of visual queries on distribution data and the advantages of KDE and dictionary learning methods in data expression, we proposed the DDLVis method, so far lacking in the prior studies. DDLVis aims at a low-memory storage implementation of the density map and a fast and abundant query of the spatiotemporal data distribution.

3 System Overview

The pipeline of our system is shown in Fig. 2. The input data of our system are sequential frames as shown in Fig. 2(a). Each frame is a set of spatial points with a timestamp. We define a frame as F_i, where i indicates the timestamp. Since the location points are common data in geographical applications, we assume the spatial points in our system are locations of sensors or people. Each frame is processed as a density map with the same size through PKDE as shown in Fig. 2(b). A training set generation (TSG) method selects the representative density maps for training, as shown in Fig. 2. The dictionary learning method is applied on selected density maps, as shown in Fig. 2(d). With the trained dictionary, the density maps can be represented as a set of small-size sparse codes. Fig. 2(e) shows the interface of our proposed system.

Our system is constructed as follows:

(1) To calculate the temporal data distributions, we present an aggregation method called PKDE to estimate a series of accurate density maps during a period to represent the spatiotemporal data in a regular form. PKDE produces accurate temporal density maps via an adaptive estimation kernel selection approach.

(2) To reduce physical memory usage and accelerate the query process, we present a novel learning method, density dictionary learning (DDL), which applies the dictionary learning method on the temporal density maps.

(3) To assist the query process and represent the query result, we design interactive operations to help users with querying and comparing the evolution, variation, and statistical information of the spatiotemporal data intuitively. We provide abundant visual effects in accordance with the query interactions.

4 Peak-Based Kernel Density Estimation

The peak-based kernel density estimation (PKDE) approach is presented to produce the data distribution by estimating the data density in a region. For each frame of the spatial data, we apply a peak-based clustering method to find the clusters. The data points belonging to the same cluster will be processed with the same kernel size in the density estimation process as shown in StreamMap [26]. We assume that the kernel size h is related to the

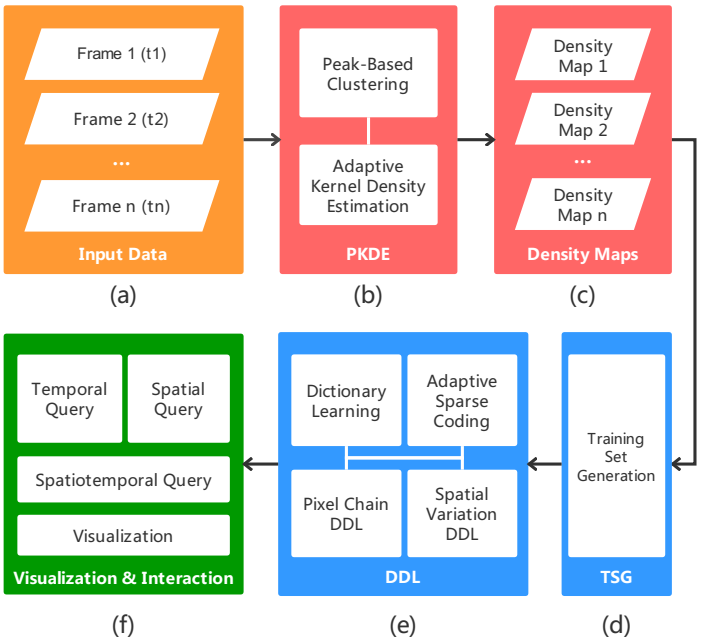


Fig. 2. The pipeline of our visual query system. The proposed system includes three main components: PKDE, DDL, and the visualization and interaction for the visual query.

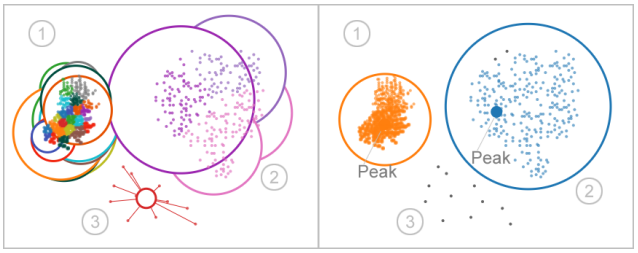
point count n and the spatial boundary box area sa of the cluster. We define $h = \lambda\sqrt{sa/n}, \lambda \geq 2$, where λ is an experimental parameter that ensures the estimated density map is a single connected region.

KDE [48] is a statistical method to estimate the distribution of spatial data. For sparse data, KDE represents the potentially influenced regions. For high-density data, KDE reduces the visual clutter as discussed in the visualization work of Roeland et al. [46]. As described in the work of Silverman et al. [48]. The kernel size selection is a key issue of the density estimation. When KDE is applied to sequential data frames, the kernel size setting is challenging. Therefore, we propose a variable kernel density estimation method, PKDE, to adaptively adjust the estimation kernels for spatiotemporal data.

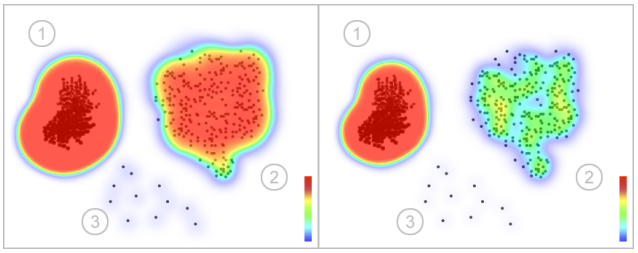
In the proposed approach of PKDE, we assume that the input data in our system are spatiotemporal data in two-dimensional space as shown in Fig. 3(b). The elements of each data record include a 2D location, feature index, and timestamp. The feature index depends on the type of dataset. For example, the feature index means the air quality index if the spatiotemporal data are related to an air quality dataset. Traditional k-means clustering [32], DBSCAN [12], and superpoint-based clustering (SKDE) [26] are not suitable for common nonspherical clustering tasks. The nonspherical clustering means the shape of the cluster is not spherical. Our proposed PKDE aims to produce nonspherical clusters and it guarantees the cluster coherences. PKDE requires fewer calculation resources. Thus, it is more suitable for sequential density estimation.

The PKDE approach not only generates a more accurate density map than using SKDE but also provides the peak point candidates for the next variation generation. Variation information is the base data for the variation query in DDLVis. We will discuss the advantage of PKDE for the preprocessing complexity reduction of the query in Section 5.5. There are three steps of the PKDE. First, we propose to adopt a peak-based clustering approach (CFSFDP) [44] to find the peak-element and then cluster the elements in the first data frame. Second, to generate the coherence of the clusters between two frames, predetermined peak-elements are used to accelerate the element clustering in the next data frame. Third, when the elements in each frame are clustered, the estimation kernels of each cluster are fixed. Then, we estimate the density of a frame with the corresponding kernel size.

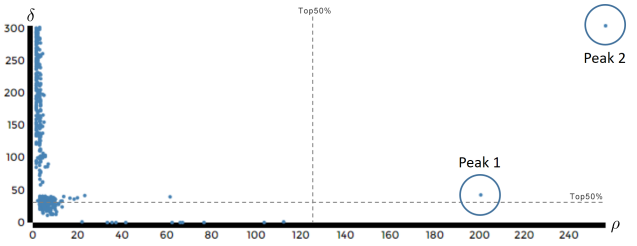
PKDE assumes that the cluster peak is always surrounded by other elements with a lower density, and the cluster peak is far away from other elements with a higher density. Following the CFSFDP [44] method, we



(a) K-means clustering (b) Peak-based clustering



(c) KDE via K-means (d) PKDE result



(e) Decision graph of PKDE. Two found peaks indicate two clusters.

Fig. 3. Estimated density maps through different approaches. Circles in (a) and (b) indicate the clusters.

define the local density of each element ρ_i as follows:

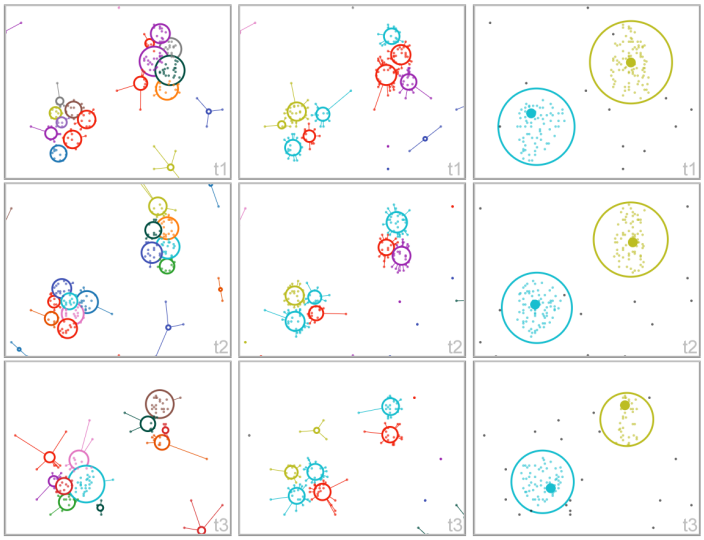
$$\rho_i = \sum_{x_j \in S} \psi(dist(x_i, x_j) - dist_c), \psi(x) = \begin{cases} 1, (x \leq 0) \\ 0, (x > 0) \end{cases} \quad (1)$$

where $dist$ means the Euclidean distance, S is the element set of a data frame, and $dist_c$ is a cutoff parameter. δ_i represents the minimum distance between element i and other elements with a higher local density. ρ_i is the number of points within a certain distance of a point x_i. δ_i is formulated as follows:

$$\delta_i = \min_{j:\rho_j > \rho_i} (dist(x_i, x_j)) \quad (2)$$

After the ρ and δ of each element i are calculated, a decision diagram is drawn to find the peak of a dataset. The point at the upper right corner of the decision diagram is the peak element of the cluster. Fig. 3(e) is an example. The clustering principles are as follows: (1) The elements with very low density are usually noise. (2) If an element's ρ and δ are with the top 50%, it will be selected as a peak. The peak selection must meet the conditions of $\delta_i \geq \frac{1}{2} \underset{j \in S}{\text{Max}} (\delta_j)$ and $\rho_i \geq \frac{1}{2} \underset{j \in S}{\text{Max}} (\rho_j)$. The remaining elements will be assigned in the cluster where the peak is closest to its location. (3) Selected peaks will be used as references in the next frame clustering. It is easy to find two peaks in Fig. 3(e). Fig. 3(a-d) shows the density estimation results using different approaches. Fig. 3(b,d) shows that PKDE has better performance of data clustering and density estimation than that of k-means as shown in Fig. 3(a,c). We evaluate the performance among k-means, SKDE, and PKDE in Section 8.

After the density estimation, the spatiotemporal data in a period can be represented as a number of regular density maps. We assume the size of each density map is the same. The estimated sequential density maps can be formulated as $\boldsymbol{DM} = \{D_1, D_2, ..., D_m\}$ where m is the count of density



(a) K-meansns (t_1, t_2, t_3) (b) SKDE(t_1, t_2, t_3) (c) Ours(t_1, t_2, t_3)

Fig. 4. Temporal clustering results through different strategies.

maps. Temporal clustering results through different strategies are shown in Fig. 4. The clustering continuity of our method is better than the other two approaches. The PKDE method also has shortcomings. When the clusters of two frames do not have any overlap, the effectiveness of the PKDE method on time series data is weak.

5 Density Dictionary Learning

Our system provides a variety of visual query approaches to query data from large-scale spatiotemporal data. These methods require the high efficiency of loading and processing. Therefore, we propose to use dictionary learning to encode the density map. Dictionary learning can reduce the time overhead of data transmission and loading, thus it improves the efficiency of the visual query. We assume that a sequence of time-series frames, each frame is a density map, has been generated using the PKDE method. Next, we will introduce how to use DDL method to encode the density map to generate a smaller sparse code.

The basic idea of DDL is a training process that learns a dictionary to represent the local patterns of density maps. DDL has a low demand for calculation memory and accelerates the query process. Our DDL method includes three steps. The first step is a training set generation process that selects the representative density maps. In the second step, we use the dictionary learning approach to iteratively find the optimal dictionary. A dictionary consists of a number of eigenvectors. The third step is to convert density maps into sparse codes (sparse eigenvalues) by a sparse coding method based on a trained dictionary.

To support the real-time visual query, we apply the DDL method to temporal density maps generated via PKDE. The learned dictionary and generated sparse codes are produced to support the high-performance visual query on the data in a period. We generate the sparse codes for each density map. The sparse codes and the dictionary will be stored for the further visual queries and visualizations. By using a dictionary and sparse codes, we can represent any of the density maps that belong to a specified type. Section 8 shows the effectiveness of DDL.

5.1 Training Set Generation

Learning the patterns from one density map is not sufficient to make it accurately represent the whole dataset. Additionally, it is a time-consuming task to learn the dictionary based on all density maps. Hence, we select representative density maps as the training set to learn the patterns. A specific clustering method is used to cluster similar density maps together. One cluster provides only one density map. The proposed approach will not only improve the training accuracy of dictionary learning but also reduce the size of training set.

Training set generation (TSG) is a fast and effective density map clustering approach. TSG takes advantage of the perceptual hash algorithm (PHA) [18] and spectral clustering method (SCM) [52]. SCM is an improvement of the k-means approach [32]. First, the original density map is scaled to a 16×16 image. Second, the mean pixel value is computed. Third, we initialize a 256-bit hash code via PHA. If the pixel value is larger than the mean pixel, we set the corresponding bit to 1. Otherwise, we set it to 0. Because there are a large number of time series density maps, it is inefficient to directly calculate the image similarity on the grayscale image. Hash code calculation is high-performance. Fourth, based on the calculation of two 256-bit hash codes, we can calculate the similarity matrix of density maps through the Hamming distance comparison [3]. Based on the similarity matrix, density maps are clustered via SCM. The K parameter setting of SCM is related to the global type number of the dataset. For example, if the global types of an air quality dataset are summarized as *Good*, *Moderate*, *Unhealthy*, and *VeryUnhealthy*, K is set to 4. One density map is selected from a cluster. We define the final selected density maps from different clusters as $\boldsymbol{TS}$. Fig. 5 shows the TSG pipeline.

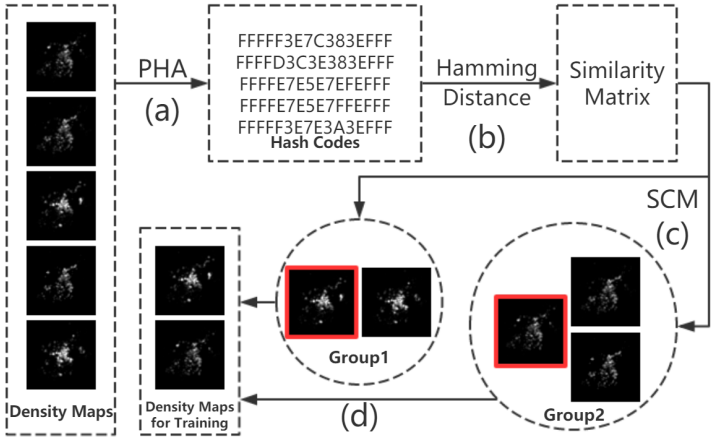


Fig. 5. The pipeline of training set generation (TSG). (a) The original density maps are scaled to a 16×16 image. A 256-bit perceptual hash code is generated for each scaled density map. (b) The similarity matrix of perceptual hashes are calculated. (c) Based on the similarity matrix, SCM is applied to cluster the density maps into different groups. (d) Representative density maps from the clusters consist of the training set.

5.2 Dictionary Learning

Dictionary learning is a process that finds the eigenvectors (representative patterns) of the input data. The input data in our approach is a patch of the density map. There are three elements in a dictionary learning process such as Y (patch), A (dictionary), and X (sparse code) as shown in Fig. 6. Y indicates a patch of the density map. A stores the learned optimal eigenvectors. X is a sparse code that stores the eigenvalues. Fig. 6 also shows the basic idea of recovering the patch from a dictionary and a sparse code. In DDLVis system, the selected density maps ($\boldsymbol{TS}$) are divided into m patches with a 16×16 size. The patch size of 16×16 is an experimental value. Hence, each patch is represented as a 256-dimensional vector. Sparse code takes very little memory and represents the eigenvalues of a patch. We use three float numbers (12 bytes) to represent a 256-dimensional vector. The experimental minimum storage requirement for a sparse code is 12 bytes. Details of the dictionary learning process are listed as follows.

We assume A_{init} as the initial dictionary ($256 \times m$ matrix) that is set via random pixel sampling from the selected density maps ($\boldsymbol{TS}$). m indicates the eigenvector number of the dictionary. The final optimal dictionary is defined as A. $\boldsymbol{y_i}$ means a density map patch. $\boldsymbol{x_i}$ means a sparse code of the density map patch with respect to the dictionary A. i indicates the patch index. Based on A_{init}, $\boldsymbol{x_i}$ is calculated by using orthogonal matching pursuit (OMP) [50]. The purpose of OMP [50] is to disassemble a known signal into a weighted sum of many atomic signals and to find the eigenvalues for the atomic signals. The number of eigenvalues in the OMP is fixed.

$$\min_{A,x_i} \sum_{i=1}^{N} \|y_i - Ax_i\|_2^2 + \lambda \sum_{i=1}^{N} \|x_i\|_1 \tag{3}$$

When all sparse codes are calculated, we update the initial dictionary through SVD [1]. An optimization model as shown in Equ. 3 is designed to minimize the error energy of the sparse coding. Therefore, DDL is an iterative optimization process that generates the optimized dictionary and the optimized sparse code. Fig. 7 offers an intuitive representation of a density map dictionary.

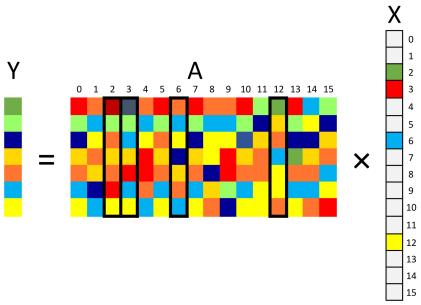


Fig. 6. A simple example shows the dictionary learning ideas. Only a dictionary (A) and various sparse codes (X) are required to store. The data (Y) can be recovered from A and X.

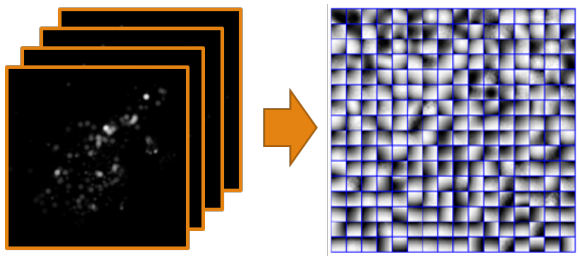


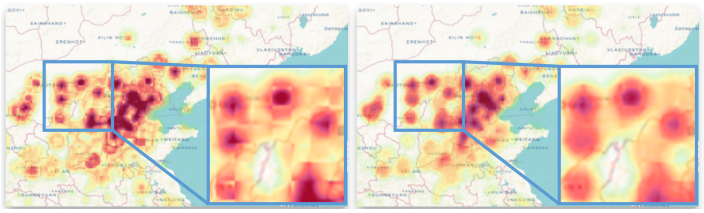
Fig. 7. An example of the learned patterns in a dictionary from the training set of density maps.

5.3 Adaptive Sparse Coding

When the dictionary is learned, we can adopt the OMP method to generate a sparse code for each patch of the density maps. Directly applying OMP to generate the sparse code will lead to the error appearance as shown in Fig. 8(a). Since the valid eigenvalue number of the traditional OMP method is fixed, the OMP coding is not accurate enough to present the important information on the density map. The density map patch has various forms, which cannot be expressed well with limited eigenvalues, and the expression with a large number of eigenvalues will cause the sparse code to be too large. Hence, we propose to use an adaptive sparse coding (ASC) approach to adjust the eigenvalue number adaptively for different forms of density map patch. ASC method guarantees the high accuracy of sparse representation by optimizing the number of eigenvalues. The sparse patch will be represented via less number of eigenvalues than the non-sparse patch. ASC is inspired by Sadeghi et al. [45]. It is an improvement of the OMP method through error optimization. ASC can achieve high accuracy by using an adaptive number of eigenvalues. ASC has a good performance on obtaining results of visual queries. Fig. 8(b) shows a better sparse coding result than that of Fig. 8(a).

5.4 Pixel Chain DDL

The previously proposed DDL approach is designed for sparse coding of the density map patch. In addition to density map, querying time-series data from a certain spatial location is also an important part of a visual query system. We assume that all density maps have a fixed number of pixels (e.g., 1024×1024). We define the pixel in the same two-dimensional position on the sequential density maps as a pixel chain, as shown in Fig. 9. Similar to the proposed DDL, we cluster the pixel chains and select the representatives



(a) Without error optimization. (b) With error optimization.

Fig. 8. A result comparison of two sparse coding approaches. We enlarge the details. Clearly, the result quality on the top is low, whereas the result quality on the bottom is high.

to build the training set. By using dictionary learning, we can generate a pixel chain dictionary. For each pixel chain, we calculate the sparse code. Only the sparse code and the pixel chain dictionary are loaded in the visual query system.

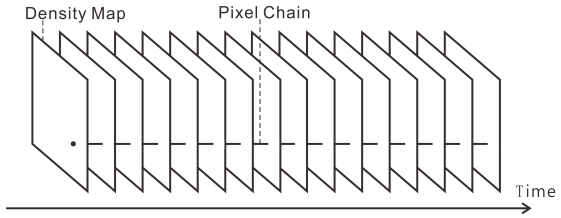


Fig. 9. An example of the pixel chain of the spatiotemporal data.

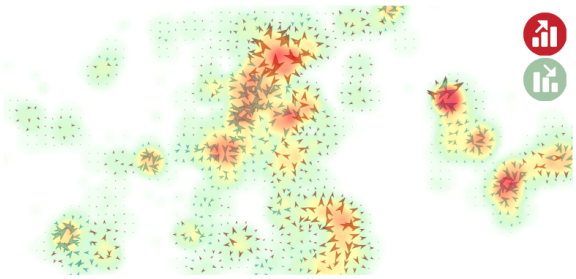


Fig. 10. An example of the spatial variation field between a pair of the adjacent density maps.

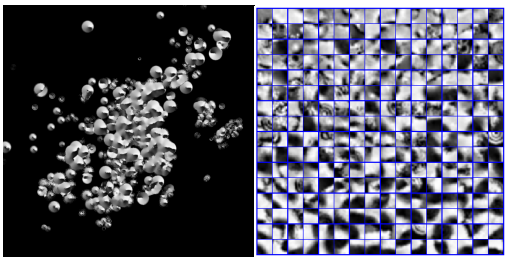
5.5 Spatial Variation DDL

Visualizations of spatial variation have been discussed in many works, such as StreamMap [26]. Based on the work of StreamMap, we generate the spatial variation field for each pair of the adjacent density maps. Fig. 10 shows an example of the spatial variation field. Since we have found the peaks of the frame in the PKDE approach, we can accelerate the variation field generation of StreamMap [26]. The generated spatial variation field will be converted to a direction map and an absolute map. The direction map as shown in Fig. 11(a) stores the spatial variation angle ($[0,359]$) and the variation-size map stores the absolute value ($[0.0,1.0]$) of the spatial variation. The representation of an absolute map is similar to the density map. We apply a dictionary learning method to compress the direction maps and absolute maps. Only the sparse codes and two dictionaries are required to load for the querying of the spatial variation information. Fig. 11 shows a dictionary trained on patches from the direction map.

Finally, we summarize the DDL pipeline in Algorithm 1. The input of DDL is various sequential density map data, such as $\boldsymbol{DM}$. The output of DDL is a dictionary ($\boldsymbol{A}$) and sequential sparse codes ($\boldsymbol{X}$). We believe the sparse coding method is an effective data encoding method to support a query on large numbers of continuously variable data. We evaluate the performance of the proposed method in Section 8.

6 Visual Query

We load the sparse codes constructed in Section 5 before the visual query. According to the time and space parameters involved in the query, the



(a) Direction map (b) Trained dictionary

Fig. 11. An example of a direction map and a dictionary trained on patches from the direction map.

Algorithm 1 DDL algorithm

procedure DDL(DM)
  $TS \leftarrow TSG(\mathrm{DM})$ ▷ Training set generation.
  $A \leftarrow Sampling(TS)$
  $X \leftarrow \mathbf{0}$
  $i \leftarrow 0$
  while $e > \tau$ **do** ▷ τ indicates the error goal.
    $X', A' \leftarrow OMP(TS, A, X)$ ▷ Orthogonal matching pursuit.
    $X \leftarrow X', A \leftarrow A'$
    $e = \sum_{i=1}^{N} \|y_i - Ax_i\|_2^2 + \lambda \sum_{i=1}^{N} \|x_i\|_1$
  end while
  while $i \leq n$ **do** ▷ n indicates the number of density map patches.
    $X_i \leftarrow ASC(DMP_i, A)$ ▷ Adaptive sparse coding
    $i \leftarrow i + 1$
  end while
  return $\boldsymbol{X} = \{X_0, X_1, ..., X_n\}$, A
end procedure

corresponding sparse codes are decoded to the approximate original data. We address various interaction approaches to freely select multiple regions or periods on the spatial map or timeline. Since the query result visualization is an important part of a visual query system, we design special representations to visualize the query results.

6.1 Query Definition

We define the query as an interactive operation with five input parameters, such as query form, query type, query period, selection size, and key point. Fig. 12 shows the query definition. For the spatiotemporal data, we define two query forms to satisfy the query requirements from two perspectives: temporal query and spatial query. The query types are defined as AVG, MAX, MIN, and SUM. The query type can be extended according to the specific requirement. The query period indicates the time range. The selection size is defined to set the selection range on a geographical map. The key point defines the sampling positions over the query region.

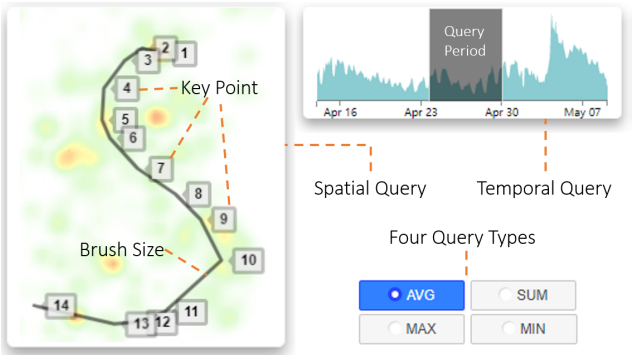


Fig. 12. Query definition. We define two forms of query operations. Spatial query means the operation is applied on a geographical map. Temporal query means the operation is applied on a time period.

6.2 Temporal Query

We define the query operation on the timeline view as the temporal query (TQ). It is designed to query the sequential density maps according to the time period. A real-time brushing operation is allowed on the timeline view. We apply the dictionary learning of the density map patch for the temporal query. The query input of TQ is a period of time. The brush interval can be set by the user. The output of the temporal query is a statistical density map. Statistical information includes the maximum value, the minimum value, and the mean value of the density maps in a specified period. We directly visualize the statistical density map over the geographical map to show the query result.

6.3 Spatial Query

We define the query operation on the geographical map as the spatial query (SQ). The query input of SQ is a series of geographical positions. Multiple positions can be selected through a query operation on the geographical map. The query operations include: *brush*, *pointbypoint*, *lasso*, and *rectangle*. *Brush* will generate a brushing trajectory that includes a set of points. Points are generated through evenly dividing the brushing trajectory. The brush-based interactive operation helps users query and compare the evolution, variation, and statistical information of the spatiotemporal data in an intuitive way. For example, it is convenient when the querying regions are along coastlines or rivers on a geographical map. *pointbypoint* lets the user manually select the key points by clicking on the area. *Lasso* and *rectangle* will create a region where we will generate the key points through average sampling. We define the average sampling parameter as avg_{samp}. avg_{samp} indicates the interval of the key points. The selection of the avg_{samp} is related to the query speed. For the SQ, we adopt the stream view and statistical bar to visualize the query result. Since the spatial query requires the result in a whole period, we apply Pixel Chain DDL to speed up the query performance.

6.4 Spatiotemporal Query

The spatial variations among a time period are difficult to explore on a geographical map. Based on the Spatial Variation DDL discussed before, we can load the variation data effectively when the query for the spatial variations is required.

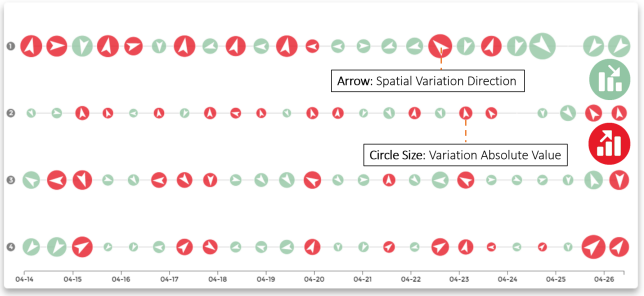


Fig. 13. Query result of our spatiotemporal query approach. An arrow indicates the spatial variation direction, meanwhile the circle size outlines the variation absolute value.

6.5 Query Result Visualization

According to the characteristics of the spatiotemporal data, we design three visualization forms to visualize the query result.

Stream View. The stream view is a stream-based design to show the data evolution in a period. We depict the query result vertically, as shown in Fig. 16. All of the spatial chains in different timestamps are rendered as a stream with a different color, where the stream width indicates the amplitude. The color setting is based on three principles as follows: (a) warm color, such as dark red, is chosen to render the continuously large streams; (b) cold color, such as deep blue, is chosen to outline the continuously small streams; (c) medial color, such as green, is chosen to outline the streams that are continuously small. The contrast color usage follows the work of Kim et al. [19].

Statistical Bar. Since the analysis of day-related and hour-related data is frequently required in the visualization application, we design a new statistical bar that can present the maximum and mean values. The spatial id on the view of the statistical bar links to the key points on the geographical map. In our experiments, we show the information about seven days of a week and twelve hours of a day. Fig. 14 outlines two forms of statistical bars. The center of the horizontal bar indicates the mean, and the width of the horizontal bar means the maximum. The color setting principles are similar to the stream view (Section 6.5).

Variation View. Fig. 13 shows the query result of our designed spatiotemporal query approach. A red circle indicates an increasing variation, whereas a cyan circle means a decreasing variation. The circle size presents the variation absolute value.

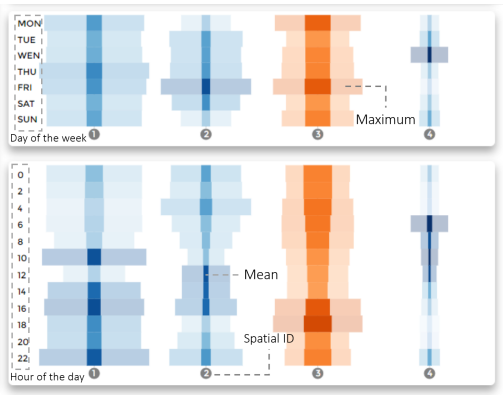


Fig. 14. A description of the statistical bar visualization. (top) Statistical bar for the day of the week. (bottom) Statistical bar for the hour of the day. Four columns are related to four selected positions on a geographical map.

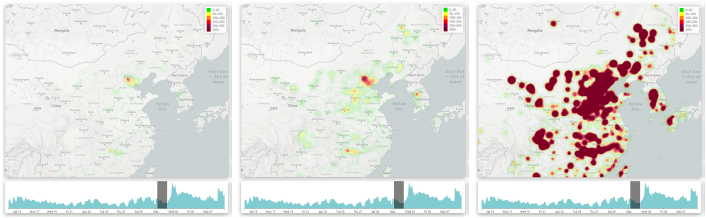
7 Case Studies

The experiments are performed on a MacBook Pro with an Intel Core i7 CPU and 16 GB RAM. We use the open-source libraries D3 [4] and Leaflet [25] to provide interactions and visualize the query results. All density maps are visualized through color mapping on the geographical map.

7.1 Air Pollution

To help users query the air pollution distribution, we apply our system to a real air pollution dataset. This dataset involves the data of the air quality index (AQI) in China. They were collected from aqicn.org over 75 days from April 14th, 2017, to June 27th, 2017. AQI data were monitored at 3365 monitoring stations. Each record includes an AQI value and the corresponding monitoring location. A high AQI means a bad air quality. Nine hundred density maps were generated in advance through PKDE. Since the air pollution monitoring stations are limited, it is significant to estimate the air pollution in the regions without physical monitors. We adopt DDLVis to visualize the air pollution information and query the potential polluted regions from a temporal perspective and a spatial perspective.

DDLVis provides an interactive operation to compare the different AQI distributions at two time periods as shown in Fig. 15(a-c). Fig. 15(a-b) shows the query results following the MIN and AVG query type. When the query type is changed to MAX, the places that have been very polluted over the time period will be visualized, as shown in Fig. 15(c).



(a) Period selection through MIN query. (b) Period selection through AVG query. (c) Period selection through MAX query.

Fig. 15. DDLVis is applied on an air pollution dataset.

We try to use the SQ to analyze the air pollution distribution from eastern China to western China. Pixel Chain DDL supports the pollution distribution data query in a high-speed way. We use a *brush* operation to draw a path

from Nanjing city to Chengdu city and obtain the result, as shown in Fig. 16. Fig. 16 depicts that Nanjing and Chengdu cities have bad air pollution in the selected period. The air quality of some regions, such as region Chengdu in Fig. 16, was varying in a very large range over 75 days. Figure 22 shows a stream information pair for the Pixel Chain DDL accuracy evaluation. The result demonstrates that the sparse coding accuracy satisfies the requirements of the visual query. We also apply the Spatial Variation DDL on AQI dataset and visualize the spatial variation and evolution of four regions. Fig. 13 shows that the first region is often exposed to air pollution from the South. Besides, the circle size indicates that the AQI variation is strong in this region.

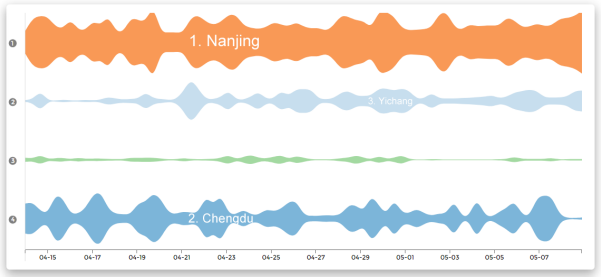


Fig. 16. Air pollution query result from eastern to western China.

7.2 Flight Noise

We found that sound from flights had a detrimental noise influence on the people living near the airport. To help the residents avoid the noise, we try to apply our system to explore the noise impact of air traffic. Since the streaming trajectories from aircraft consist of high-density geographical points, it is suitable to build the density map on the geographical map. Over fifty days of flight trajectories near Shanghai Hongqiao Airport from Jan 12th, 2016, to Mar 15th, 2016, were collected from flightradar24.com.

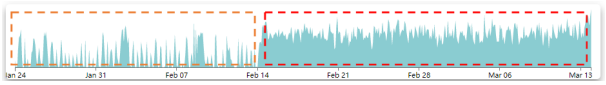


Fig. 17. Statistical information of the flight noise amplitude.

We merge all the points of daily trajectories into a frame. We apply the PKDE method to estimate the noise distribution. Different Gaussian kernel sizes in the PKDE indicate the noise impact size. Then, we can generate the corresponding density maps of the noise for each frame. Since it is difficult to calculate the flight noise accurately, noise density maps are helpful to visualize and query the noise impact.

By using DDLVis, much useful information can be explored. The view for the temporal query can offer an overview of the flight noise as shown in Fig. 17. Although Fig. 17 demonstrates that the flight noise after Feb. 13 is greater than before, it is also necessary to query the flight noise information on a spatial space. We adopt the spatial query and draw the key points crossing the airport on the geographical map. Clearly, the residents near the airport are affected by flight noise. Hence, understanding the specific areas that will be affected by the airport is an unknown issue. We adopt DDLVis to explore the temporal noise from east to west as shown in Fig. 18. It was a significant finding that the affected regions to the west of the airport are larger than the regions to the east of the airport. In addition, the query type of SUM is interesting in this case. We calculate the summation of all the density maps through the temporal query during all periods. We found that the top three regions are affected less in the whole period. We believe that the regions of A, B, and C are suitable for living with less influence from the flight noise.

8 Evaluation

To evaluate our approach, we applied it to a large-scale dataset called Xingyun Map [33]. Xingyun Map is a dataset collected from the Tencent map website, which presents the geo-locations of the map users. The total number of user records per day is nearly 10 billion. We aggregated the user locations per 5 minutes into a density map. Finally, we generated

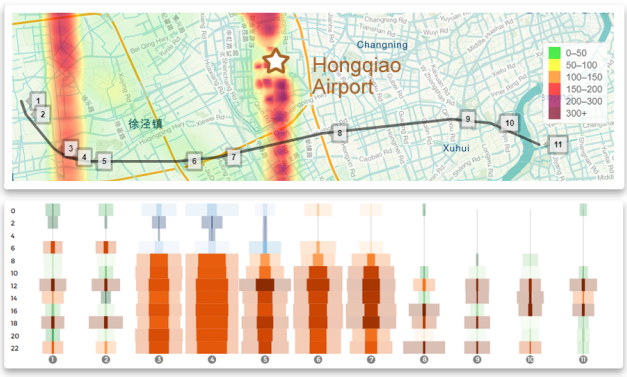


Fig. 18. Spatial query from the west to the east.

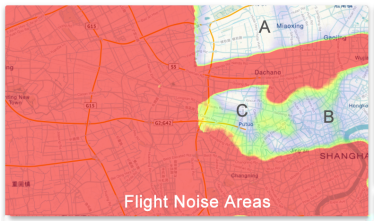


Fig. 19. Temporal query using SUM type. A, B, and C are suitable for living with less influence from the flight noise.

18,247 density maps (frames) to represent the data in 64 days. PKDE and the dictionary learning is applied on the dataset before the query. When the data space is 1024×1024, the PKDE time cost for one data frame is nearly $0.08s$. The time cost of the dictionary learning is nearly $1.2s$ for a representative density map (1024×1024).

8.1 Query Performance

DDLVis can support a real-time interactive query over the spatiotemporal data. We evaluate the time cost of our visual query method. The resolution of the query space is 1024×1024 on the display. The dataset for measurement is the Xingyun Map. To the best of our knowledge, the real-time visual query of the spatiotemporal data distribution has not been investigated before. HashedCubes [38] is the most relevant one. If the spatial resolution of the cubes is small enough, HashedCubes is an approximation of density estimation for a fixed bandwidth. Besides the time performance, we also focus on the distribution query and a few special query forms such as the spatiotemporal query. The generated density maps are estimated through PKDE. Although the visual query pipelines of traditional work (e.g. HashedCubes) and our approach are different, we are trying to evaluate the time cost in a closely same experimental condition.

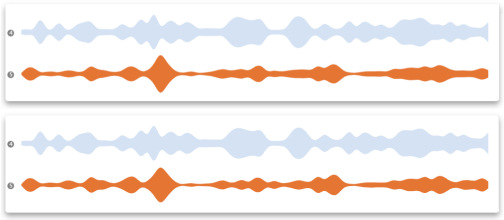
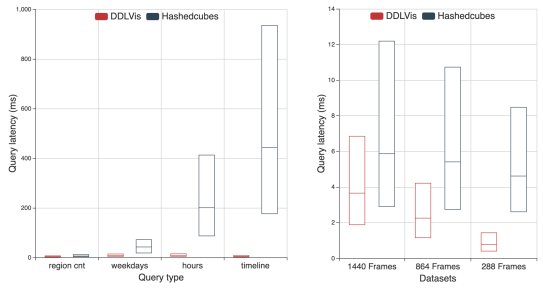


Fig. 20. A pair of stream information instances for the DDL accuracy evaluation. (top) Original stream information. (bottom) Stream information with sparse coding.

We tested DDLVis and Hashedcubes on the same three datasets with respectively 1,440 frames, 864 frames and 288 frames, using a set of 10,800 queries. Frames are randomly selected from the Xingyun datasets, and the selection process avoids repetition. The size of the three datasets (number of records) is respectively 241,894,250, 147,373,025 and 48,154,078. The map is focused on one location and the same zoom level in all test cases. We set avg_{samp} as 8 pixels when we apply *rectangle* query in DDLVis system. Since HashedCubes is not designed for distribution query, no sampling

interval is applied. We apply *rectangle* query in HashedCubes system. The rectangle is randomly selected in the range of query space. Results of four types of queries shared by both systems were recorded, which are as follows: 1. region count, 2. day of the week, 3. hour of the day and 4. timeline. We built Hashedcubes using the schema where spatial information is firstly encoded, then day of the week, hour of the day as categorical information, ending with a temporal dimension.



(a) Different query types (b) Different datasets

Fig. 21. Query performance results. (a) The latency of different query types at the largest datasets with 1440 frames. (b) The latency of querying "region count" at different datasets.

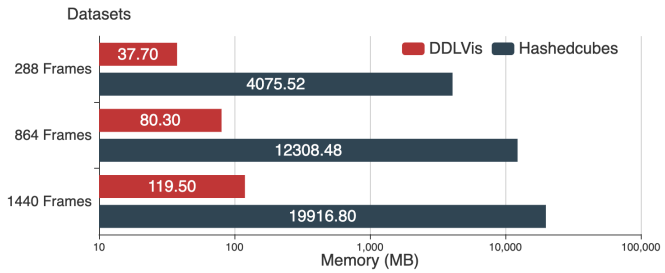


Fig. 22. Memory usage of DDLVis compared with Hashedcubes.

Fig. 21(a) reveals different relations between the two systems about the type of query and the latency. At each type of query, DDLVis outperforms Hashedcubes. The latency of "timeline" is similar to that of "region count", and the latency of "day of the week" is similar to that of "hour of the day", while Hashedcubes shows an "increasing" trend along the dimension of the query type. We assume this "increasing" trend was decided by the order of dimensions when building Hashedcube, for the number of pivots increases when subdividing records at each dimension. Fig. 21(b) shows the relationship between query latency and the size of dataset. The two systems both show a positive correlation. The overall performance of DDLVis is better with lower latency and variance. The increasing rate of query latency of DDLVis is larger than that of Hashedcubes, which may prove less scalability on a larger dataset. Fig. 22 shows the memory costs of both systems on the three datasets. DDLVis costs much less memory than Hashedcubes, which is because we only keep sparse codes and dictionaries rather than information of each record. We conclude that the DDL-based approach is suitable for a large-scale query of spatiotemporal data distribution.

8.2 Compression Quality

Because sparse code is small, it is necessary to calculate the similarity between the original density map and transformed sparse code. We adopt the SSIM [54] method to evaluate the compression quality of the DDL approach. SSIM considers the structure of two images; thus, it can represent the difference between two data frames. SSIM is used to calculate the similarity of two density maps: one is an original density map; the other is a compressed density map using DDL. In our experiments, the calculated average SSIM is 96.4%. This result indicates our proposed approach has a high-quality coding performance. We also use SSIM to calculate the similarity of two pixel chains. We set the stream view as the images for the comparison. The average calculated similarity is 99.3% in our experiments. Fig. 20 shows a stream information pair for the Pixel Chain DDL accuracy evaluation. The result demonstrates that the sparse coding accuracy satisfies the requirements of the visual query.

Table 1. Average similarity comparisons of the different density estimation approaches.

Methods	K-means	SKDE [26]	PKDE
Benchmark	0.748	0.787	**0.804**

8.3 PKDE Performance

To further evaluate the proposed PKDE method, we hired volunteers to artificially generate 16 density maps and point frames. We developed a painting tool. First, the volunteers paint the density map on the canvas by themselves. Second, a point set is sampled on the density map using the Perlin noise method [41], which is a well-known sampling method. The generated 16 point sets and 16 density maps are the benchmarks for the evaluation. In our evaluation, three strategies are applied on the 16 point sets. Estimated density maps are compared with the density maps in the benchmarks. Fig. 23(a) shows a generated colorful density map. Fig. 23(b-d) shows an example of the comparison results of the different density estimation strategies. We adopt SSIM [54] method to compare the similarity between the benchmark and the density estimation result. Table 1 shows that the PKDE result is most similar to the benchmark.

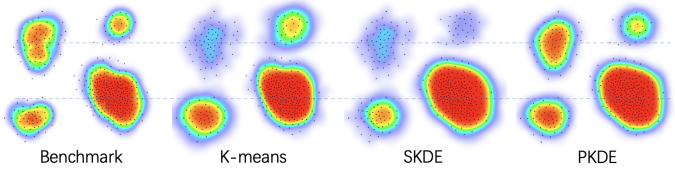


Fig. 23. An example of the result comparisons of different density estimation strategies.

9 Conclusion

We outline a new framework for the real-time visual query of spatiotemporal data. A temporal extraction of density features is addressed to aggregate the spatiotemporal data into density maps. Based on the density maps, we introduce a novel dictionary learning approach to compress the density map and accelerate the data processing. We provide a pair of query approaches and the representations to visualize and explore the query results. Our experiments show that this system can support the real-time querying of sparse spatiotemporal data.

Our system still has some limitations. First, our system does not perform as well as traditional methods when processing queries with a very small dataset. Second, using a single display to query the spatiotemporal data has weaknesses since the 2D map for interaction occupies most of the display thus the space for visualization is limited. A hybrid interaction, as shown in the work of Kister et al. [21], seems to be a solution to visualize the data through the combination of a large display and a mobile device. Third, the current version of DDLVis also has limitations in addressing different levels of detail of the spatiotemporal data because of density maps at different levels of detail. It is possible to use a mixed visualization strategy using DDLVis for high-level query, and traditional scatter plots or density maps for low-level data exploration.

Sketch-based query interaction [8] is a direction to improve the performance of DDLVis via a two-dimensional sketch query. Also, an approach based on CNNs [22] has the potential to be used to further improve the compression ratio of the density maps. The method based on spectral representation [60] can further improve the effectiveness of sparse representation, so it may be a potential direction. Because the purpose of feature maps of CNNs is mainly to obtain image features. CNNs needs further improvement before it can be used for density map compression. We plan to conduct the experimental verification for the auto-encoders [43] approach to present the context of spatiotemporal data in the future. Furthermore, we plan to design visual query systems for line-based and region-based spatiotemporal data.

Acknowledgments

The authors wish to acknowledge the support from NSFC under Grants (No. 61802128 and 62072183).

REFERENCES

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[2] B. Bach, C. Shi, N. Heulot, T. Madhyastha, T. Grabowski, and P. 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.

[3] A. Bookstein, V. A. Kulyukin, and T. Raita. Generalized Hamming Distance. *Information Retrieval*, 5(4):353–375, 2002.

[4] M. Bostock, V. Ogievetsky, and J. Heer. D3: Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309, 2011.

[5] M. Brehmer, B. Lee, B. Bach, N. H. Riche, and T. Munzner. Timelines Revisited: A Design Space and Considerations for Expressive Storytelling. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):449–458, 2016.

[6] O. Bryt and M. Elad. Compression of Facial Images Using the K-SVD Algorithm. *Journal of Visual Communication and Image Representation*, 19(4):270–282, 2008.

[7] A. Coates and A. Y. Ng. Learning Feature Representations with K-means. pp. 561–580, 2012.

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

[9] M. Derthick, J. Kolojejchick, and S. F. Roth. An Interactive Visual Query Environment for Exploring Data. In *Proceedings of 10th Annual ACM Symposium on User Interface Software and Technology*, pp. 189–198. ACM, 1997.

[10] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[11] M. Elad, M. A. Figueiredo, and Y. Ma. On the Role of Sparse and Redundant Representations in Image Processing. *Proceedings of the IEEE*, 98(6):972–982, 2010.

[12] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD*, vol. 96, pp. 226–231, 1996.

[13] N. Ferreira, J. Poco, H. T. Vo, J. Freire, and C. 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.

[14] T. Guo, K. Feng, G. Cong, and Z. Bao. Efficient Selection of Geospatial Data on Maps for Interactive and Visualized Exploration. In *Proceedings of the 2018 International Conference on Management of Data*, pp. 567–582. ACM, 2018.

[15] C. Hurter, O. Ersoy, and A. Telea. Graph Bundling by Kernel Density Estimation. 31(3pt1):865–874, 2012.

[16] S. Jänicke, C. Heine, and G. Scheuermann. Geotemco: Comparative visualization of geospatial-temporal data with clutter removal based on dynamic delaunay triangulations. In G. Csurka, M. Kraus, R. S. Laramee, P. Richard, and J. Braz, eds., *Computer Vision, Imaging and Computer Graphics. Theory and Application*, pp. 160–175. Springer, 2013.

[17] J. Johansson, P. Ljung, M. Jern, and M. Cooper. Revealing Structure within Clustered Parallel Coordinates Displays. In *Proceedings of IEEE Symposium on Information Visualization*, pp. 125–132, October 2005.

[18] F. Khelifi and J. Jiang. Perceptual Image Hashing based on Virtual Watermark Detection. *IEEE Transactions on Image Processing*, 19(4):981–994, 2010.

[19] H.-R. Kim, M.-J. Yoo, H. Kang, and I.-K. Lee. Perceptually-based Color Assignment. In *Computer Graphics Forum*, vol. 33, pp. 309–318, 2014.

[20] S. Kim, S. Jeong, I. Woo, Y. Jang, R. Maciejewski, and D. S. Ebert. Data Flow Analysis and Visualization for Spatiotemporal Statistical Data without Trajectory Information. *IEEE Transactions on Visualization and Computer Graphics*, 24(3):1287–1300, 2018.

[21] U. Kister, K. Klamka, C. Tominski, and R. Dachselt. GraSp: Combining Spatially-Aware Mobile Devices and a Display Wall for Graph Visualization and Interaction. 2017.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, pp. 1097–1105, 2012.

[23] M. Krstajic, E. Bertini, and D. Keim. Cloudlines: Compact Display of Event Episodes in Multiple Time-series. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2432–2439, 2011.

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

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

[26] C. Li, G. Baciu, and Y. Han. StreamMap: Smooth Dynamic Visualization of High-Density Streaming Points. *IEEE Transactions on Visualization and Computer Graphics*, 24(3):1381–1393, 2018.

[27] L. Lins, J. T. Klosowski, and C. Scheidegger. Nanocubes for Real-time Exploration of Spatiotemporal Datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2456–2465, 2013.

[28] Z. Liu, B. Jiang, and J. Heer. imMens: Real-time Visual Querying of Big Data. *Computer Graphics Forum*, 32(3):421–430, 2013.

[29] Y. Lu, M. Steptoe, S. Burke, H. Wang, J.-Y. Tsai, H. Davulcu, D. Montgomery, S. R. Corman, and R. Maciejewski. Exploring evolving media discourse through event cueing. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):220–229, 2016.

[30] J. Lukasczyk, R. Maciejewski, C. Garth, and H. Hagen. Understanding hotspots: A topological visual analytics approach. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, p. 36. ACM, 2015.

[31] R. Maciejewski, S. Rudolph, R. Hafen, A. Abusalah, M. Yakout, M. Ouzzani, W. S. Cleveland, S. J. Grannis, and D. S. Ebert. A visual analytics approach to understanding spatiotemporal hotspots. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):205–220, 2010.

[32] J. 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*, pp. 281–297. University of California Press, Berkeley, Calif., 1967.

[33] X. Map. http://xingyun.map.qq.com. 2013.

[34] M. W. Marcellin, M. J. Gormish, A. Bilgin, and M. P. Boliek. An overview of jpeg-2000. In *Proceedings of the Data Compression Conference*, pp. 523–541. IEEE, 2000.

[35] A. Mayorga and M. Gleicher. Splatterplots: Overcoming Overdraw in Scatter plots. *IEEE Transactions on Visualization and Computer Graphics*, 19(9):1526–38, Sept. 2013.

[36] H. Mei, W. Chen, Y. Wei, Y. Hu, S. Zhou, B. Lin, Y. Zhao, and J. Xia. RSATree: Distribution-Aware Data Representation of Large-Scale Tabular Datasets for Flexible Visual Query. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):1161–1171, 2020.

[37] F. Miranda, L. Lins, J. T. Klosowski, and C. T. Silva. Topkube: A Rank-aware Data Cube for Real-time Exploration of Spatiotemporal Data. *IEEE Transactions on Visualization and Computer Graphics*, 24(3):1394–1407, 2018.

[38] C. A. Pahins, S. A. Stephens, C. Scheidegger, and J. L. Comba. Hashedcubes: Simple, Low Memory, Real-time Visual Exploration of Big Data. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):671–680, 2017.

[39] C. A. L. Pahins, N. Ferreira, and J. L. Comba. Real-time exploration of large spatiotemporal datasets based on order statistics. *IEEE Transactions on Visualization and Computer Graphics*, 26(11):3314–3326, 2020.

[40] V. Papyan, Y. Romano, and M. Elad. Convolutional Neural Networks Analyzed via Convolutional Sparse Coding. *The Journal of Machine Learning Research*, 18(1):2887–2938, 2017.

[41] K. Perlin. Improving noise. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pp. 681–682, 2002.

[42] A. Perrot, R. Bourqui, N. Hanusse, F. Lalanne, and D. Auber. Large interactive visualization of density functions on big data infrastructure. In *2015 IEEE 5th Symposium on large Data Analysis and Visualization (lDAV)*, pp. 99–106. IEEE, 2015.

[43] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin. Variational autoencoder for deep learning of images, labels and captions. In *Advances in neural information processing systems*, pp. 2352–2360, 2016.

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

[45] M. Sadeghi, M. Babaie-Zadeh, and C. Jutten. Dictionary Learning for Sparse Representation: A Novel Approach. *IEEE Signal Processing Letters*, 20(12):1195–1198, 2013.

[46] R. Scheepens, N. Willems, N. Andrienko, N. Andrienko, N. Andrienko, and J. J. V. Wijk. Composite Density Maps for Multivariate Trajectories. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2518–2527, 2011.

[47] G. Shurkhovetskyy, N. Andrienko, G. Andrienko, and G. Fuchs. Data Abstraction for Visualizing Large Time Series. 37(1):125–144, 2018.

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

[49] A. Slingsby and E. Van Loon. Exploratory Visual Analysis for Animal Movement Ecology. 35(3):471–480, 2016.

[50] J. A. Tropp and A. C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on information theory*, 53(12):4655–4666, 2007.

[51] C. Turkay, A. Slingsby, H. Hauser, J. Wood, and J. Dykes. Attribute Signatures: Dynamic Visual Summaries for Analyzing Multivariate Geographical Data.

IEEE Transactions on Visualization and Computer Graphics, 20(12):2033–2042, 2014.

[52] U. von Luxburg. A Tutorial on Spectral Clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[53] G. Wang, J. Guo, M. Tang, J. F. de Queiroz Neto, C. Yau, A. Daghistani, M. Karimzadeh, W. G. Aref, and D. S. Ebert. STULL: Unbiased Online Sampling for Visual Exploration of Large Spatiotemporal Data. In *2020 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 72–83. IEEE, 2020.

[54] 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.*, 13(4):600–612, 2004.

[55] N. Willems, H. van de Wetering, and J. J. van Wijk. Visualization of Vessel Movements. In *Proceedings of the 11th Eurographics / IEEE - VGTC Conference on Visualization*, pp. 959–966, 2009.

[56] K. Wongsuphasawat and B. Shneiderman. Finding Comparable Temporal Categorical Records: A Similarity Measure with an Interactive Visualization. In *Proceedings of the IEEE Symposium on Visual Analytics Science and Technology (VAST2009)*, pp. 27–34. IEEE, 2009.

[57] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image Super-resolution via Sparse Representation. *IEEE transactions on image processing*, 19(11):2861–2873, 2010.

[58] Y. Yankelevsky and M. Elad. Structure-aware classification using supervised dictionary learning. In *Proceedings of 2017 IEEE International Conference on the Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4421–4425. IEEE, 2017.

[59] J. Zhao, X. Liu, C. Guo, Z. C. Qian, and Y. V. Chen. Phoenixmap: An Abstract Approach to Visualize 2D Spatial Distributions. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2000–2014, 2021.

[60] Y. Zhao, X. Luo, X. Lin, H. Wang, X. Kui, F. Zhou, J. Wang, Y. Chen, and W. Chen. Visual analytics for electromagnetic situation awareness in radio monitoring and management. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):590–600, 2019.