ABSTRACT
Urban data (e.g., real estate data, crime data) often have multiple attributes which are highly geography-related. With the scale of data increases, directly visualizing millions of individual data points on top of a map would overwhelm users’ perceptual and cognitive capacity and lead to high latency when users interact with the data. In this demo, we present ConvexCubes, a system that supports interactive visualization of large-scale multidimensional urban data in a multi-granularity way. Comparing to state-of-the-art visualization-driven data structures, it exploits real-world geographic semantics (e.g., country, state, city) rather than using grid-based aggregation. Instead of calculating everything on demand, ConvexCubes utilizes existing visualization results to efficiently support different kinds of user interactions, such as zooming & panning, filtering and granularity control. Our system can be accessed at http://115.146.89.158/ConvexCubes/.

1 INTRODUCTION
In recent years, both the volume and the availability of urban data related to various social issues, such as real estate, crime, population, etc., are increasing rapidly. Visually analyzing such data is of practical importance: (i) the policy makers can make better informed decisions; and (ii) the residents can benefit from understanding the urban life in different areas on many occasions, for example, when selecting a property to live in.

In our previous work [4], we have designed and developed a visualization system for Australia’s real estate data. In the property-level visualization, we displayed each property on top of a geographic map, where users can explore the data and compare properties in multiple dimensions. However, as the scale of data increases (e.g., over one million data points), existing information visualization methods, including directly visualizing each data point on top of a map, suffer from the following drawbacks: (i) data over-plotting, (ii) overwhelming users’ perceptual and cognitive capacities[perceptual scalability problem [6]], and (iii) difficulties to support interactive exploration, since users’ interaction with large-scale datasets would easily lead to high latency (interactive scalability problem [6]). It motivates us to investigate this problem.

From our literature study, we find two major types of methods that have been proposed to address the scalability issue of visualizing large-scale datasets: (i) visualization techniques, e.g., progressive visual analytics [9], predictive visual analytics, pixel-based visualization, spatial displacement, parallel data processing and rendering; and (ii) data reduction methods [6], such as filtering, sampling, and binned aggregation.

However, visualization techniques still need to scan each data item, which makes them difficult to scale. For data reduction methods, since filtering and sampling techniques fail to provide an overview of a dataset [3], one recent research trend in big data visualization is to intersect data management and visualization based on binned aggregation, and several studies [5–7, 10] have applied the idea of data cubes into visualization. They pre-compute possible data aggregations to support efficient visualization.

While their contributions are significant to the field, there exists a gap with real-life applications in terms of the following three drawbacks. First, existing methods (e.g., [5, 7, 10]) often partition geographic dimensions based on a grid-based hierarchical spatial data structure, such as Quadtree [2]. Such design fails to capture geographic semantics (e.g., country, state, city, etc.), thus cannot answer many real-life questions, such as “what is the difference of house prices in different neighbourhoods?”, or “how many tweets have been posted for an event in different countries or states?” Second, in some studies [5, 7, 10], numerical dimensions are transformed to categorical dimensions based on pre-defined binned aggregation, which results in the case that data with close-by values possibly fall into different bins. Third, most existing data structures are designed to support visual encoding of some specific visualization methods (e.g., heatmaps); however, they fail to support interactive visualization because all intermediate results have to be recomputed from scratch whenever the user conducts a further interaction.

To overcome the above drawbacks, we propose ConvexCubes to support the visualization of large-scale multidimensional urban data in multiple levels of granularity. The details of how we address the three problems with ConvexCubes can be found at Section 3.1, 3.3 and 4.2, respectively. The implemented demo system1 and a demonstration video2 are available online.

2 VIS ABSTRACTION & SYSTEM OVERVIEW
In this section, we first discuss the data and visualization tasks, followed by our visual encoding and interaction designs. Finally, we present an overview of the system.

1Demo system: http://115.146.89.158/ConvexCubes/
2Demonstration video: http://115.146.89.158/ConvexCubes/video.html
Multidimensional view: line chart + histograms. We design a linked multidimensional view to visualize the statistical information of a highlighted cluster. The view compares local information with the global statistics (e.g., comparing the house prices in a suburb with the house prices in the entire state). As shown in Figure 3(c), for each selected dimension, we draw a vertical line chart to show the global distribution of data, and on top of it, we draw histograms to visualize the data distribution in the highlighted cluster.

Based on our task abstraction and Shneiderman’s visual information-seeking mantra, the visualization needs to support the following interactions: (1) zooming & panning, (2) filtering, (3) granularity control, (4) other interactions, such as highlighting & linking and changing focused measures.

2.3 System Overview

Figure 1 shows an overview of our system. We construct our proposed data structure (ConvexCubes) from the raw data files, and store them as JSON files; the statistical information for each cluster and individual data items are stored in a database (Section 3). When a user initializes the visualization from a browser, the JSON files are loaded from the server, and a visualized overview is generated for the user (Section 4.1). Different layers of ConvexCubes are affected by different kinds of user interactions (shown with an arrow from each interaction to a layer of the ConvexCubes). After each interaction, only the part of the current visualization that is different from the previous visualization will be refreshed (Section 4.2).

3 DATA MANAGEMENT AND INDEXING

In this section, we describe how we store and index the data, and build the ConvexCubes. The construction of ConvexCubes requires a multi-criteria sorting of the dataset, which is similar to Hashedcubes [7]. As shown in Figure 2, the construction of ConvexCubes starts from geo-semantic layers (Section 3.1); then categorical dimensions (Section 3.2), followed by granularity clustering of numerical dimensions (Section 3.3). Finally, we calculate a convex hull for each cluster and store the statistical information of each cluster in the database (Section 3.4).

3.1 Geographic Semantic Layers

The geographical layer of most state-of-the-art data cubes [5, 7, 10] are based on grid-based hierarchical spatial data structures, such as Quadtree [2] which recursively divides the space into four regions. Such data structures are effective for querying on the geographical dimension, but might not preserve the geographical semantics. For example, if we partition the data based on the geo-locations of the real estate properties using Quadtree, in one of the middle layers, two properties from two different states may belong to the same grid, while they have no semantic relationship. Therefore, we design the geographical level of ConvexCubes based on real-world geographical hierarchy, such as country, state, city, etc.

For the real estate data, we apply a multi-criteria sorting based on different levels of geographic semantics. As shown in Figure 2, all properties in Victoria will be first grouped together in Level 1, then all properties in Melbourne CBD and Richmond will be grouped together, respectively. We use a 2-/3-digit number to represent each geographic semantic level.
We calculate a convex hull for each cluster in ConvexCubes based on geo-spatial information to support our map-based visual encoding design (Map-based view 2). Instead of storing the locations for all data points, we store the convex hull (bounding polygon) of the points. If multiple clusters are required to be combined, we apply an approximate algorithm to combine the convex hulls instead of re-calculating the convex hull for the combined set of points.

We also calculate statistical information for each cluster and store them in the database. For each cluster, we store a binned statistical result based on each dimension measure to support the multidimensional view. Such information is stored as views in the database, so that when a cluster is highlighted, the information of the selected dimension measures will be loaded and visualized.

4 DEMONSTRATION

In this section, we illustrate how we use ConvexCubes to support our visualization and user interactions. Each method is followed by a demonstration scenario based on the real estate dataset. While the dataset in our previous work [4] has 50k properties, we include 1.42 million properties in Victoria and New South Wales, Australia. A detailed description of the dataset and the schema, along with demonstration scenarios on the other datasets (e.g., social network check-in datasets) can also be found in the online demo system.

4.1 Initializing the Visualization

The initialization of visualization requires four parameters, a default map window (i.e., maximum and minimum latitudes and longitudes), user-selected categorical dimensions (e.g., 3-bedroom houses), a default clustering granularity (e.g., 400m), and a default focused measure (e.g., price). Based on the initialization settings, the query starts from the root node of ConvexCubes and the following steps are executed: (i) the nodes in geographic semantic layers that are within the map window are accessed, (ii) the categorical dimensions are scanned, and (iii) the clusters are obtained when the right level of the clustering granularity is reached. In Fig. 2, only the highlighted clusters in level 5 will be visualized for the default initialization parameter values. Then based on the stored convex hull and the default focused measure from the database, a dot map or the corresponding polygons on the map are shown to the user.

Demonstration scenario 1: When the user opens the webpage, a map will be initialized with a default map window and clustering granularity. After the user selects the categorical dimension (Figure 3 (a)-1), either of the two map views will be shown to the user (Figure 3 (a) or (b)) based on the stored information. For example, in Figure 3 (b), the user can observe how the prices of 3-bedroom houses are distributed in the geographical space.

4.2 Supporting User Interactions

We illustrate how we use ConvexCubes to efficiently support different types of user interactions. The key idea is to reuse existing calculations and visualizations while updating for user interactions.

4.2.1 Zooming and panning: These are the basic operations on top of a map, which directly influences the map window. The change of the map window corresponds to the level of geographic semantics in ConvexCubes. Therefore, after the user applies zooming in/out or panning, we calculate a set subtraction between the current and previous map window, and then refresh the visualization based on the subtraction result. In contrast to a quadtree node, the boundary of a semantic geographical node often contains a lot of geo-points. Therefore, we also store the minimum bounding rectangle for each geo-boundary. If the map window covers the entire rectangle, we include all the clusters in the lower level; otherwise, the next geographic semantic level is accessed.

Figure 2: An example of ConvexCubes with 6 layers: 2 geo-semantic layers (Level 1 & 2), 2 categorical dimension layers (Level 3 & 4) and 2 numerical dimension layers (Level 5 & 6).

3.2 Categorical Dimensions

Categorical dimensions are often divided based on specific values or ranges. For example, Property type is a categorical dimension which has values such as "house" and "apartment". In ConvexCubes, we use a 1-/2-digit number to represent each categorical dimension. As shown in Figure 2, two categorical dimensions (property type and bedroom number) are considered (Level 3 & 4).

3.3 Numerical Dimensions and Granularity Clustering

Existing data cubes [5, 7, 10] often transform numerical dimensions into categorical dimensions by partitioning data values into data segments. A drawback of such design is that data with close-by values may fall into different data segments. To solve the problem, we apply a granularity clustering for numerical dimensions. In our implementation, geo-spatial dimensions (latitude and longitude) are a special case of numerical dimensions. As shown in Figure 2, in Level 5, we apply DBScan [1] with a threshold of 400m to cluster the properties based on Euclidean distance. Then in Level 6, we re-apply DBScan with a smaller threshold (100m in our demonstration) to possibly divide some of the original clusters into multiple smaller clusters. Such structure allows us to visualize the clusters in a multi-granularity way, and provides better support for granularity controls in an interactive way. For each cluster, we store the pivot indices which indicate the start index number and the end index number of the sorted properties.

3.4 Convex Hull and Aggregation Functions

We calculate a convex hull for each cluster in ConvexCubes based on geo-spatial distance. If multiple clusters are required to be combined, we apply an approximate algorithm to combine the convex hulls instead of recalculating the convex hull for the combined set of points.

4 DEMONSTRATION

In this section, we illustrate how we use ConvexCubes to support our visualization and user interactions. Each method is followed by a demonstration scenario based on the real estate dataset. While the dataset in our previous work [4] has 50k properties, we include 1.42 million properties in Victoria and New South Wales, Australia. A detailed description of the dataset and the schema, along with demonstration scenarios on the other datasets (e.g., social network check-in datasets) can also be found in the online demo system.

4.1 Initializing the Visualization

The initialization of visualization requires four parameters, a default map window (i.e., maximum and minimum latitudes and longitudes), user-selected categorical dimensions (e.g., 3-bedroom houses), a default clustering granularity (e.g., 400m), and a default focused measure (e.g., price). Based on the initialization settings, the query starts from the root node of ConvexCubes and the following steps are executed: (i) the nodes in geographic semantic layers that are within the map window are accessed, (ii) the categorical dimensions are scanned, and (iii) the clusters are obtained when the right level of the clustering granularity is reached. In Fig. 2, only the highlighted clusters in level 5 will be visualized for the default initialization parameter values. Then based on the stored convex hull and the default focused measure from the database, a dot map or the corresponding polygons on the map are shown to the user.

Demonstration scenario 1: When the user opens the webpage, a map will be initialized with a default map window and clustering granularity. After the user selects the categorical dimension (Figure 3 (a)-1), either of the two map views will be shown to the user (Figure 3 (a) or (b)) based on the stored information. For example, in Figure 3 (b), the user can observe how the prices of 3-bedroom houses are distributed in the geographical space.

4.2 Supporting User Interactions

We illustrate how we use ConvexCubes to efficiently support different types of user interactions. The key idea is to reuse existing calculations and visualizations while updating for user interactions.

4.2.1 Zooming and panning: These are the basic operations on top of a map, which directly influence the map window. The change of the map window corresponds to the level of geographic semantics in ConvexCubes. Therefore, after the user applies zooming in/out or panning, we calculate a set subtraction between the current and previous map window, and then refresh the visualization based on the subtraction result. In contrast to a quadtree node, the boundary of a semantic geographical node often contains a lot of geo-points. Therefore, we also store the minimum bounding rectangle for each geo-boundary. If the map window covers the entire rectangle, we include all the clusters in the lower level; otherwise, the next geographic semantic level is accessed.
Demonstration scenario 2: The user can zoom in or zoom out, or pan the map just as they do on Google Maps. We can illustrate how ConvexCubes re-use existing visualization results and partially refresh the interface by asking the audiences to pay attention to the borders of the map region. An example is shown in Figure 3(a).

4.2.2 Filtering. Filtering, i.e., selections, would affect the level of categorical dimensions. For example, if previously, only 3-bedroom units in Melbourne are shown on the map, and the user selects to include 3-bedroom houses as well, then the corresponding clusters in Level 5 will be added to the visualization result while the previous result remains. Note that, having multiple selections might result in some intersecting convex hulls. To effectively detect intersections at runtime, we store the minimum bounding rectangle for each cluster along with the convex hull information, and we compute the union of the intersected convex hulls at runtime.

Demonstration scenario 3: The user can filter on top of the pre-selected dimensions and see how properties/clusters change on top of the map. As shown in Figure 3(b)-3, we change the filtering setting from 3-bedroom houses to 2-/3-bedroom units in Melbourne.

4.2.3 Granularity control. This will affect the level of numerical dimensions. For example, in Figure 2, if the user changes the current granularity level from Level 5 to Level 6, clusters that are not split will stay on the map, while clusters falling into multiple smaller clusters (i.e., the first and the sixth node at Level 5) will be replaced by their children clusters. To make the interaction easier, we separate zooming and granularity controls as two different operations. In our implementation, user zooming in or out will trigger the function of granularity control; users can also change to different granularities while staying at the same zooming level.

Demonstration scenario 4: The user can change the granularity level by zooming in or zooming out, or they can directly select different levels from the selection panel (Figure 3(a)-1). As shown in Figure 3(b)-2, the user can change to a finer granularity and see how the clusters in (b)-1 split into multiple smaller clusters. For example, the user observing that properties in Brighton (which is closer to the coastline) is more expensive than nearby suburbs, might want to change it to a finer granularity and check whether property prices in different blocks of Brighton are also affected by the distance to the coastline.

4.2.4 Other interactions. Highlighting & linking need to access the database. A row in the cluster table needs to be accessed when the user selects a cluster to highlight; while highlighting an individual property requires a row in the property table. For interactions such as changing focused measures, a column in the cluster table will be accessed.

Demonstration scenario 5: The user can click to highlight a cluster, and a linked multidimensional view is generated for users to (i) check the detailed statistics of the cluster in several user-defined measures and (ii) compare properties in the cluster with properties at a higher geo-semantic level. As shown in Figure 3(c), the user highlights a cluster in Brighton, Victoria; simultaneously, in the multidimensional view, two sets of bar charts + histograms are highlighted to present a comparison of properties in the highlighted cluster and those in the entire Victoria based on two user-defined measures (the price and the distance to the nearest supermarket).

Acknowledgements. This work was partially supported by ARC DP170102726, DP170102231, DP180102050, and National Natural Science Foundation of China (NSFC) 61728204, 91646204. Zhifeng Bao is supported by a Google Faculty Award.

REFERENCES