



VISUAL ANALYTICS OF GEO-RELATED MULTIDIMENSIONAL DATA

A thesis submitted in fulfilment of the requirements for
the degree of Doctor of Philosophy

MINGZHAO LI

M.Eng. (Computer Technology), Sichuan University, China

B.Eng. (Software Engineering), Sichuan University, China

School of Science

College of Science, Engineering, and Health

RMIT University

Melbourne, Victoria, Australia

March 2019

Declaration

I certify that except where due acknowledgement has been made, the work is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program; any editorial work, paid or unpaid, carried out by a third party is acknowledged; and, ethics procedures and guidelines have been followed.

Mingzhao Li

School of Science (Computer Science)

RMIT University

6 Mar 2019

Acknowledgement

I wish to express my deepest respect and gratitude to my supervisors Dr. Zhifeng Bao and Prof. Timos Sellis. I would not have finished my Ph.D. without their tremendous support. I truly appreciate Zhifeng's supervision in every detail of my research, and Timos' guidance from a high level has made the journey easier. I have learnt a lot from my supervisors regarding how to do research and how to live a life.

I would also like to thank those who have helped with my Ph.D. research. I feel honoured and grateful to have collaborated with Dr. Farhana Chaudhry and Prof. Hanan Samet for my second and third research questions. Special thanks go to Shi Yan, Binbin Tang and Sing Fai Law who have assisted the experiments of my research. I also appreciate the help from the following friends and colleagues who have given useful suggestions to our work: Prof. Antoon Moorman, Prof. Matt Duckham, Prof. Mark Sanderson, Paresh Kevat, Darren McCoy.

A special thanks to my master supervisor, Prof. Min Zhu from Sichuan University, who has not only created collaboration opportunities throughout my Ph.D. period but also shown her endless care as a teacher and a friend.

I am grateful to have met those lovely and inspiring friends during my Ph.D.: Mohammad Haqqani the friendship with whom has shaped my mind of both research and life; Hui Luo and Peishan Li who have always made me laugh and feel loved; Jaimie Jin, Shi Yan, Yassien Shaalan, Bilih Priyogi, Chengzhe Yuan, Iman Amini, Liangjun Song, Vasilis Efthymiou, John Ryan, I-Chen Huang, Pei-Hsuan Kuan, Paul Kusmanoff, Ahmed Mourad and Kevin Ong; I also wish to thank my beloved friends: Qing Ding, Ting Liang, Sisi Wu, Yue Zhao, Binbin Tang, Fengwei Zhang, Qingyue Cui, Xiaoqing Hu and Ang Zeng who have always believed me and been there for me when needed.

This Ph.D. is dedicated to my parents who have raised me up with all their love; my partner, my sister and my nephew who have made my life much more enjoyable.

Abstract

In recent years, both the volume and the availability of urban data related to various social issues, such as real estate, crime and population are rapidly increasing. Analysing such urban data can help the government make evidence-based decisions leading to better-informed policies; the citizens can also benefit in many scenarios such as home-seeking. However, the analytic design process can be challenging since (i) the urban data often has multiple attributes (e.g., the distance to supermarket, the distance to work, schools zone in real estate data) that are highly related to geography; and (ii) users might have various analysis/exploration tasks that are hard to define (e.g., different home-buyers might have requirements for housing properties and many of them might not know what they want before they understand the local real estate market). In this thesis, we use visual analytics techniques to study such geo-related multidimensional urban data and answer the following research questions.

In the first research question, we propose a visual analytics framework/system for geo-related multidimensional data. Since visual analytics and visualization designs are highly domain-specific, we use the real estate domain as an example to study the problem. Specifically, we first propose a problem abstraction to satisfy the requirements from users (e.g., home buyers, investors). Second, we collect, integrate and clean the last ten year's real estate sold records in Australia as well as their location-related education, facility and transportation profiles, to generate a real multi-dimensional data repository. Third, we propose an interactive visual analytic procedure to help less informed users gradually learn about the local real estate market, upon which users exploit this learned knowledge to specify their personalized requirements in property seeking. Fourth, we propose a series of designs to visualize properties/suburbs in different dimensions and different granularity. Finally, we implement a system prototype for public access (<http://115.146.89.158>), and present case studies based on real-world datasets and real scenario to demonstrate the usefulness and effectiveness of our system.

Our second research question extends the first one and studies the scalability problem

to support cluster-based visualization for large-scale geo-related multidimensional data. Particularly, we first propose a design space for cluster-based geographic visualization. To calculate the geographic boundary of each cluster, we propose a concave hull algorithm which can avoid complex shapes, large empty area inside the boundary and overlaps among different clusters. Supported by the concave hull algorithm, we design a cluster-based data structure named ConcaveCubes to efficiently support interactive response to users' visual exploration on large-scale geo-related multidimensional data. Finally, we build a demo system (<http://115.146.89.158/ConcaveCubes>) to demonstrate the cluster-based geographic visualization, and present extensive experiments using real-world datasets and compare ConcaveCubes with state-of-the-art cube-based structures to verify the efficiency and effectiveness of ConcaveCubes.

The last research question studies the problem related to visual analytics of urban areas of interest (AOIs), where we visualize geographic points that satisfy the user query as a limited number of regions (AOIs) instead of a large number of individual points (POIs). After proposing a design space for AOI visualization, we design a parameter-free footprint method named AOI-shapes to effectively capture the region of an AOI based on POIs that satisfy the user query and those that do not. We also propose two incremental methods which generate the AOI-shapes by reusing previous calculations as per users' update of their AOI query. Finally, we implement an online demo (<http://www.aoishapes.com>) and conduct extensive experiments to demonstrate the efficiency and effectiveness of the proposed AOI-shapes.

Contents

Declaration	iii
Acknowledgement	v
Abstract	vii
Contents	ix
List of Figures	xv
List of Tables	xix
1 Introduction	1
1.1 Background	1
1.2 Motivation and Research Questions	3
1.2.1 RQ1. Visual analytics of real estate data	5
1.2.2 RQ2. Supporting large-scale geographic visualization	6
1.2.3 RQ3. Interactive visualization of urban areas of interest	8
1.3 Contributions	10
1.3.1 HomeSeeker: a visual analytics system of real estate data	12
1.3.2 ConcaveCubes: a cluster-based data cube supporting large-scale geographic data visualization	12
1.3.3 AOI-shapes: a parameter-free footprint method supporting visualization of AOIs	13
1.4 Research Outcomes	13
1.4.1 Publications	13
1.4.2 Data and systems	15

1.5	Thesis Structure	15
2	Literature Review	17
2.1	Data Visualization and Visual Analytics	17
2.2	Visualization Methods for Geo-related Multidimensional Data	21
2.2.1	Multidimensional visualization methods	21
2.2.2	Geographic visualization methods	23
2.2.3	Composite visualization design	26
2.3	Supporting Large-scale Data Visualization	26
2.3.1	Visualization reduction methods	27
2.3.2	Data reduction methods	28
2.4	Summary	29
3	A Visual Analytics System of Real Estate Data	31
3.1	Introduction	32
3.2	Related Work	34
3.3	Problem Abstraction	35
3.3.1	Domain situation	35
3.3.2	Data collection and integration	37
3.3.3	Data and task abstraction	39
3.3.3.1	Data characteristic analysis	39
3.3.3.2	Task abstraction	39
3.4	System Overview	40
3.4.1	Design maxims	40
3.4.2	System pipeline	42
3.5	Visualization Design and Justification	43
3.5.1	Profile-based region visualization (T.1)	43
3.5.2	Suburb-level visualization (T.2)	44
3.5.2.1	The Google maps view (T.2.1)	44
3.5.2.2	The historical view (T.2.2)	46
3.5.2.3	The multidimensional view (T.2.3)	47
3.5.3	Property-level visualization (T.3 & T.4)	48
3.5.3.1	The Google maps view (T.3.1)	48

3.5.3.2	The multidimensional view (T.3.2)	50
3.5.3.3	The image card view (T.3.3) & the word cloud view (T.3.4) . .	51
3.5.3.4	The spider chart view (T.4)	52
3.6	System Implementation	52
3.7	Experiments and Discussion	53
3.7.1	Dataset description	53
3.7.2	Case studies	53
3.7.2.1	Case I. From school zone to discover properties	53
3.7.2.2	Case II. From understanding regions to discover properties . .	55
3.7.2.3	Case III. Investment on properties	56
3.7.2.4	Case IV. Exploring the properties as beginners	57
3.7.3	Domain expert feedback	59
3.7.4	Discussion	60
3.8	Summary	61
4	ConcaveCubes: Supporting	
	Large-scale Geographic Visualization	63
4.1	Introduction	64
4.2	Related Work	66
4.3	Cluster-based Visualization Abstraction	67
4.3.1	Data and task abstraction	68
4.3.2	A baseline solution	68
4.3.3	Visualization design	69
4.3.4	Design justification: why cluster-based visualization?	71
4.4	Algorithm for Generating Boundary-based Cluster Maps	73
4.4.1	An outline of desired features	73
4.4.2	Our concave hull construction approach	75
4.5	ConcaveCubes: Cluster-based Data Cubes index	79
4.5.1	An example of ConcaveCubes	79
4.5.2	The construction of ConcaveCubes	80
4.5.3	Index operations	82
4.5.3.1	Initialization	82
4.5.3.2	User interactions	82
4.6	A demonstration system	83

4.6.1	Initializing the visualization	83
4.6.2	Supporting user interactions	84
4.6.2.1	Zooming and panning	84
4.6.2.2	Filtering	84
4.6.2.3	Granularity control	85
4.6.2.4	Other interactions	85
4.7	Experiments	86
4.7.1	Datasets and schemas	86
4.7.2	Visualization results	88
4.7.3	Query time	89
4.7.4	Discussion	90
4.8	Summary	92
5	AOI-shapes: Interactive Visualization of Urban Areas of Interest	93
5.1	Introduction	93
5.2	Related Work	97
5.2.1	Visualization of urban areas of interest (AOIs)	97
5.2.2	Footprint methods: generating boundaries for AOIs	98
5.2.2.1	Convex hull	98
5.2.2.2	Simple concave hull	99
5.2.2.3	Non-simple concave hull	101
5.2.2.4	A summary of existing footprint methods	103
5.3	Problem illustration	103
5.3.1	Visualization tasks	104
5.3.2	Visualization idioms	104
5.3.3	Why a novel footprint method is needed?	104
5.4	A baseline method	106
5.4.1	Preliminaries	106
5.4.1.1	Main idea of the proposed baseline method	106
5.4.1.2	Labels of vertices and edges	106
5.4.2	The proposed baseline algorithm	108
5.4.3	Time complexity of the baseline method	112
5.5	The proposed AOI-shapes	112

5.5.1	The DCEL data structure	112
5.5.2	Why DCEL structure?	114
5.5.3	Constructing the AOI-shapes	115
5.5.3.1	Step 1: finding boundary edges	115
5.5.3.2	Step 2: recognizing multiple regions	116
5.5.3.3	Step 3: detecting inner holes	117
5.5.3.4	A “digging” process	117
5.5.4	Time complexity of the AOI-shapes method	119
5.6	The incremental AOI-shapes	119
5.6.1	Incremental method I	119
5.6.2	Incremental method II	120
5.6.2.1	Removing existing points	121
5.6.2.2	Adding new points	122
5.6.3	Time complexity of the incremental algorithms	123
5.7	A demonstration system	123
5.8	Experiments	124
5.8.1	Quality of the AOI-shapes	124
5.8.1.1	Qualitative evaluation	126
5.8.1.2	Quantitative evaluation	129
5.8.2	Efficiency of the AOI-shapes	131
5.8.3	Efficiency of the incremental AOI-shapes	134
5.8.4	Discussion	134
5.9	Summary	135
6	Conclusion and Future Work	137
6.1	Conclusion	137
6.2	Future work	138
	Bibliography	145

List of Figures

1.1	An overview of the three research questions and how they are linked together. . .	4
1.2	An illustration of the map view in the existing real estate applications.	6
1.3	Illustration of the motivation of the RQ2.	7
1.4	An example of user-defined AOIs and different representations.	9
1.5	Illustration of four nested levels of vis design and the related design problem at each level.	11
1.6	An overview of our three research questions mapped to four levels of visualization design.	11
2.1	Knowledge generation model for visual analytics.	19
2.2	Examples of existing visualization methods for geographic point data.	25
3.1	Illustration of the process of data collection and integration.	37
3.2	The system pipeline of HomeSeeker.	42
3.3	Design of profile-based region visualization.	44
3.4	Design of the suburb-level visualization	45
3.5	Illustration of VIS Design 4: glyphs on maps.	46
3.6	Illustration of VIS design 7: Parallel coordinates.	47
3.7	Design of property-level Visualization.	49
3.8	Illustration of Case I: from school zone to discover properties.	54
3.9	Illustration of Case II: from understanding regions to discover properties.	55
3.10	Illustration of Case III: investment on properties.	57
3.11	Illustration of Case IV: exploring the properties as beginners.	58
4.1	An overview of our solution in Sections 4.3, 4.4 & 4.5.	68
4.2	Illustration of cluster-based geographic visualization design.	70

4.3	A comparison of four geographic visualizations based on the real estate data in Melbourne.	72
4.4	Illustration of the problem using existing concave hull methods.	74
4.5	Illustration of the proposed Concave hull algorithm.	77
4.6	An example of ConcaveCubes based on real estate data	80
4.7	Illustration of the ConcaveCubes construction procedure.	80
4.8	Demonstration examples of ConcaveCubes.	83
4.9	Visualization of the Brightkite dataset.	88
4.10	Visualization of the Australian census data.	89
4.11	Query latency per type of user interactions based on the real estate dataset. . . .	90
5.1	An example of user-defined AOIs and the comparison of different footprint methods.	94
5.2	Illustration of the proposed incremental AOI-shapes based on a global Delaunay triangulation.	96
5.3	Examples of different kinds of hulls.	99
5.4	The interface and some examples of the visualization of user-defined urban areas of interest.	105
5.5	Illustration of different types of points, edges and regions.	107
5.6	Illustration of finding boundary edges using the baseline method.	110
5.7	Illustration of Proof 5.1.	110
5.8	Illustration of detecting holes.	111
5.9	Illustration of the DCEL structure.	114
5.10	Illustration of step 1 (finding boundary edges) of the AOI-shapes algorithm. . . .	116
5.11	Illustration of the problem of the shapes generated from AOI-baseline.	117
5.12	Illustration of the “digging” process to construct the AOI-shapes.	118
5.13	Illustration of the incremental AOI-shapes algorithm.	122
5.14	Demonstration of the incremental AOI scenario.	123
5.15	A comparison of different “footprint” methods (example 1).	125
5.16	A comparison of different “footprint” methods (example 2).	126
5.17	A comparison of different “footprint” methods (example 3).	127
5.18	A comparison of different “footprint” methods (example 4).	128
5.19	A comparison of different “footprint” methods (example 5).	128
5.20	A comparison of different “footprint” methods (example 6).	129
5.21	Region-based F1-score for quantitative evaluation.	129

5.22	Efficiency-based evaluation of the AOI-shapes.	132
5.23	The construction time of the AOI-shapes.	132
5.24	Experiment result of the incremental AOI-shapes.	133
6.1	Illustration of a future research direction (data cleaning supported by visual analytics).	140

List of Tables

2.1	A summary of multidimensional visualization methods.	22
2.2	A summary of visualization methods for geographic point data.	24
3.1	A summary of our data profiles, and the comparison against existing commercial systems.	36
3.2	Statistics of the integrated real estate data.	39
3.3	Domain situation, visualization abstraction and visual encoding idioms.	41
3.4	User requirements of four use cases, corresponding tasks and visualization design choices.	54
4.1	A comparison between our proposed algorithm and 7 existing hull construction algorithms.	75
4.2	A summary of the relevant information for building ConcaveCubes.	86
5.1	A summary of footprint methods	102
5.2	Alternative methods compared in the experiment.	125

Chapter 1

Introduction

“Great things are not done by impulse, but by a series of small things brought together.”

— Vincent Van Gogh

1.1 Background

With the rapid development of information technology and communication, a large amount of geographic urban data is becoming more and more available. For example, the Australian government has released a variety of open data¹ related to various social issues, such as housing estate, population, crime, transportation and education. Analysing those data can assist in the development of evidence-based decisions leading to better-informed policies, research and social outcomes: the government to make data-driven policies and manage the city [Department of the Prime Minister and Cabinet of Australia, 2015], and the citizens can also benefit from the data analytics and further understand the urban life in different areas.

Taking home seeking as an example, finding a suitable property to purchase or rent is critical to people’s life. However, the process of it can be both complex and time-consuming. According to a report from the National Association of Realtors [2017], 95% of home buyers searched online during their home buying process, and they spent an average of more than two months doing their research before buying a property. Unfortunately, with such time and efforts spent, many real estate customers still suffer from post-purchase or post-rental regrets. The study from Trulia [2017] shows that 44 per cent of real estate consumers regret their purchase or rent decisions. Chen et al. [2011] also argue that 46 per cent of homebuyers regret their choices.

¹Australian Government Open Data Platform: <https://data.gov.au/>

Why visual analytics? One of the most critical reasons behind those statistics (the time-consuming process and the high number of purchase regret) is the lack of information and analysis provided by existing commercial real estate systems, e.g., Real Estate² and Domain³ in Australia. In those systems, users are often requested to pre-define their requirements (e.g., finding properties below one million dollars), and the real estate properties that satisfy the user query will be displayed in the form of lists or on maps; however, home buyers might not be familiar with the local real estate market and different lifestyles in different regions, and they “often do not know what they want until they see it”. Those existing platforms fail to provide effective tools to help users analyze the housing estate data and compare different regions/houses so that they could make data-driven choices. In this thesis, we use the technique of visual analytics to solve the problem, which supports exploratory data analysis (EDA) with human participation, and allows users to gradually understand the local real estate markets and local lifestyles in a visualized way and find an ideal home based on their individual requirements.

Why geo-related multidimensional data? Another reason of why home seeking is laborious is that there are many factors for homebuyers to consider when they purchase a property, such as budgets, facilities like supermarkets and nearby transportation, as well as public schools associated with children’s education. This is a typical example of geo-related multidimensional data, as (i) there are a considerable amount of attributes related to each property, and (ii) most of the attributes (e.g., associated public school rating, distance to hospital and shopping centres) are highly related to geography. The design challenge for visualizing such geo-related multidimensional data is to provide space-efficient presentations that neither clutter visually nor confuse cognitively [Lam et al., 2007]. It is a challenging task since position- and size-based visual channels are already used by maps [Beecham et al., 2016]: existing geographic visualization methods (based on maps) are limited to a very few dimensions, not to mention supporting the visualization in large data scales.

The visual analytics problem of geo-related multidimensional data has attracted researchers from different fields. First, the visualization community has focused on the visual design part, where novel visualization idioms have been proposed to represent multiple attributes in the geographic data. However, most existing works (e.g., [Guo et al., 2011, Turkay et al., 2014]) either focus on visualizing the multidimensional or the geographic features in the data, but have not linked them together. Second, the database researchers have proposed various kinds of solutions (e.g., approximate query processing [Jugel et al., 2014], visualization-based data

²<http://www.realestate.com.au/>

³<http://www.domain.com.au/>

cubes [Liu et al., 2013, Lins et al., 2013]) to address the scalability problem when visualizing large-scale geographic data; but they only support a limited number of geographic visualization designs such as heatmaps and binned plots, but not cluster maps which might have more semantic meanings. Third, researchers from the GIS (Geographic Information System) field have proposed different “footprint” methods [Zhong and Duckham, 2017, Asaeedi et al., 2017] to compute the arbitrary shape of a group of geographic points and visualize them on maps; however, both the efficiency and effectiveness could be a problem when using state-of-the-art “footprint” algorithms to support an interactive visualization system.

To address the above drawbacks, this thesis studies visual analytics techniques for geo-related multidimensional data and answers the following research questions. First, since visualization design is believed to be highly domain-specific that requires customization and fine-tuning for any new applications [Parameswaran, 2018, Munzner, 2014], in our first research question (RQ1), we use the real estate domain as an example, and propose a visual analytics framework for the geo-related multidimensional data. Specifically, we collected and integrated a comprehensive real estate dataset, designed and implemented a visual analytics system to help users understand local real estate markets, compare regions/properties in detail and find ideal homes. The second research question (RQ2) extends RQ1 to solve the scalability problem: it addresses the perceptual and interactive scalability problem when visualizing individual data points on maps; and we propose a data structure called ConcaveCubes which supports cluster-based geographic visualization in large data scale. Our last research question (RQ3) studies the problem of visualizing user-defined areas of interest (AOIs), where we visualize geographic points that satisfy the user query as a limited number of regions instead of a large number of points; we propose a parameter-free algorithm called AOI-shapes, which effectively generates region boundaries based on a group of geographic points to avoid interactive latency.

1.2 Motivation and Research Questions

Figure 1.1 presents an overview of our three research questions and illustrates how they are linked together. In this subsection, we discuss the motivation of each research question and propose our research questions.

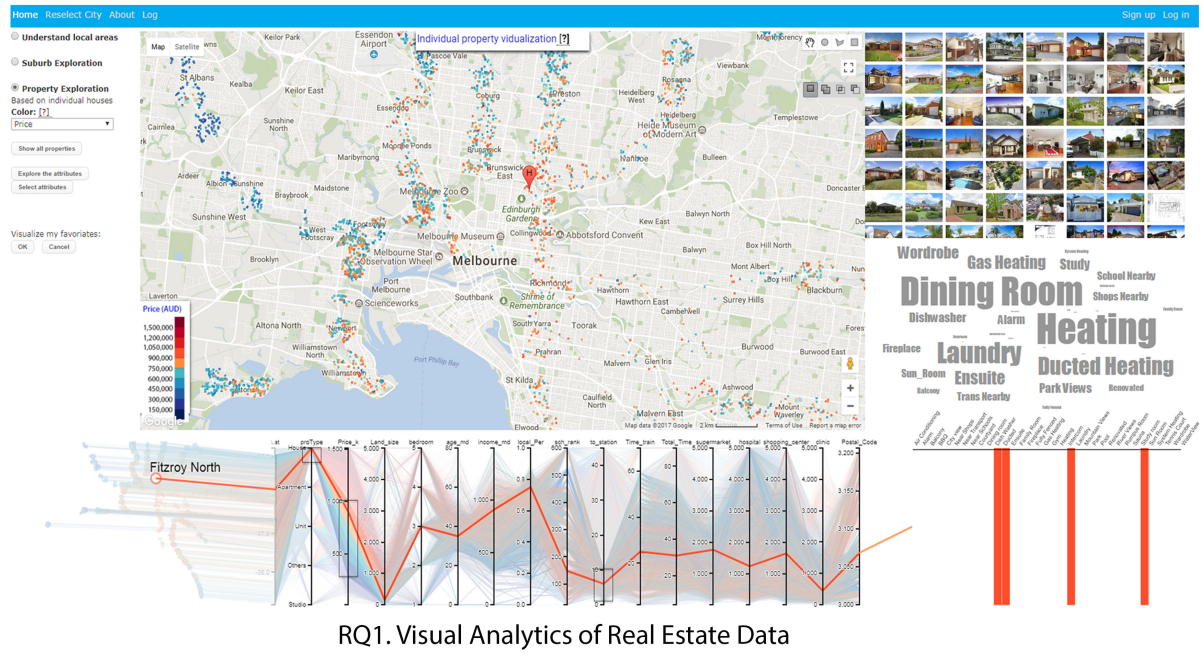


Figure 1.1: An overview of the three research questions and how they are linked together.

1.2.1 RQ1. Visual analytics of real estate data

Our first research question tries to propose a visual analytics system/framework of the geo-related multidimensional data. However, the design of visualization systems is based on visualization tasks and data [Ward et al., 2015], and visualization tasks are highly domain-specific that require customization and fine-tuning for any new application [Munzner, 2014]. Therefore, we use real estate data and the home-seeking task as an example to discuss the first research problem. While finding an ideal home has been a crucial but laborious task, we illustrate the drawbacks of existing real estate platforms (i.e., the motivation of our first research question), based on a real-life scenario when a person, *John*, tried to find a housing property to live in Melbourne.

Example 1.1. *John used Real Estate Australia to search for properties. He first searched by the suburb, South Yarra (including surrounding suburbs), and there were more than 2,000 results, displayed in the form of lists or on maps. At the list view, there were 20 properties in each page. He could sort the results by price or listing date but found it still hard to find his favoured properties in the first several pages; also, he needed to click the links one by one to check the details of each property. At the map view, John refined the search based on the price, the number of bedrooms, etc. After that, about 500 properties were left on the map (Figure 1.2); however, there was a lack of an efficient tool to compare the features of the housing properties in which he was interested (e.g., which houses are cheaper, closer to train stations and supermarkets).*

We summarize the problems when using existing commercial systems to search for real estate properties. First, data in those systems are merely about properties, while a lot of useful information that is critical to users' choices is not captured. Second, users are requested to pre-define their requirements, and then the corresponding housing properties are displayed in the form of a list or map. However, neither SQL nor Top-K based search adopted by those commercial systems manages to provide users with enough candidate properties based on their individual requirements. Third, they only show the locations of properties on the map and do not support presentation and comparison of properties in multiple aspects. Last, those commercial systems fail to provide a way for users to learn about the real estate knowledge and truly explore the real estate market data.

The first research question (RQ1) is “how do we design an effective visual analytics system for geo-related multidimensional data?”. Since the design of visualization systems are highly do-

⁴<https://www.realestate.com.au/buy/with-2-bedrooms-between-0-1500000-in-south+yarra,+vic+3141/map-1?numParkingSpaces=1&numBaths=1> (accessed at 16 Feb 2019)

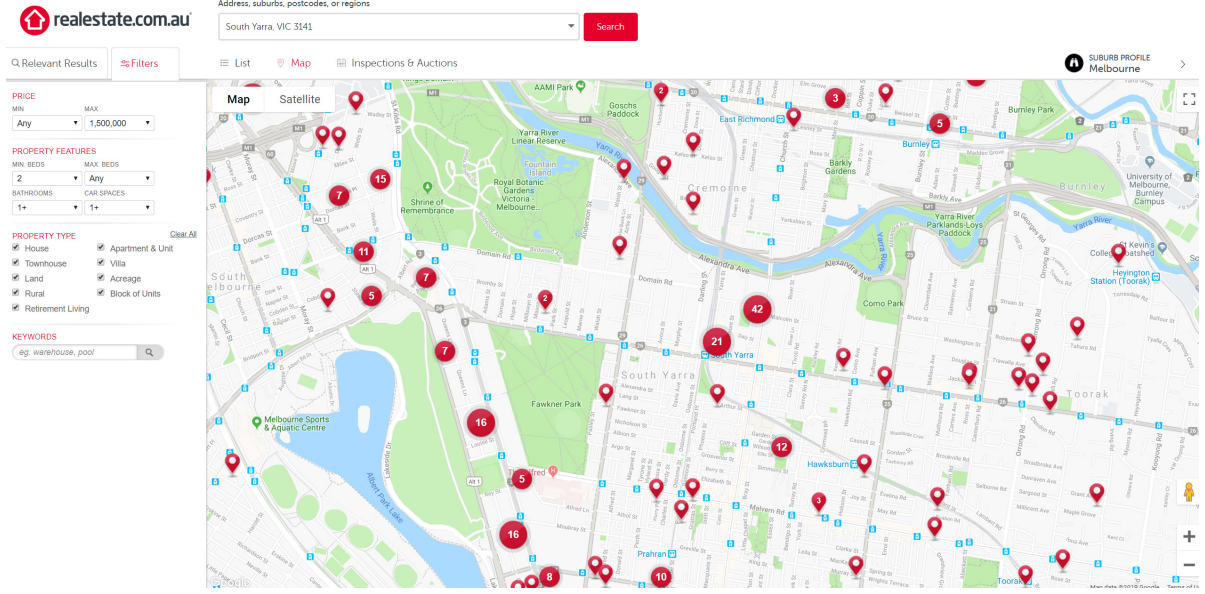


Figure 1.2: An illustration of the map view in the Real Estate Australia website⁴.

main specific, we use real estate data as an example to study this research question. Specifically, we try to answer the following questions:

- RQ 1.1 How to design task/data abstraction for geo-related multidimensional data using the real estate domain as an example?
- RQ 1.2 What kind of a visual analytics process is effective to enable users to visually analyse and explore geo-related multidimensional data?
- RQ 1.3 What kinds of visualization and interaction idioms are effective to support the visual analytics process?

1.2.2 RQ2. Supporting large-scale geographic visualization

After we propose a visual analytics framework for the geo-related multidimensional data, one main problem is how do we handle the scalability problem. Particularly, we display each real estate 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), the existing information visualization methods, including directly visualizing each data point on top of a map, suffer from the following drawbacks (as illustrated in Figure 1.3): (i) data over-plotting, (ii) overwhelming users' perceptual and cognitive capacities (*perceptual*



Figure 1.3: Illustration of the motivation of the RQ2. (a) data over-plotting; (b) perceptual scalability problem; (c) interactive scalability problem.

scalability problem [Liu et al., 2013]), and (iii) difficulties to support interactive exploration, since users’ interaction with large-scale datasets would easily lead to high latency (*interactive scalability problem* [Liu et al., 2013]). 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: (1) visualization techniques, e.g., progressive visualization [Turkay et al., 2017b], pixel-based visualization, spatial displacement, predictive visual analytics, parallel data processing and rendering are proposed; (2) data reduction methods [Liu et al., 2013], such as filtering, sampling, and binned aggregation are widely adopted to solve the scalability problem.

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 [Godfrey et al., 2016], one recent research trend in big data visualization is to intersect data management and visualization based on binned aggregation, and several studies [Liu et al., 2013, Lins et al., 2013, Wang et al., 2017b, Pahins et al., 2017] have applied the idea of data cubes into visualization. They pre-compute possible data aggregations to support efficient visualization.

While these data cube structures are effective for solving the scalability problem, we observe that they only support heatmaps [Bojko, 2009] based on binned aggregation in terms of visualizing geographical features. Although Liu et al. [2013] have shown that heatmaps have advantages vis-à-vis sampled symbol maps (e.g., heatmaps can show the overall geographical distribution while sampled symbol maps cannot), the information that heatmaps can provide is often very limited (Details in Chapter 4).

To deal with the limitation of heatmaps, we will describe a design space for geographical visualization based on the result of data clustering (rather than relational aggregation). Fol-

lowing Tobler’s First Law of Geography - *Everything is related to everything else, but near things are more related than distant things* [Tobler, 1970], we first cluster geographical data points that are close in distance (assuming Euclidean distance in contrast to other distances such as the Hausdorff distance [Nutanong et al., 2011]) and share similar features, and then represent each cluster on top of the map (e.g., cluster maps). Cluster maps, which is hard to support by existing data cubes (e.g., imMens [Liu et al., 2013], Nanocubes [Lins et al., 2013], Hashedcubes [Pahins et al., 2017]), can provide additional information compared to heatmaps or binned plots.

To conclude, our second research question (RQ2) of the thesis is: “How to use clustering visualization to address perceptual and interactive scalability of the visual analytics model for geo-related multidimensional data?”. Specifically, we try to solve the following research questions:

RQ 2.1 What kinds of visualization and interaction idioms are effective to visualize cluster-based visualization of geo-related multidimensional data?

RQ 2.2 How to present the locations of different clusters on top of the map?

RQ 2.3 What kind of data structure can support the visualization and interaction idioms in an efficient and scalable manner?

1.2.3 RQ3. Interactive visualization of urban areas of interest

One of the most important issues related to geo-related multidimensional data is to understand urban areas of interest (AOIs). An AOI refers to the area within an urban environment which attracts people’s attention [Hu et al., 2015]. Understanding urban AOIs can assist various urban data exploration tasks such as house seeking, and decision making for urban planning such as facility deployment of petrol stations, supermarkets, and general practitioners. Still taking house seeking as an example, location is a key factor, based on which various AOIs can be defined by users to cater to their respective concerns/interest. A user may want to find a region to live which is reachable from the nearest train station, supermarket and one of the top-10 public/private schools by at most 15-minute walk. The region that satisfies such a user need can be considered as a user-defined AOI.

AOIs could be represented as either polygon-based regions or individual points (e.g., real estate properties). Previous studies have shown that the polygon-based representation performs better from both cognitive and computational perspectives [Hu et al., 2015]. However, unlike

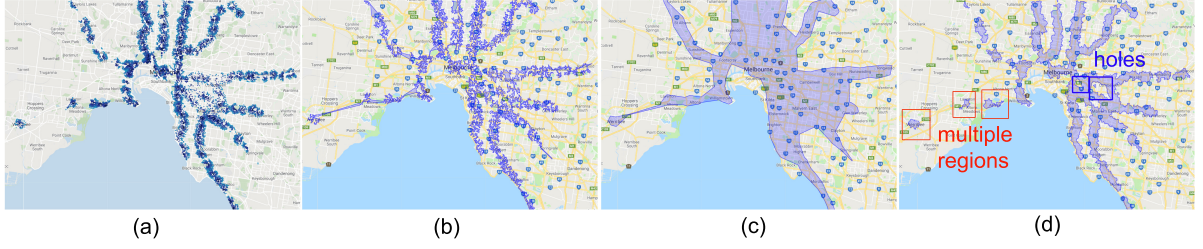


Figure 1.4: An example of user-defined AOIs (properties within 15-min walk to the nearest train station) presented as: (a) individual property points, (b) a polygon-based region generated using χ -shapes [Duckham et al., 2008] ($\chi=0.05$); (c) a polygon-based region generated using χ -shapes ($\chi=0.2$); (d) polygon-based regions generated using our proposed AOI-shapes.

most well-defined administrative districts (e.g., suburbs, cities), the region (shape) of an urban AOI could be vague [Hu et al., 2015] and often depends on users' preference.

One of the most popular methods to characterize the shape of an AOI is to generate the region boundaries, which are called “footprints” [Duckham et al., 2008, Hu et al., 2015], based on the POIs (points of interest) that satisfy the user query (e.g., in Example 5.1, the POIs are those real estate properties (Figure 5.1(a)) that satisfy the user query). Different footprint algorithms have been proposed, including concave hulls [Park and Oh, 2012a], χ -shapes [Duckham et al., 2008, Zhong and Duckham, 2017], etc. However, footprints from a definite set of points are usually not unique, i.e., the constructed footprints can be different in different algorithms and/or different parameters; and there is no consensus on the effectiveness of the footprint algorithms.

Based on a comprehensive literature review of the existing footprint methods, we find that, the state-of-the-art footprint algorithms suffer from at least one of the following drawbacks: (1) additional parameter tuning is needed from the user; (2) multiple regions and/or outliers cannot be recognized; (3) inner holes cannot be detected; and (4) incremental generation of the footprint cannot be easily supported. The following example highlights the drawbacks of the existing methods.

Example 1.2. We generate different footprints for the properties shown in Figure 5.1(a) using χ -shapes [Zhong and Duckham, 2017]. The algorithm first constructs a convex hull, and then executes a “digging” process (progressively replace an outer edge with two adjacent inner edges) to get a concave hull (i.e., footprint). The only parameter is a length-related threshold which decides whether each boundary edge is too “long” or not; and the algorithm will terminate if all boundary edges are shorter than the threshold. We use two different parameters to generate

the footprints and present both the results on Google Maps (Figure 5.1(b & c)). The results illustrate the first three drawbacks of existing footprint algorithms: (i) a smaller threshold may make the result too “concave” for users to recognize the region (Figure 5.1(b)), and a larger threshold may cause the region too “big” to characterize the shape of the region (Figure 5.1(c)); unfortunately, it is hard to find an appropriate threshold (which does not have intuitive real-life meanings) for an unknown set of data points. (ii) since the algorithm cannot recognize multiple regions, those regions (as highlighted in red in Figure 5.1(b)) are unexpectedly connected; (iii) there are regions in the middle of two train lines (as highlighted in Figure 5.1(d)) that do not satisfy the user query; however, since the χ -shapes algorithm cannot detect holes, these holes get covered in the resulted footprint.

Our last research question in this thesis is how to effectively visualize user-defined areas of interest (AOIs) based on large-scale geo-related multidimensional data. Specifically, we will answer the following research questions:

- RQ 3.1 What kinds of visualization and interaction idioms are effective to visualize AOIs of the geo-multi data?
- RQ 3.2 How to calculate the shape of an area of interest based on a group of geographic data points?
- RQ 3.3 How to efficiently re-compute AOI boundaries in an incremental manner as per user’s update of their AOI query?

1.3 Contributions

A guideline of visualization research. Figure 1.5 presents the four nested levels of design at the visualization research, proposed by Munzner [2014]. There are two types of visualization work: (i) Problem-driven work starts at the top domain situation level and the way down through abstraction, idiom, and algorithm decisions; (ii) In technique-driven work, we work at one of the bottom two levels, idiom or algorithm design, where the goal is to invent new idioms that better support existing abstractions, or new algorithms that better support existing idioms. The contributions of this thesis correspond to each of the three research questions. Figure 1.6 illustrates how each research question contributes to the four levels of visualization design [Munzner, 2014].

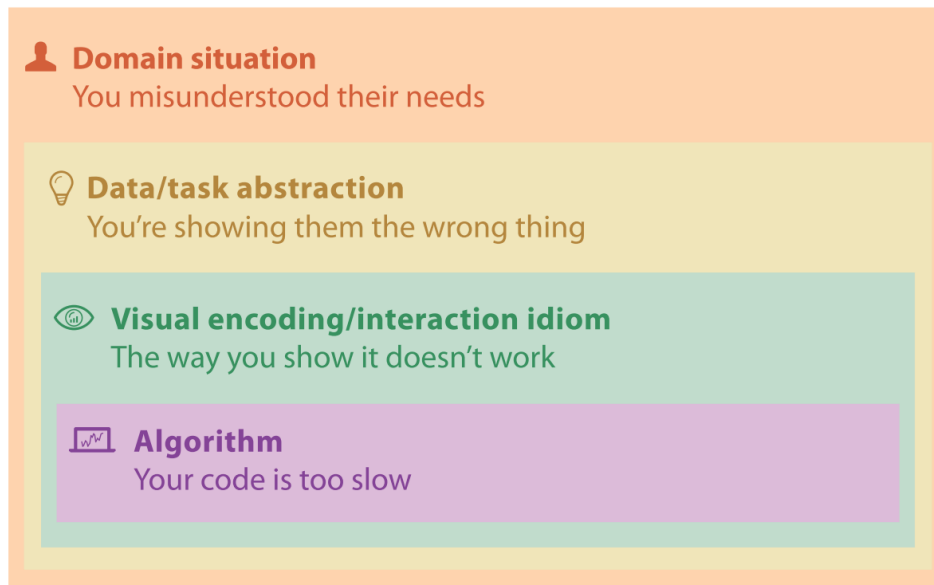


Figure 1.5: Four nested levels of vis design and the related design problem at each level (source: [Munzner, 2014]).





		RQ1. Visual Analytics of Real Estate Data	RQ2. Supporting Large-scale Geographic Visualization	RQ3. Interactive Visualization of Urban Areas of Interest
Domain Situation		✓		
Data/Task Abstraction		✓	✓	
Visual Encoding/ Interaction Idiom		✓	✓	✓
Algorithm			✓	✓

Figure 1.6: An overview of our three research questions mapped to four levels of visualization design (defined in [Munzner, 2014]).

1.3.1 HomeSeeker: a visual analytics system of real estate data

Our first research question is problem-driven and includes three levels of design contributions (Figure 1.6). First, we propose a systematic problem abstraction that guides the visualization design of real estate data for home buyers. Particularly, we collect data from different channels and integrate them to get a comprehensive location-centred real estate dataset. Second, we design an interactive visual analytic procedure to help zero-knowledge users gradually understand the real estate market, and ultimately find appropriate properties based on their individual preferences. Third, we propose a series of novel visualization designs with multiple coordinated views to visualize real estate data, which can also be used to solve other geo-related multidimensional analysis problems. Specifically, we propose a set of visualization designs (in one interface) which are interconnected to each other to support multiple views of different subsets of attributes as per different user’s preferences or interests.

Integrating the three-level visualization design, we build a visual analytics system named HomeSeeker (<http://115.146.89.158>) for real estate users (including home buyers, real estate investors and agents) to visually explore and analyze the geo-related multidimensional real estate data. HomeSeeker augments existing commercial real estate systems and assists users in understanding the local areas and local real estate market, exploring and finding candidate properties based on their individual requirements, visually comparing properties/suburbs in multiple aspects as given by the user, and finally finding candidate properties that suit for their lifestyles and also possibly grow in value over time.

1.3.2 ConcaveCubes: a cluster-based data cube supporting large-scale geographic data visualization

Our second research question (RQ2) starts from the abstraction level and deals with the scalability problem (Figure 1.6). First, we propose and design a cluster-based visualization abstraction of geographical data. Second, we propose an algorithm to efficiently generate concave hulls that include geographical points in each cluster as compactly as possible. As a result, it is able to avoid large empty areas inside the hull, complex shapes and overlaps among different clusters. Third, we propose a cluster-based data cube (ConcaveCubes) to efficiently support interactive response to users’ visualized exploration on large-scale geographic multidimensional data.

We build a demo system (available at <http://115.146.89.158/ConcaveCubes>) to demonstrate the design of cluster-based geographic visualization. We conduct extensive experiments using real-world datasets and compare ConcaveCubes with state-of-the-art cube-based data structures

to verify the efficiency and effectiveness of ConcaveCubes.

1.3.3 AOI-shapes: a parameter-free footprint method supporting visualization of AOIs

Our last research question (RQ3) deals with the bottom two levels of visualization design (Figure 1.6). After a study of data and tasks related to AOI visualization, we propose visualization idioms and illustrate why a novel footprint method is needed to support the visualization of AOIs. Then we propose a footprint algorithm named AOI-shapes to calculate the geographic shape of disconnected urban areas of interest based on a group of geographic data points. AOI-shapes can be easily extended to incrementally compute the boundary of AOIs after user interaction and avoid the interactive scalability problem.

We present extensive experiments to demonstrate both the efficiency and effectiveness of our proposed AOI-shapes algorithm. An online demo system is available for public access at <http://www.aoishapes.com>.

1.4 Research Outcomes

In this section, we first present the papers published by the candidate during this Ph.D. study; then, we describe some additional outcomes, e.g., published dataset and several online systems.

1.4.1 Publications

The following papers that are published by the candidate are directly related to this Ph.D. thesis. For each paper, we refer to the corresponding chapter in which the content of the paper is included.

- **Mingzhao Li**, Zhifeng Bao, Timos Sellis, Shi Yan, & Rui Zhang. HomeSeeker: a visual analytics system of real estate data. *Journal of Visual Languages & Computing*, vol. 45, pp. 1–16, 2018. The content of this paper is included in Chapter 2 & 3.
- **Mingzhao Li**, Zhifeng Bao, Timos Sellis, & Shi Yan. Visualization-aided exploration of the real estate data. *Proceedings of Australasian Database Conference (ADC)*, pp. 435–439, 2016. This work won **the Best Demo Paper Award** at ADC 2016. The content of this paper is included in Chapter 3.

- **Mingzhao Li**, Zhifeng Bao, Farhana Choudhury, & Timos Sellis. *Supporting large-scale geographical visualization in a multi-granularity way. Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 767–770, 2018. The content of this paper is included in Chapter 4.
- **Mingzhao Li**, Farhana Choudhury, Zhifeng Bao, Hanan Samet, & Timos Sellis. Concave-Cubes: supporting cluster-based geographical visualization in large data scale. *Computer Graphics Forum (Proceedings of EuroVis 2018)*, vol. 37, no. 3, pp. 217–228, 2018. The content of this paper is included in Chapter 4.
- **Mingzhao Li**, Zhifeng Bao, Farhana Choudhury, & Timos Sellis. Interactive visualization of urban areas of interest: a parameter-free and efficient footprint method. *Proceedings of the 12th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 782–785, 2019. The content of this paper is included in Chapter 5.
- **Mingzhao Li**, Zhifeng Bao, Farhana Choudhury, Hanan Samet, Matt Duckham, & Timos Sellis. AOI-shapes: supporting interactive visualization of user-defined urban areas of interest. Under review in *IEEE Transactions on Visualization and Computer Graphics*. The content of this paper is included in Chapter 5.

Additional publications.

During the period of the Ph.D., the following papers have also been published, but not included in this thesis, as they are not directly connected to the research topic.

- **Mingzhao Li**, Zhifeng Bao, Liangjun Song, & Henry Duh. Social-aware visualized exploration of tourist behaviours. *International Conference on Big Data and Smart Computing (BigComp)*, pp. 289–292, 2016.
- **Mingzhao Li**, Zhifeng Bao, Shi Yan, & Timos Sellis. A visual analytics framework for the housing estate data. *IEEE PacificVis (Poster)*, 2016.
- Tao Guo, **Mingzhao Li**, Peishan Li, Zhifeng Bao, & Gao Cong. POIsam: a system for efficient selection of large-scale geospatial data on maps. *International Conference on Management of Data (SIGMOD)*, pp. 1677–1680, 2018.
- Sheng Wang, **Mingzhao Li**, Yipeng Zhang, Zhifeng Bao, David Tedjopurnomo, & Xiaolin Qin. Trip planning by an integrated search paradigm. *International Conference on Management of Data (SIGMOD)*. 1673–1676, 2018.

- Qiuhui Zhu, Min Zhu, **Mingzhao Li**, Min Fu, Zhibiao Huang, Qihong Gan, & Zhenghao Zhou. Transportation modes behaviour analysis based on raw GPS dataset. *International Journal of Embedded Systems*, vol. 10, no. 2, pp. 126-136, 2018.
- Binbin Lu, Min Zhu, Qinglai He, **Mingzhao Li**, & Ruoyu Jia. TMNVis: visual analysis of evolution in temporal multivariate network at multiple granularities. *Journal of Visual Languages and Computing*, vol. 43, pp. 30–41, 2017.
- Min Fu, Min Zhu, Yabo Su, Qiuhui Zhu, & **Mingzhao Li**. Modeling temporal behavior to identify potential experts in question answering communities. *International Conference on Cooperative Design, Visualization and Engineering*, pp. 51-58, 2016.

1.4.2 Data and systems

Apart from the published paper, the outcome of this thesis also includes the following data and systems:

- A comprehensive real estate dataset which includes 1.5 million real estate properties sold in Australia between 2006 to 2018, and 72 dimensions associated with each individual property;
- A preliminary online real estate system (<http://115.146.89.158>) to help home buyers understand the local real estate market and find their ideal home;
- A demonstration system of ConcaveCubes (<http://115.146.89.158/ConcaveCubes>); and
- An online demonstration system of AOI-shapes (<http://aoishapes.com>).

1.5 Thesis Structure

The rest of the thesis is organized as follows: Chapter 2 presents an overview of the fundamental literature in visualization. Chapters 3-5 correspond to each of our three research questions respectively. Finally, Chapter 6 summarizes the whole thesis and discusses possible extensions of our work.

Chapter 2

Literature Review

“A picture is worth a thousand words.”

— Tess Flanders, 1911

In this chapter, we first give an introduction of the visualization and visual analytics research, and discuss when visual analytics is needed and how a visual analytics system can be designed. Then, we review existing visualization methods for geo-related multidimensional data, including multidimensional visualization methods, geographic visualization methods and composite visualization strategies. Finally, we discuss two types of methods (i.e., visual reduction methods and data reduction methods) to support large-scale data visualization, and summarize this chapter in the end.

2.1 Data Visualization and Visual Analytics

Visualization refers to any technique for creating images, diagrams, or animations to communicate a message [Ward et al., 2015]. In most cases, communication via visualization can be more efficient than text. On the one hand, pictures can be processed by humans much more quickly compared to text which contains the same amount of information. This is because image interpretation is performed in parallel within the human perceptual system, while the speed of text understanding is limited by the sequential process of reading [Patterson et al., 2014]. On the other hand, pictures can also be independent of local language, as a graph or a map may be understood by a group of people with no common tongue. As a result, visualization has been used to effectively record and present information and solve practical problems. For example, John Snow’s map of the deaths [Tufte, 2001] helped him to find the origin and the transmission way of cholera at London in 1854.

While the use of visualization to convey information has a long history, computer-based data visualization provides visual representations of datasets designed to help users carry out tasks more effectively. The users of a visualization system can be data analysts (e.g., urban computing researchers), domain experts (e.g., real estate agents) or general users of a specific purpose (e.g., home buyers). There are at least two general aims for using data visualizations:

- **Presentation and communication.** As we explained before, visualization can present information in a more effective way for humans to understand, and it can be used as a way of communication even among a group of people speaking different languages. While humans (often visualization designers) can control how the visual elements are presented to let humans perceive the information more effectively, computers can help to generate the graphs more efficiently.
- **Exploration and discovery.** The central part of data analysis is to explore the data and discover patterns and trends, both to confirm expected hypotheses and to discover unexpected insights. Visualization-based data exploration is complementary to computational analysis and provides a more detailed and visual view of the data than just a few statistical features. While a modern dataset is often too complex to be presented in a single picture, one of the most popular and effective strategies is to involve human interactions to control the granularity of the visualization. Following Shneiderman’s visual information-seeking mantra: “overview first, zoom and filter, then details-on-demand” [Shneiderman, 1996], interactions allow users to provide their understanding of the information (presented in the current visualization) as an input to the next-step visualization, so that the visualization system can combine both the advantages of human intelligence and machine computational abilities. We refer to this kind of data analytics via interactive visual interfaces as “visual analytics” [Ward et al., 2015].

Visual Analytics. Traditional visualizations can be divided into two major categories [Nagel, 2006]: (i) Scientific visualization (SciVis) which focuses on the visualization of physical (non-abstract) data (e.g., the structure of human bodies, wind flow data); and (ii) Information visualization (InfoVis) which focuses on the visualization of abstract, non-physically data (e.g., text data, multidimensional data, network data). Visual analytics is an outgrowth of the fields of InfoVis and SciVis that focuses on analytical reasoning facilitated by interactive visual interfaces [Ward et al., 2015]. It is suitable when there is a need to augment human capabilities rather than replace people with computational decision-making methods. Compared to traditional InfoVis

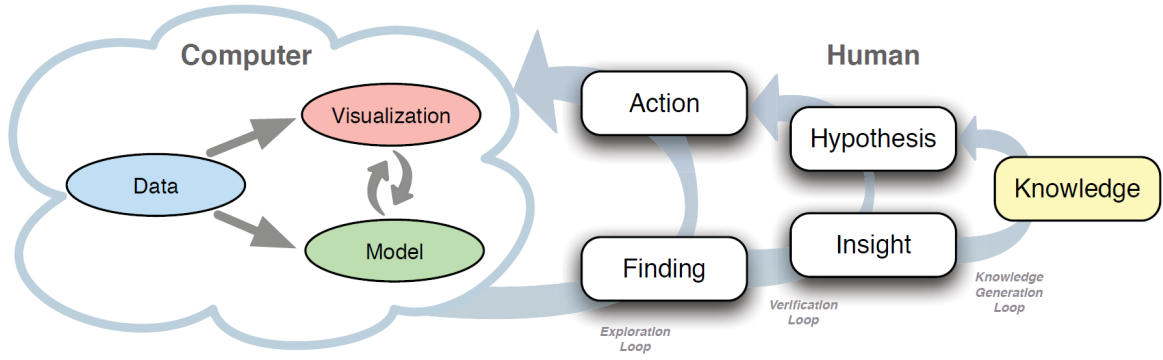


Figure 2.1: *Knowledge generation model for visual analytics (source: [Sacha et al., 2014]). The model consists of a computer part and a human part. The left-hand side illustrates a visual analytics system, whereas the right-hand side illustrates the knowledge generation process of the human.*

and SciVis which focus more on the data representation process, visual analytics includes a process of data exploration and knowledge generation, as shown in Figure 2.1. Users can explore data supported by the visual analytic system with user interactions and observing the feedback. The verification loop guides the exploration loop to confirm hypotheses or form new ones. If a concrete analysis goal is missing, the exploration loop becomes a search for findings, which may lead to new analytical goals. Even when the exploration loop is controlled by an analysis goal, the resulting findings are not necessarily related to it but can lead to insights solving different tasks or opening new analytical directions. In this thesis (esp. the first research question), we explore the data exploration process (regarding geo-related multidimensional data) based on a real-world scenario when home buyers try to find a real estate property to purchase. Some users might have specific requirements such as finding a property within one million dollars in the eastern suburbs. Most users do not know what kinds of property they want until they understand more of the local real estate market and different lifestyles in different regions, and then gradually form a more concrete analysis/search goal.

The design process of a visual analytics system. Many researchers have discussed the design process of visualization and visual analytics systems. For example, Card et al. [1999] describes the process of information visualization design from raw data processing to visual mapping and view transformations, with user interactions existing in any of those processes. The design process in this thesis is guided by one of the most visual analysis frameworks proposed by Munzner [2014], which splits the complex problem of visual analytics design into the following four cascading levels:

- **Domain situation**, where the details of a particular application domain¹ for visual analytics is considered. This is important as the process of visual analytics design has been believed to be extremely domain-specific, requiring customization and fine-tuning for any new applications [Parameswaran, 2018, Munzner, 2014]. The outcome of the design process is an understanding that the designer reaches about the needs of the user. The methods typically used by designers to identify domain situation blocks include interviews, observations, or careful research about target users within a specific domain. Our first research question uses the real estate domain as an example to analyze the geo-related multidimensional visualization problem.
- **Data/task abstraction**, i.e., a what-why abstraction level [Heer et al., 2012], where we map those domain-specific problems and data into forms that are independent of the domain. First, questions from very different domain situations can map to the same abstract visualization tasks. Second, Selecting a data block is a creative design step rather than simply an act of identification. In this thesis, both our first and second research questions include data/task abstraction, with RQ1 transforming the real estate domain situation into visualization data and tasks, and RQ2 focusing on data/task design related to cluster-based geographic visualization.
- **Visual encoding/interaction idiom**, i.e., the design of idioms that specify the approach to visual encoding and interaction. Visual encoding is a creative process that maps data with graphical elements called marks (e.g., points, lines, areas), and visual channels (e.g., position, color, size, shape) to control their appearance [Maguire et al., 2012]. While the visual encoding idiom controls exactly what users see, the interaction idiom controls how users change what they see. All the three research questions in this thesis have included this level of visualization design. We have presented novel visual encoding designs with careful design justification related to each of the research questions.
- **Algorithms** to instantiate the visualization idioms computationally. In this step, the goal is to efficiently handle the visual encoding and interaction idioms that are chosen in the previous level. Both our second and third research questions include such level of contributions. RQ2 deals with the visualization scalability problems when visualizing a large number of geographic points on maps. RQ3 proposes an algorithm to effectively calculate the shape of a region (AOI).

¹The term “domain” is frequently used in the visualization literature to mean a particular field of interest of the target users of a visualization tool, for example microbiology or high-energy physics or e-commerce.

Most visualization works can be divided into the following two types [Munzner, 2014]: (i) problem-driven work, where we start at the top domain situation level and work your way down through abstraction, idiom, and algorithm decisions (RQ1); and (ii) technique-driven work, where we mainly work at one of the bottom two levels, idiom or algorithm design, and our goal is to invent new idioms that better support existing abstractions (RQ1, RQ2, RQ3), or new algorithms that better support existing idioms (RQ2, RQ3).

2.2 Visualization Methods for Geo-related Multidimensional Data

In this section, we review existing visualization methods for geo-related multidimensional data. Particularly, we first summarize multidimensional visualization and geographic visualization separately, then we discuss some composite visualization design strategies.

2.2.1 Multidimensional visualization methods

Multidimensional visualization (or multivariate visualization²) helps users discover hidden patterns and understand the complex relationships with respect to multiple dimensions. Based on the differences of graph representations [Ware, 2012], we divide multidimensional visualization methods into the following categories:

- *point-based methods*: scatterplots [Carr et al., 1987], RadViz [Hoffman et al., 1997], etc.
- *line-based methods*: parallel coordinates [Siirtola and R  ih  , 2006], multivariate line graphs [Hemminger, 1972], Andrew Curves [Andrews, 1972], star graphs [Rubio-Sanchez et al., 2016], etc.
- *region-based methods*: multivariate bar charts [Ware, 2012], Heatmap [Wilkinson and Friendly, 2012], etc.
- *combined methods* such as multivariate glyphs [Ward et al., 2010] and pixel-based methods ([Keim, 2000]).

²In this thesis, we will not distinguish between multidimensional visualization and multivariate visualization. We use the term multidimensional visualization throughout the proposal to refer to the visualization of the dataset that has more than about five variables. Some other works (e.g., Hoffman and Grinstein [2002]) make a distinction between independent and dependent variables and refer to independent multiple variables with the term multidimensional and to dependent multiple variables with the term multivariate.

Table 2.1: *A summary of multidimensional visualization methods.*

Category	Method	Dimension Reduction	No. of Dimensions	Advantages	Disadvantages
Based on points	Scatterplot Matrix [Carr et al., 1987]	No	[2, 10]	Relations between each pair of dimensions are clear.	Overview of the whole dataset is missing.
	RadViz [Hoffman et al., 1997]	YES	[3, ~50]	Clustering effect is good.	Relations among the dimensions are missing.
Based on lines	Parallel Coordinates [Siirtola and R��ih��, 2006]	No	[2, ~20]	The multiple characteristics are clearly displayed.	Complex interactions are needed.
	Multivariate Line Graphs [Hemminger, 1972]	No	[1, ~20]	Distribution of each dimension is clear.	Relations among dimensions are hard to discover.
	Andrew Curves [Andrews, 1972]	Yes	[1, + ∞]	Linear relations among dimensions are clear.	It is complex to compute and hard to understand for users.
	Star Graphs [Rubio-Sanchez et al., 2016]	No	[3, ~20]	Space utility rate is high.	It is easy to confuse cognitively, especially the central part.
Based on regions	Multivariate Bar Chart [Ware, 2012]	No	[1, ~5]	Simple, and easy to understand	It only applies to very limited dimensions.
	Heatmap [Wilkinson and Friendly, 2012]	No	[1, ~50]	Colour mapping is intuitive.	Good colour recognition ability is needed from users.
Combined methods	Multivariate Glyphs [Chuah and Eick, 1998, Ward et al., 2010]	Yes	[1, ~10]	The design is simple and interesting.	Users need to memorize the mapping patterns
	Pixel Display [Keim, 2000]	No	[1, ~256]	Data periodicity is displayed.	Good colour recognition ability is needed from users.

We summarize those four categories of multidimensional visualization methods in Table 2.1, and analyze each method based on whether dimension reduction is applied and the number of dimensions that it supports. In the last two columns of Table 2.1, we discuss the most notable advantages and disadvantages of each method. We have the following observations based on the comparison. First, each method has its own advantages and disadvantages of depicting different features of multidimensional data. For example, parallel coordinates [Siirtola and R  ih  , 2006] perform well to display the multiple characteristics of the dataset, but complex

interactions are needed; relations between each pair of dimensions are easily depicted with scatterplot matrix [Carr et al., 1987], but an overview of the whole dataset is missing. Second, Visualization methods with dimension reduction can handle more dimensions. Third, Line- and point-based methods have better space utility rates than region-based methods. Finally, Line-based methods have better presentations of the multiple attributes and relations among the dimensions compared with other methods.

While each method might focus on visualizing some aspects of the multidimensional data, many researchers have also integrated different multidimensional visualization methods to solve practical problems. For example, Yuan et al. [2009] proposed SPPC which makes use of the advantages of both parallel coordinates and scatterplot to visualize the multidimensional dataset. Viau et al. [2010] designed P-SPLM, a hybrid approach that combines a scatterplot matrix and parallel coordinates to visualize and select features within the network.

2.2.2 Geographic visualization methods

Geographical Visualization (geoViz) communicates geospatial information in ways that, when combined with human understanding, allow for data exploration and decision-making processes [Jiang and Li, 2005]. Researchers started to use scientific visualization methods to portray geographical information from the early 1990s [Taylor, 1994]. MacEachren and Taylor [2013] explored the way of a more scientific approach to visualizing the geographical phenomena. The research at the time focused more on the application to provide users imagining spatial relationships, and was applied to a number of visualization types, including animations, spatial modeling, and collaboration, using GIS, mobile computing, virtual reality, etc.

In recent years, more information visualization methods have been applied to solve the geographic visualization problem. The design of geoViz is not limited to mapping elements based on their exact geo-location. One of the most notable examples is the design of modern train maps [Vertesi, 2008], such as Beck’s London Underground Maps. The design challenge behind is to reduce connection overlaps while places are still easy to find. Many existing works have shown that places, even not mapped based on their original features, such as the location [Vertesi, 2008] and the boundary shape [Ward et al., 2010], could be still or even better recognizable. With urban data (e.g., the census data) becoming more and more available, many online visualization tools have been developed in recent years to help citizens understand the urban life [Foth, 2015]. For example, AURIN (Australian Urban Research Infrastructure Network) [Pettit, 2015] provides the visualization for informing smarter city planning based

Table 2.2: *A summary of visualization methods for geographic point data.*

		Location	Colour	Shape	Size	Texture
Point-based	Dot Map	Yes	Yes	Yes	Yes	Yes
	Heat Map	Approximate	pre-defined	No	No	No
Region-based	Bubble Map	Approximate	Yes	Yes	Yes	Yes
	Choropleth Map	Yes	Yes	Pre-defined	Pre-defined	Yes
	Voronoi Diagram	Pre-defined	Yes	Pre-defined	No	Yes
	Binned Plot	Approximate	Yes	No	No	Yes
Cluster-based	Cluster Map	Approximate	Yes	Yes	Yes	Yes
	Boundary Map	Yes	Yes	Pre-defined	Pre-defined	Yes

Yes/No: The visual channel is/is not supported; Approximate: The visual channel is approximately supported;
Pre-defined: the visual channel is used for other purposes.

on heterogeneous data resources from the definitive data across Australia. The CityX Lab³ also presents a variety of visualizations to help understand the ethnicity and age group in Melbourne, the air travel in Australia, etc. Those tools use colour-based mapping on Maps to depict a geographically related variable. However, this is only effective when there are only a few attributes: since position- and size-based visual variables are already used by maps, retinal variables to depict other attributes are limited.

Visualizing geographic point data. While geographic trajectory (network) data is out of our scope, we mainly discuss the visualization of geographic point data⁴ in this thesis. Many researchers have proposed various visualization methods for geographic point data, as shown in Figure 2.2. We divided them into the following three main categories.

- Point-based methods, which represent each data point as an individual dot on the map.
- Region-based methods, which divide the data points into different groups based on regions. Typically, data can be aggregated by the pre-defined regions such as suburbs and school zones.
- Cluster-based methods, which normally aggregate the data based on data attributes, and then represent each cluster on maps.

³<https://www.cityxlab.com>

⁴In Geographic information system (GIS), vector data is often split into three types: polygon, line (or arc) and point data. “Line” is often associated with path/network, which is out of our scope. In this thesis, geographic point data can be represented as individual “points” or a “polygon” that represent multiple points.

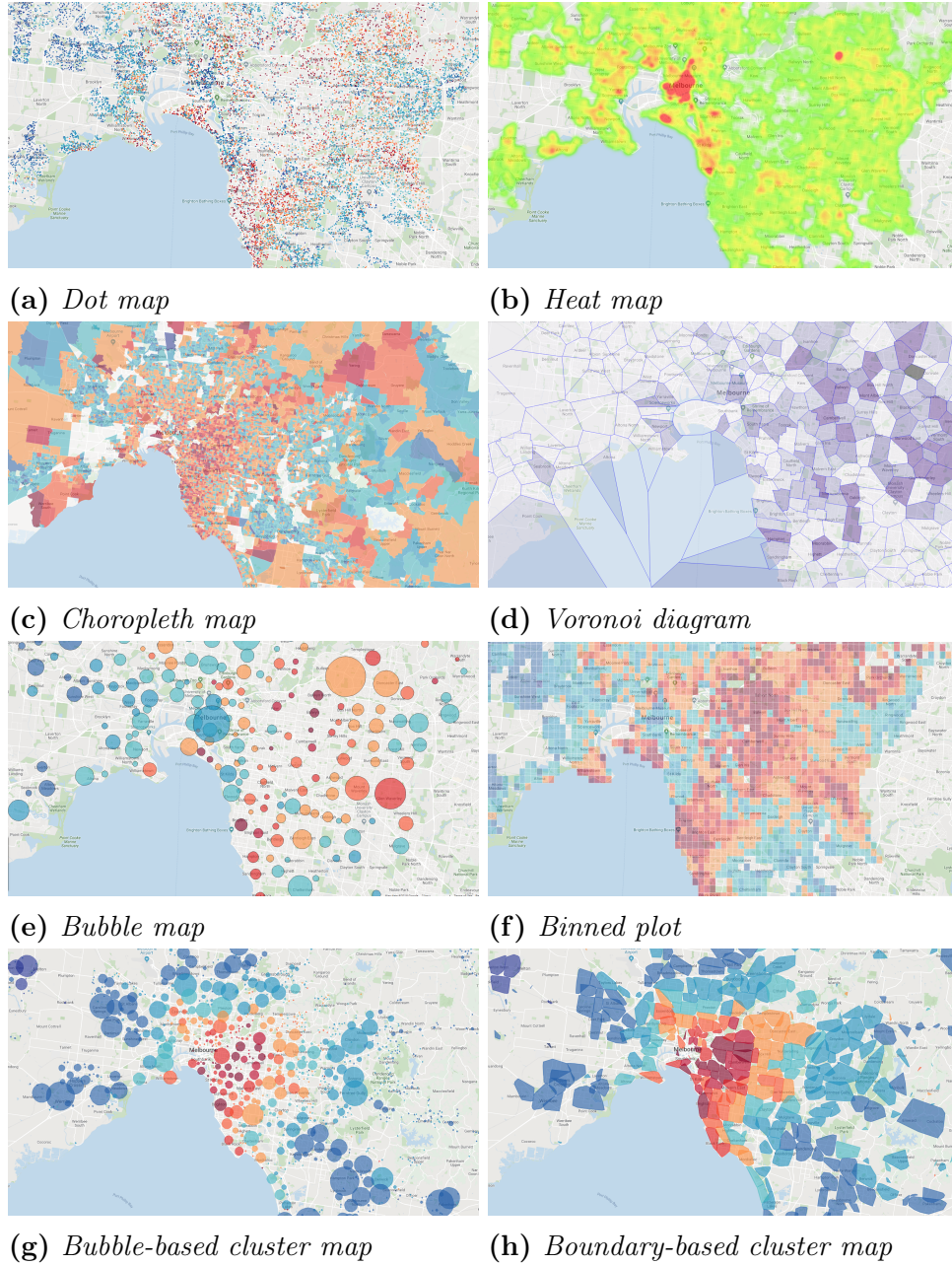


Figure 2.2: Examples of existing visualization methods for geographic point data (All screens are taken from our HomeSeeker website).

Different geographic visualization methods might depict different aspects of geographic patterns. In this thesis, we choose different methods for different visualization tasks with careful design justification. For example, we use bubble maps to visualize the real estate statistics at

the suburb level, and we combine Choropleth maps and Voronoi diagrams to present public school boundaries (details presented in Chapter 3).

2.2.3 Composite visualization design

It has been identified as a challenging work by many researchers to visualize the multiple attributes in the geographic data [Singleton and Longley, 2009]. One of the most popular and effective solutions is to use composite visualization views (CVVs). Javed and Elmqvist [2012] discussed five strategies for CVVs, including

- Juxtaposition, i.e., placing visualizations side-by-side in one view;
- Superimposition, i.e., overlaying two visualizations in a single view, such as different layers of information in Google Maps;
- Overloading, which uses the space of one visualization for another;
- Nesting, which nests the contents of one visualization inside another visualization; and
- Integration, which places visualizations in the same view with visual links.

Following those strategies, Guo et al. [2011] designed TripVista with multiple views to analyze the traffic at a road intersection. Turkay et al. [2014] proposed attribute signatures to dynamically generate summaries in another view after selecting elements on a map view. In this thesis, we have used several of those CVV strategies. For example, all of our three research questions use integration strategies with multiple coordinated views and the superimposition strategy to place multiple layers on maps; and in the first research question, we propose a composite visualization design (using the juxtaposition design) to visualize both multidimensional and geographic attributes in a view, by integrating parallel coordinates with a geo-coded scatter plot and a coloured Boolean table in a seamless way.

2.3 Supporting Large-scale Data Visualization

The increase of data size might result that an elegantly designed visualization system unusable. The following three types of resource limitations [Munzner, 2014] need to be considered to address such scalability problems.

- **Display capacity.** Visualization designers often run out of pixels, i.e., the resolution of the screen is not enough to show all the desired information simultaneously.
- **Human perceptual and cognitive capacity.** On the human side, people have limited ability to recognize/rank different visual channels (e.g., color, textual information), and humans' memory and attention are finite resources as well.
- **Computational capacity.** Same as any applications in computer science, computer time and memory are limited resources for visualization systems. In visual analytics, the latency of interaction, i.e, how much time it takes for the system to respond to the user action, matters immensely for interaction design. Existing research shows that users might feel latency if the response time is larger than 0.1 seconds [Card et al., 1991].

In this section, we discuss some general solutions to address the scalability of visualization systems. The techniques to handle large-scale data visualization can be broadly classified into two categories: (i) visualization methods, and (ii) data reduction methods. It is worth noting that many studies exploit techniques in both categories to achieve a scalable solution.

2.3.1 Visualization reduction methods

Different techniques are proposed to address the perceptual and interactive scalability problem at the visualization end. In particular, the following three types of methods are proposed.

Visual reduction. Since the screen resolution is limited, retinal variables to represent different attributes in a large dataset will easily clutter visually. To this end, researchers have proposed a number of methods, such as pixel-oriented visualization methods [Keim, 1996], alpha blending [Jerding and Stasko, 1998], spatial displacement methods [Trutschl et al., 2003], and dimension re-ordering [Peng et al., 2004] for parallel coordinates. However, all of these techniques still need to scan and draw each data item, which will cause interactive scalability problems.

Progressive methods. Progressive visual analytics [Stolper et al., 2014] presents visualization results in a progressive way, so that the system will be responsive to users' interactions immediately. The method is also associated with approximate query processing techniques, which provide query result estimations with bounded errors [Galakatos et al., 2017]. However, experiments conducted by Turkay et al. [2017b] have shown that possible wrong judgments are easily made before rendering is finished. Even with uncertainty in estimates [Fisher, 2011] reported during the progressive rendering, the result remains unconvincing until the whole progress is completed.

Predictive methods. Predictive visual analytics [Lu et al., 2017] often use query pre-fetching techniques, where the performance depends on the level of predictability of future queries. For example, the Atlas system [Chan et al., 2008] stores large historical time-series data, allows six directions of exploration (pan left/right, scroll up/down, and zoom in/out). The algorithm is based on the observation that a sense of momentum is associated with the direction of exploration. However, these techniques only work well in those domains where user interactions are naturally limited.

2.3.2 Data reduction methods

We refer to data reduction methods as those techniques that aim to reduce the size of the data before the data enters the visual rendering process. We consider three main directions in reducing the scale of data for different scenarios or purposes.

Sampling & filtering methods. Traditional data reduction methods such as sampling [Drosou and Pitoura, 2012] & filtering [Ahlberg and Shneiderman, 1994] are widely used to select a smaller subset of data before applying standard visualization techniques. However, as argued by Lins et al. [2013], sampling & filtering might elide interesting structures or outliers, and usually cannot provide an overview of the data distribution.

Approximate query processing methods. Several approximate algorithms (M4 [Jugel et al., 2014], VDDA [Jugel et al., 2016], etc.) are proposed by exploiting the fact that a display has a limited number of pixels based on its resolution [Keim, 1996]. These algorithms return approximate results that rasterize to the same image as the exact query result would. However, they can only be applied to a very limited number of visualization methods, e.g., line charts and scatter plots.

Data cube aggregation methods. Data cubes [Gray et al., 1997] have been intensively studied in the area of data warehouses. In this technique, aggregation results of the raw data are pre-computed on pre-defined dimensions to support data exploration. While a complete data cube is often too large to fit in memory, data cubes are often constrained by the number of pixels used in visualization [Keim, 1996]. Based on that principle, many researchers have proposed visualization-constrained data cubes (e.g., imMens [Liu et al., 2013], Nanocubes [Lins et al., 2013], and Hashedcubes [Pahins et al., 2017]) to minimize data memory usage and reduce the query latency. In Chapter 4, we will use the data cube methods to support cluster-based visualization of large-scale geographic data.

2.4 Summary

In this section, we summarized different aspects of visualization research that are related to this thesis. First, we introduced the research of visualization and visual analytics, and discussed the design process of visual analytics systems. Then, we reviewed existing visualization methods for multidimensional data and geographic data respectively and presented possible composite visualization design strategies. At last, we discussed two kinds of methods (data reduction methods and visual reduction methods) for supporting large-scale data visualization.

Chapter 3

A Visual Analytics System of Real Estate Data

“Design is so simple. That’s why it’s so complicated.”

— Paul Rand

The very first research question of this thesis is “what kinds of visual analytics framework/design are effective for geo-related multidimensional data?”. Since visualization design is believed to be highly domain-specific [Parameswaran, 2018, Munzner, 2014], we use the real estate domain as an example to explore our first research question.

In this chapter, we present an interactive visual analytics system for real estate data, named HomeSeeker, to explore the visual analytics process of geo-related multidimensional data. HomeSeeker augments existing commercial systems to help users discover hidden patterns, link various location-centred data to the price, as well as explore, filter and compare the properties, in order to easily find their preferred properties. In particular, we first present a problem abstraction for designing visualizations that help home buyers analyse real estate data. Second, we collect, integrate and clean last ten year’s real estate sold records in Australia as well as their location-related education, facility and transportation profiles, to generate a real multidimensional data repository (the dataset includes 1.42 million properties in Australia and 72 dimensions). Third, we propose an interactive visual analytic procedure to help less informed users gradually learn about the local real estate market, upon which users exploit this learned knowledge to specify their individual requirements in property seeking. Fourth, we propose a series of designs to visualize properties/suburbs in different dimensions and in different granularities. After implementing a system prototype for public access (<http://115.146.89.158>),

we finally present case studies based on real-world datasets and real scenario to demonstrate the usefulness and effectiveness of our system.

3.1 Introduction

We choose the real estate domain as an example to explore the visual analytics design process for geo-related multidimensional data. The reason is two-fold. On the one hand, the process of visual analytics design has been believed to be extremely domain-specific, requiring customization and fine-tuning for any new applications [Parameswaran, 2018, Munzner, 2014]. On the other hand, finding an ideal real estate property is critical to people’s life but the process can be both complicated and time-consuming since users could have many factors to consider, such as budgets, facilities like supermarkets and nearby transportation, as well as public schools associated with children’s education; moreover, most home buyers (esp. first home buyers) also prefer a property that will grow in value over time so they can take a step up the property ladder in a few years [Jacqui, accessed March 2017, Thornhill].

There are several commercial systems, such as Real Estate Australia¹ and Zillow², which provide access for people to browse or search for real estates. We have noticed three common problems in using existing commercial systems.

Problem 1: data in those systems are only about the description of properties themselves such as the number of bedrooms and bathrooms. However, a lot of useful geographically related information which is critical to users’ choices is not captured, such as the regional, educational and transportation profiles (see Section 3.3.1), not to mention presenting them to users. Without such information, it is also very difficult to evaluate the robustness and effectiveness of the current visualization design of those commercial systems in visualizing such geo-related multidimensional information.

Problem 2: existing commercial systems fail to provide an efficient way for users to learn about the local real estate market and get to know their own needs based on their understanding of the market. Since “*users do not know what they want until they see it*” [Pressman, 2005], many users, especially first home buyers, do not know much about the local real estate market; and they might continually change their preferences while they browse properties. For example, a user has a budget of one million dollars to buy a 3-bedroom house in Camberwell and he wants the property within a 10-minute walk to the nearest train station. However, after studying the

¹<http://www.realestate.com.au/>

²<https://www.zillow.com/>

market, he understands that such kind of a house might be not easy to find in Camberwell, so he will either find a property that is further to the station or in other suburbs.

Problem 3: although some of the systems (such as Real Estate Australia¹) provide map-based presentation, the design can only display the locations on the map, which is not sufficient to support comparison of properties in multiple aspects.

In a nutshell, existing systems fail to provide location-aware services, not to mention offering those services at different levels of details, due to the lack of a systematic approach to model and collect a comprehensive location-centred real estate data. To address those problems, we introduce HomeSeeker (<http://115.146.89.158>), an interactive real estate visualization system to augment current commercial systems. In particular, HomeSeeker is able to assist users in understanding the local areas and local real estate market, exploring and finding candidate properties based on their individual requirements, and visually comparing properties/suburbs in multiple aspects as given by the user. In particular, we make the following contributions from the perspective of visualization design.

1. We present a systematic problem abstraction that guides the visualization design of real estate data for home buyers. Particularly, we have collected data from different channels and integrated a comprehensive location-centred real estate dataset (**Section 3.3**).
2. We propose an interactive visual analytic procedure to help zero-knowledge users gradually understand the real estate market, and ultimately find appropriate properties based on their individual preferences (**Section 3.4**).
3. We design and implement a system with existing visualization design choices and novel visualization designs, and multiple coordinated views to visualize real estate data, which can also be used to solve other geographically related multidimensional analysis problems (**Section 3.5**).

The rest of this section is organized as follows. Section 3.2 discusses related works with regard to the real estate domain and visualization. Section 3.3 presents the problem abstraction, followed by a system overview in Section 3.4. In Section 3.5, we present our visualization design choices in detail together with design justifications. Section 3.6 discusses system implementation. We present experiments based on real-world datasets in Section 3.7, followed by a summary in Section 3.8.

3.2 Related Work

Visual exploration of real estate data has been considered as an interesting application and attracted attention from both the academic and industry communities.

Many online platforms in different countries allow people to search for real estate properties, such as Real Estate¹ and Domain³ in Australia, Right Move⁴ in the UK, Zillow² and Trulia⁵ in the USA, and Soufang⁶ in China. Most of those systems provide functions such as searching among or filtering numerous properties available in the real estate market, and then display the result in the form of a list or a map. Some systems such as Zillow also allow users to virtually look into properties of interest using Matterport 3D view. Although these commercial systems have provided users convenient access to available real estates, they still have some drawbacks. First, the data that most of these commercial systems have used is mainly about the property itself, while a lot of useful information that is critical to users' choices is neither captured nor presented. Although some systems have provided information about the neighbourhood, they have not linked such information with individual properties. For example, Trulia⁵ has provided school locations on top of the map, but it does not have the school zone information, and users cannot link individual properties with schools. Second, the visual design of these platforms cannot support presentation and comparison of properties in multiple aspects. In this chapter, we try to augment existing commercial systems by providing a systematic data abstraction related to real estate and a visual analytics procedure that help users understand the local real state market and explore the properties by themselves. Table 3.1 represents the comparison between our data profiles and existing systems. The detail will be illustrated in Section 3.3.

Many researchers from the data mining community have proposed methods and systems [Fu et al., 2015b,a, Fu and Liu, 2016, Tan et al., 2017] related to real estate data. For example, Fu and Xiong [2016] designed a discovery system for users to find high-value homes. Shahbazi et al. [2016] estimated the investability of real estate properties through text analysis. Zhu et al. [2016] proposed a prediction model to measure liquidity in real estate markets based on DOM (Days on Markets). Different from those works, we use interactive visualization to help users first understand the local real estate market, then find properties based on their own preferences. Research on visualization of real estate data dates back to Williamson and Shneiderman's classic Home Finder project [Williamson and Shneiderman, 1992]. They used dynamic query interfaces

³<http://www.domain.com.au/>

⁴<http://www.rightmove.co.uk/>

⁵<https://www.trulia.com/>

⁶<https://www.fang.com/>

to help users explore a real estate database and find houses that meet specific search criteria. They also use sliders to allow limits to be placed on different dimensions such as house price and time to reach one’s workplace; only those houses satisfying all criteria are represented by red dots on a map. Later, Tweedie et al. [1994] proposed Attribute Explorer, which added graphical feedback to the sliders themselves. Both works have focused on the efficient query processing part but failed to provide a way for users to compare the result houses/homes. With real estate data as examples, Spence [2014] presented different kinds of visualization, including scatter plots, parallel coordinates and iconic representations to compare the attributes of houses in the visualization textbook. Nevertheless, there is still a lack of framework/procedure of how to combine those visualization designs and help users explore real estate data. Closest to our work is a web-based system [Sun et al., 2013] to visually analyse Hangzhou real estate market data. Comparing to it, our system is built upon a more comprehensive dataset that has more location-based attributes which are critical to users’ purchasing decisions and we provide a way for users to explore individual properties.

3.3 Problem Abstraction

In this section, we first describe the real estate domain situation, then present the design of our task and data abstraction.

3.3.1 Domain situation

Our system is mainly targeted at home buyers who are interested in finding properties to live in, as well as real estate agents who help them to do so.

We use Melbourne and Sydney, two of the largest cities in Australia, as examples to understand the domain situation and to collect the data. To understand the information (i) that most home buyers are concerned about, and/or (ii) that they are suggested to pay attention to from domain experts’ viewpoint, we first reviewed related articles (from related blogs⁷ such as [Jacqui, accessed March 2017, Thornhill]) written by real estate domain experts, and got a list of information which is critical to users’ property purchase choices; then we discussed with two real estate agents to further refine the data list.

Table 3.1 presents our conclusion of the information that is critical to home buyers’ choices, as well as whether our system has the data and support data index comparing to existing

⁷Source: <http://www.realestate.com.au/advice/buying/>, <https://www.domain.com.au/advice/>

Table 3.1: A summary of our data profiles, and the comparison against existing commercial systems.

Data Example	Real Estate ¹ Rightmove ⁴ Rightmove ⁴ Our system Data* Index* Data Index Data Index						Query Example
Basic Information	price, address, no. of bedrooms	✓	✓	✓	✓	✓	apartments with 3 bedrooms
	geo-information, address, suburb	✓	✓	✓	✓	✓	properties within an area
	airconditioning, heating	✓	✗	✓	✗	✓	houses with air conditioning
	market information	✓	✗	✓	✗	✓	houses in a region with an annual growth rate >10%
Educational Profile	images, floorplan	✓	✗	✓	✗	✓	✗ -
	nearly schools	✓	✓	✓	✗	✓	houses near Camberwell High School
	school rating or ranking	✗	✗	✓	✗	✓	houses near a top 10% high school
	school zones	✗	✗	✗	✓	✓	houses with the school zone of Camberwell High School
Transportational Profile	nearly public transport	✗	✗	✓	✗	✓	houses near Camberwell train station
	travel time to work	✗	✗	✗	✗	✓	houses within 20-min train to work
Facility Profile	supermarkets, clinics	✗	✗	✗	✓	✓	where those facilities locate
	distance to supermarket	✗	✗	✗	✓	✓	houses within 1km to supermarkets
Regional Profile	cultural diversity, income	✓	✗	✗	✗	✓	a region with more people speaking Chinese
Environmental Information	street view	✓	✗	✓	✗	✓	houses within an area with good street views
	tree-lined street	✗	✗	✗	✗	✓	houses within an area with tree-lined streets
	quiet environment	✗	✗	✗	✗	✓	houses within a quiet area

Data: whether the system has the data;
Index: whether the system has the data linked to each property (as index) and support searching with the indexed data.

popular real estate websites. We have defined five profiles: (1) the basic profile describes basic information of each property itself; (2) the transportation profile is associated with distance and travel time to other places; (3) the facility profile defines nearby facilities, such as supermarkets and shopping centers; (4) the educational profile contains the information of schools; (5) the regional profile is related to the census data and describes the statistical information of the neighbourhood. For environmental information: it is critical to users' choice as well, but the data is hard to obtain and verify. Therefore, in our system, we extract some of the environmental information from the text description and include them in the basic profile, such as whether the property has mountain views.

Table 3.1 also presents some example data in each profile. Although including everything in our system seems impossible, we claim that the data that is critical to users' choice of properties should at least fit in one of our profiles. For example, nearby restaurants (which our system does not include) should fit in facility profile. The safety of a region can be described by the regional profile. Price per m^2 is in the basic profile.

3.3.2 Data collection and integration

Figure 3.1 shows the process of how we collect data from different sources and integrate them to form a location-centred real estate dataset.

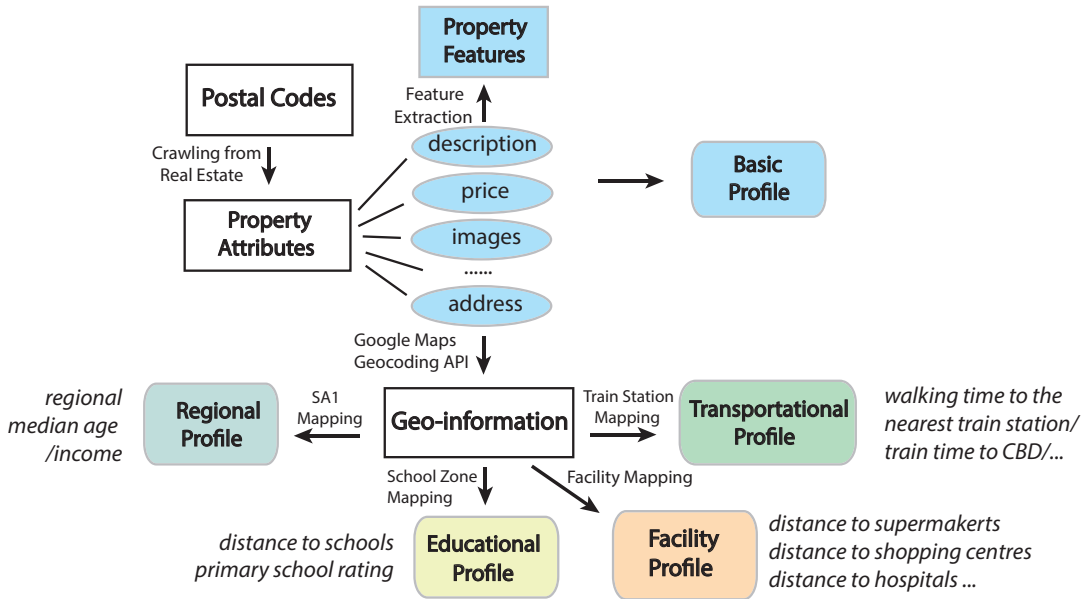


Figure 3.1: Illustration of the process of data collection and integration.

We first crawl the basic information of sold properties from real estate websites¹ (**basic profile**). Based on the address, we get the geo-location (latitude and longitude) of each property using Google Maps API⁸. Based on the geo-location, we describe the other four profiles for each property in the following way:

- **Transportational profile:** we calculate (i) the walking time from each property to its nearest train station using Google Maps API⁸, and (ii) the travel time between each pair of train stations based on the GTFS (General Transit Feed Specification) data⁹. We are then able to compute efficiently how long it takes from each property to a specific place (such as the user's workplace) at runtime.
- **Regional Profile:** we get Australian census data from the Australian Bureau of Statistics website¹⁰ which describes the census information based on different levels of statistical areas in Australia. We then map each property to an SA1 (Statistical Area Level 1, the smallest unit for the processing and release of Australian census data [Australian Bureau of Statistics, 2016a]), and then have the information such as median age and median income of the SA1 to describe the region around each property.
- **Educational profile:** we find either the exact school zone¹¹ that each property belongs to, or map the property with its nearest primary/secondary school. We get the corresponding school rating¹² which is also linked with individual properties.
- **Facility profile:** we first crawl the information (including geo-location) of different facilities such as supermarkets (e.g., Coles¹³, Woolworth¹⁴) and hospitals¹⁵. Then, for each facility, we map each property to the nearest one and calculate the distance between them using Google maps API⁸.

A summary of all data sources can be found in Table 3.1, and some data statistics are shown in Table 3.2.

⁸<https://developers.google.com/maps/>

⁹<https://www.data.vic.gov.au/data/dataset/>

¹⁰<http://www.abs.gov.au/>

¹¹<http://melbourneschoolzones.com/>

¹²<http://bettereducation.com.au/>

¹³<https://www.coles.com.au/store-locator>

¹⁴<https://www.woolworths.com.au/Shop/StoreLocator>

¹⁵<https://healthengine.com.au/>

Table 3.2: *Statistics of the integrated real estate data.*

	Melbourne (VIC)	Sydney (NSW)
# of properties	680,327	757,352
# of train stations	218	373
# of supermarkets	447	507
# of shopping centres	125	194
# of clinics	2,673	3,536
# of hospitals	469	622
# of public primary schools	1,149	1,607
# of public secondary schools	282	401
# of private primary schools	465	567
# of private secondary schools	167	200
# of regions	13,339	17,895
# of total dimensions	72	

3.3.3 Data and task abstraction

In this subsection, we analyse the data attributes and transform the domain tasks into specific visualization tasks.

3.3.3.1 Data characteristic analysis

Based on different characteristics, we divide all attributes associated with each property into the following categories:

- **Geographical Attributes**, i.e., the geo-information;
- **Numeric Attributes**, such as price, number of bedrooms, the distance to the nearest train station, related secondary school rankings;
- **Categorical Attributes**, such as property type;
- **Boolean Attributes**, i.e., 34 features generated from the text description, such as whether a property has air conditioning.
- **Other Types of Attributes**, such as text and images.

3.3.3.2 Task abstraction

Based on the literature [Jacqui, accessed March 2017, Thornhill], and as confirmed by domain experts, most home buyers want a property which suits their lifestyle and will grow in value over time so they can take a step up the property ladder in a few years' time.

There are two main challenges to define specific tasks. On the one hand, users have various levels of knowledge on the local areas and the local real estate market. On the other hand,

users have different individual requirements and their requirements could change during the process of discovering and knowing better about the local real estate market.

Based on the above observation, we define four levels of basic tasks, as shown in Table 3.3. Users with different backgrounds of the local real estate market and different requirements can use different kinds of combinations of the four basic tasks to discover their preferred properties. For example, a user who does not know much of the local real estate market and/or who is not sure of his requirement might go through all the four tasks to find candidate properties; while an advanced home buyer might skip some of the steps and find the properties based on his requirements (Examples will be illustrated as case studies in Section 3.7.2, with a summary of different cases in Table 3.4).

Each basic task includes several sub-tasks, as shown in Table 3.3. The table also presents the corresponding data to support the tasks, as well as the process of how we analyse the tasks and the corresponding data based on Munzner’s *what-why-how* model [Munzner, 2014].

3.4 System Overview

In this section, we first define some design maxims, then design system procedure.

3.4.1 Design maxims

Based on our problem abstraction and the characteristics of our targeted users, we define three design maxims (DMs) to help us design the system and choose proper visualization idioms.

DM1 Incremental learning. Our targeted users are home-buyers, who might have very limited knowledge of the local real estate market. This also greatly affects their ability to describe their requirements. For example, since a person does not know what kind of a property he can buy with one million dollars in Melbourne, he will also easily feel confused in defining the properties that he prefers. Therefore, our system should allow such less informed users to learn about the local real estate market step by step.

DM2 Intuitive and simple designs. Our users, as home-buyers, might not know much about data analytics/visualization. While being informative, our visualization design choices should be also intuitive and simple for end users to easily understand. We should be careful with complex visualization designs; and if we do use them, we should provide detailed guidelines.

Table 3.3: Domain situation, visualization abstraction and visual encoding idioms.

Domain Situation			Vis Abstraction		VIS Design Choices			
Domain Tasks		Example d Data	Task Abstraction ([actions], [targets])	Data Abstraction	Name	Abbr.*	No.*	Example
T.1 Understand Locations	1.1 Understand educational profile	school location, zone and ranking	[present, discover, compare], [features, shapes]	geo, shape, numeric	Choropleth map	CM	1	Fig.3 (a)
	1.2 Understand neighbourhoods	population census data		geo, shape, numeric				
	1.3 Understand transportations	Train stations and distance to work	[present, discover, compare], [features]	geo, numeric	Dot map 1	DM1	2	Fig.3 (b)
	1.4 Understand facilities	location of facilities	[present, discover], [features]	geo, categorical				
T.2 Link Price with Locations	2.1 Overview of price linking with locations	price, sold numbers by regions	① [present, compare], [distribution] ② [discover], [dependency]	geo, numeric	Dot map 2	DM2	3	Fig.4 (b)
		price, sold numbers by regions & bedrooms		geo, categorical, numeric	Glyphs on map 1	GM1	4	Fig.5
	2.2 Compare Historical information of selected locations	price changes over time by regions	① [present, compare], [trends] ② [discover], [outliers]	numeric, temporal	Multiple line charts	MLC	5	Fig.4 (c)
		sold number changes over time by regions		numeric, temporal	Stream graphs	SG	6	Fig.4 (d)
T.3 Explore and Filter Individual Properties	2.3 Present and compare detailed price information of selected locations	median price of properties, capital growth information	① [present, compare], [features] ② [discover], [outliers]	geo, numeric	Parallel coordinates 1	PCs1	7	Fig.6
		price distribution of 2-/3-/4-bedroom houses in different region		geo, numeric	PCs + histograms	PCsH	8	Fig.4 (e)
	3.1 Overview of individual properties in geography	locations, prcie, bedrooms, property type	[present, browse, explore], [features, dependency]	geo, numeric, categorical	Glyphs on map 2	GM2	9	Fig.7 (b)
		geo-locations, suburbs		geo	Geo-coded scatterplot	gSP	10	
T.4 Compare properties in details	3.2 Explore and filter properties based on user-selected attributes	price, bedrooms, distance to train stations	[present, compare], [features, dependency]	numeric	Parallel coordinates	PCs2	11	Fig.7 (c)
		property type, etc.		categorical	Coloured boolean table	CBT	12	
	4.1 Compare selected properties in detail based on user-selected attributes	air conditioning, heating, etc.	[compare], [features]	boolean	Colour d boolean table	WC	13	Fig.7 (e)
		description of properties		text	Word cloud	IC	14	Fig.7 (d)
	4.2 Present how a property differs from other candidate properties	images		images	Image card			
		price, bedrooms, distance to train stations		numeric	Spider chart	SC	15	Fig.7 (f)
		price, bedrooms, distance to train stations	[identify, present], [outliers]	numeric	PCs2, SC, CBT	-	-	-

* Abbr. (Abbreviations) and No. are used in Figure 3.4 and Table 3.5; No. is corresponding to the VIS Design Number in Section 3.5.

DM3 Consistent visualization designs. Since we may involve several different visualization design choices, we should try to use a consistent set of visual encodings in order not to confuse users. For example, colours should have the same meaning in different visualization design choices.

3.4.2 System pipeline

Based on the task abstraction and the DMs, we design an interactive procedure (Figure 3.2) to help different levels of home-buyers incrementally learn about the local real estate market and find their preferred properties.

First, models in our system refer to different descriptive statistics. We have defined different abstracted models based on our basic tasks, as shown in Figure 3.2. For example, we have defined different granularities [Beecham et al., 2016] of summarized information at the suburb level. At the high granularity, we calculate the median price of properties; at the medium granularity, we compute the median price of, for example, 3-bedroom houses or 2-bedroom units; at the low granularity, we summarize the detailed price distribution of all 2-bedroom units, all 3-bedroom houses, etc.

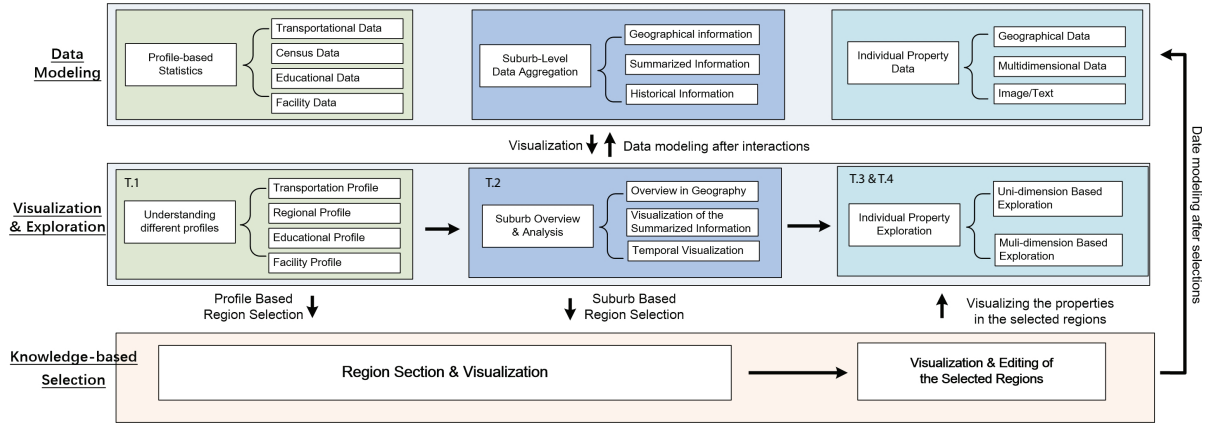


Figure 3.2: The system pipeline of HomeSeeker.

Second, we design three levels of visualization, as profile-based region visualization (T.1, Section 3.5.1), suburb-based visualization (T.2, Section 3.5.2) and property-level visualization (T.3 & T.4, Section 3.5.3) to answer the four basic domain tasks. All the visualization designs are interconnected with each other. On the one hand, we use multiple coordinated views to visualize different information in the same screen, and designs are coordinated with each

other with interactions like mouseover/click+ highlighting. On the other hand, designs cross different levels link with each other based on consistent visual encoding idioms and through user interactions such as region selections. For example, a user can select some school zones in profile-based region visualization, or select several suburbs in the suburb-based visualization, and then move to property-level visualization to explore the properties in those school zones and/or those suburbs; also, the user can search for all the properties in the same school zone in the property-level visualization.

3.5 Visualization Design and Justification

In this section, we first introduce our visualization design choices based on our task and data abstraction. Then, we describe how we have involved users in our iterative design process to make the system fit for general users.

3.5.1 Profile-based region visualization (T.1)

To help home buyers understand different lifestyles in different areas (T.1), we present each of our four profiles using choropleth maps or dot maps.

VIS Design 1: Choropleth map. We use the Choropleth map to visualize the *educational and regional profiles*. For the educational profile, we visualize the school zone on Google Maps. Schools with a better rating will be mapped with a darker colour (Figure 3.3(a)). Similarly, regional exploration presents census information on top of the map. Users can select different information such as median age, median income, the percentage of people coming from China, India, etc.

VIS Design 2: Dot map 1. Since *transportational profile* does not contain shape information, we draw circles on top of the map to represent the location, with colour illustrating the travel time to the city or to users' working place (Figure 3.3(b)). Similarly, we use different colours to represent different types of facilities (supermarkets, or shopping centers) for *facility profile*.

Justification. We choose the Choropleth map which is perhaps the most commonly used and effective method for summarized information with geo-shapes; since the data value (e.g., school ranking) is numeric, we choose colour luminance to represent the value. We choose colour luminance, instead of the area size of the circle to represent the travel time, because otherwise, it might confuse users whether a larger size means a longer time or not. An alternative for

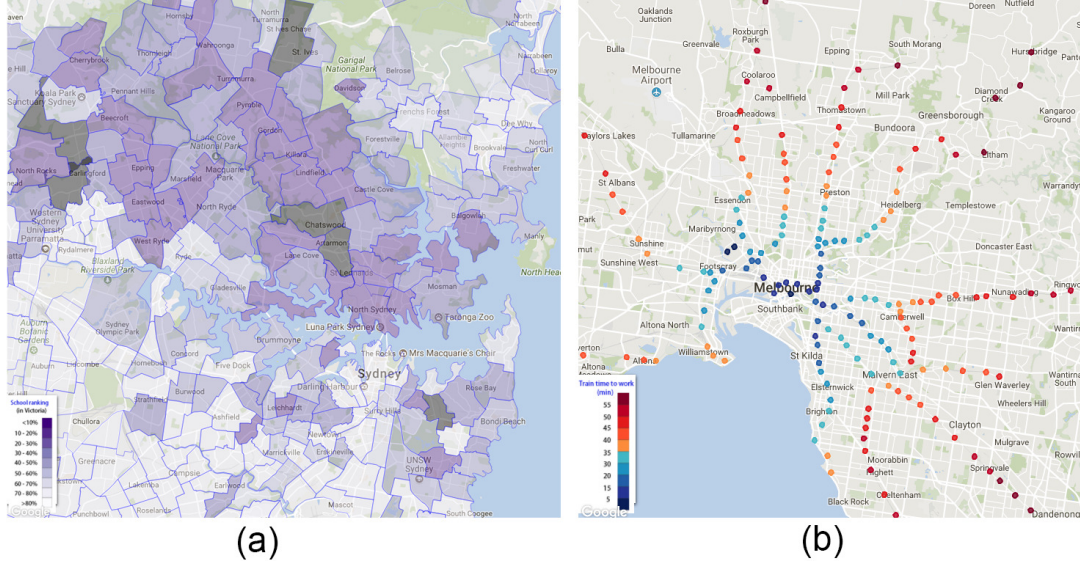


Figure 3.3: Design of profile-based region visualization. (a) VIS design 1: Choropleth map of Sydney's educational profile - primary school exploration; (b) VIS design 2: dot map 1 based on Melbourne's transportaional profile - train time exploration.

visualizing different types of facilities is to use different shapes, but we choose colour hue considering the effectiveness ranking of different visual channels [Munzner, 2014].

3.5.2 Suburb-level visualization (T.2)

At the suburb level, we have three main coordinated views: the Google Maps view, the historical and the multidimensional view (Figure 3.4).

3.5.2.1 The Google maps view (T.2.1)

We design two levels of data abstractions to provide users with an overview of how property prices in different suburbs vary across locations.

VIS Design 3: Dot map 2. To visualize the median price and number of properties in each suburb, we draw circles on top of Google Maps (Figure 3.4 (b)). The position of each circle corresponds to the centre location of a suburb in geography. The size and colour of the circle represent the number of sold properties in this suburb, and the median price of those properties, respectively.

VIS Design 4: Glyphs on map 1. Since prices could be easily affected by the number of bedrooms (or building size in some countries), we further divide the properties in a suburb

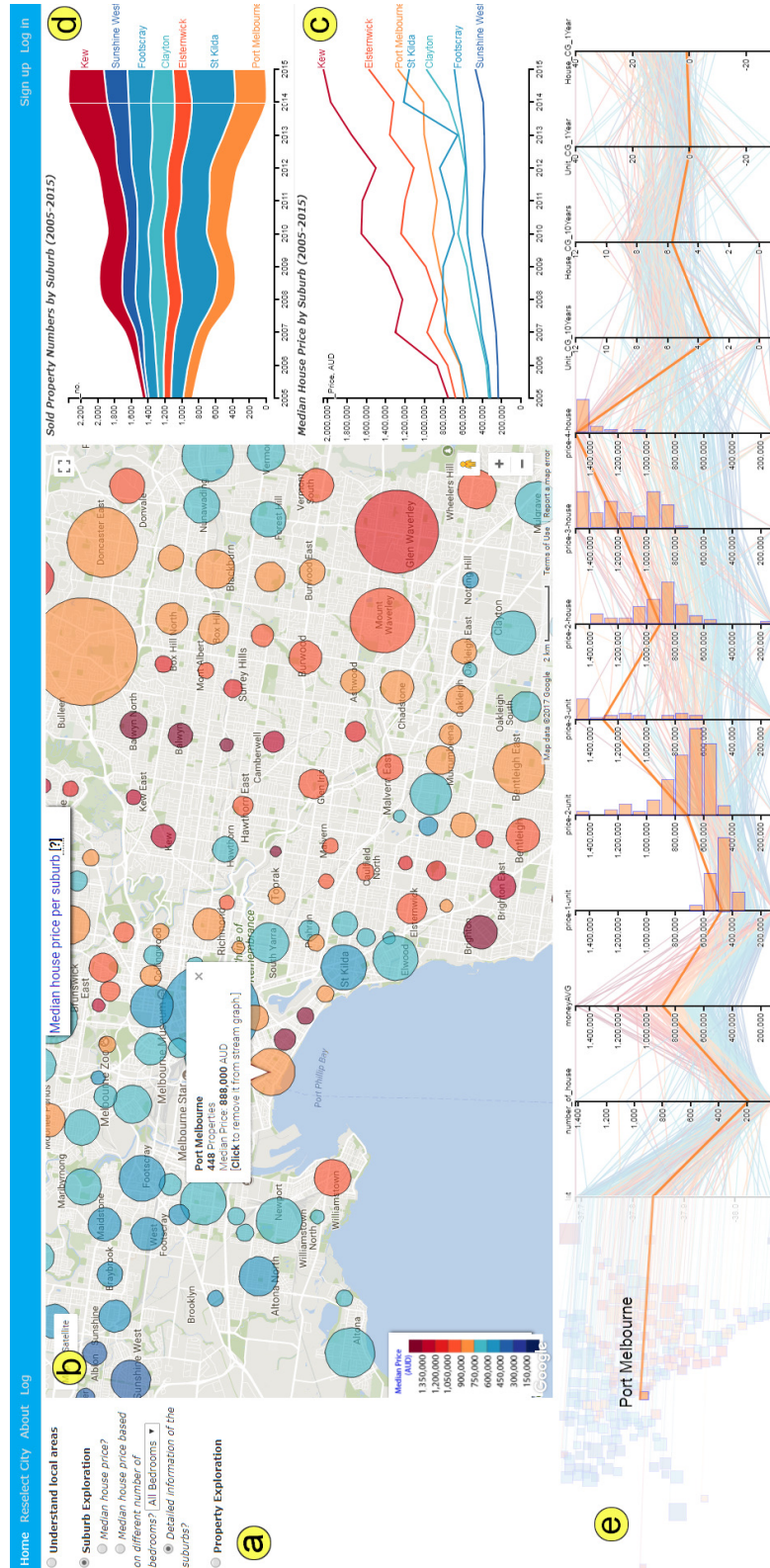


Figure 3.4: Design of the suburb-level visualization. (a) map view (a dot map); (b) historical view - multiple line graphs; (c) historical view - stream graphs; (d) historical view - map view (a dot map); (e) historical view - stream graphs.

into different groups based on the number of bedrooms. Therefore, we depart the circle into different sections to display one more dimension. As shown in Figure 3.5(a), different sections represent different numbers of bedrooms, and the number of properties that have a particular number of bedrooms is presented as the size of the section. The color is based on the median price of the properties in each section. Figure 3.5(b) shows the result when we only select 3-bedroom properties.

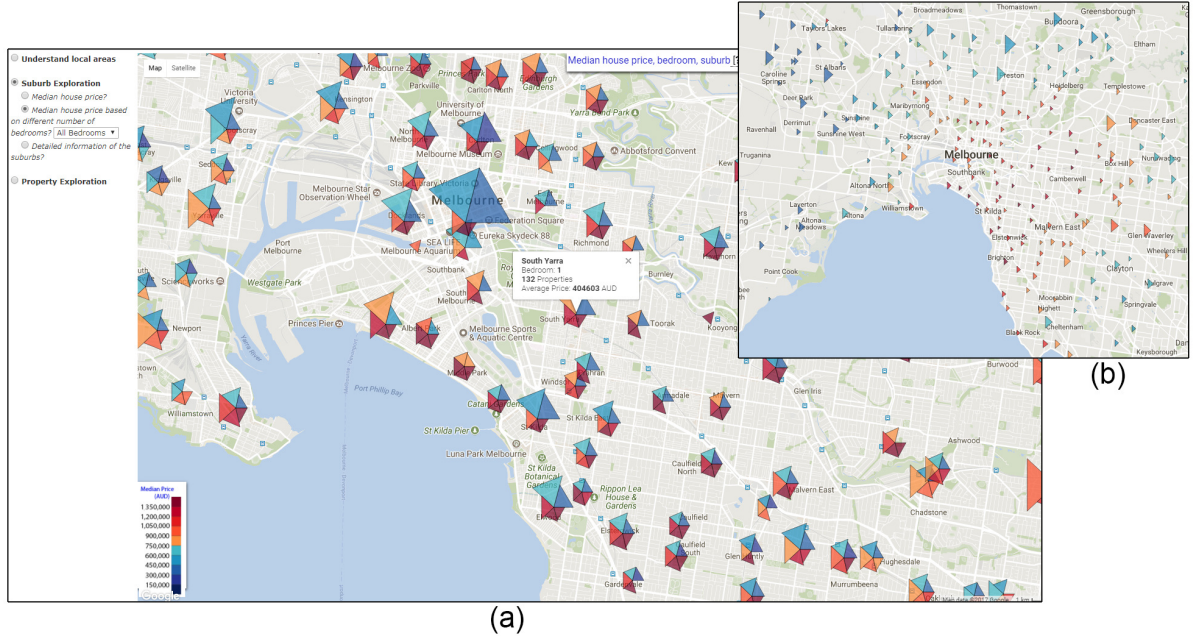


Figure 3.5: Illustration of VIS Design 4: glyphs on maps 1. (a) Median house price per different number of bedrooms; (b) only 3-bedroom properties are selected.

Justification. The two visualization designs are consistent by sharing the same visual channels, i.e., “colour” and “size”. We design a color scheme to present the median price by using different color saturation and two different colour hues (“blue” and “red”). Such colour design is widely used in other applications such as altitude maps [Cohen and Small, 1998]. The color design considers the recommendation from [Brewer and Harrower, accessed March 2017]. Instead of using choropleth maps which show boundaries of suburbs, we choose circles and glyphs on maps, so that we could use more retinal variables, i.e., “size” and “shape”.

3.5.2.2 The historical view (T.2.2)

This view presents how median prices (multiple line charts) and sold numbers (stream graphs) of houses/units in each suburb change over time. Those suburbs can be selected by clicking on

the circles (i.e. suburbs) in the Google Maps view (Section 3.5.2.1), or by filtering the attributes in the multidimensional view (Section 3.5.2.3).

VIS Design 5: Multiple line graphs. As shown in Figure 3.4(c), the horizontal axis represents time, i.e., from 2005 to 2015, and the vertical axis represents the median price. Each line represents the price changes of one suburb.

VIS Design 6: Stream graphs. As shown in Figure 3.4(d), each suburb is represented as a stream. The horizontal axis and colour have the same meaning as that in multiple line graphs. The width of the stream indicates the number of sold house/unit in that particular year.

Justification. Two of the design choices are consistent as they share the same meaning of horizontal axis and colours. The colour, which means the median price, is consistent with all the other designs in the system as well. We choose stream graphs to visualize the sold number, since the sum of the sold number of a group of suburbs make sense, and this is somehow consistent with the design in the Google Maps view (Section 3.5.2.1) since the width of a stream and the area (which can be both considered as size-related visual channels) both mean the numbers. On the other hand, if we use stream graphs to visualize the median price change, the height of stream graphs will easily confuse people, since the sum of median prices does not make sense.

3.5.2.3 The multidimensional view (T.2.3)

The multidimensional view presents detailed statistical information of each suburb.

VIS Design 7: Parallel coordinates (PCs) 1) We use parallel coordinates (Figure 3.6) to visualize suburb-based statistics, such as the median price of 2-bedroom houses, the median age in this suburb. For users to easily compare the price differences of different property types and bedroom numbers, we have all the maximal and minimal number of price-related axes consistent, such as 1.5M and 0.2M. As shown in Figure 3.6, we connect the left of parallel coordinates with a geo-coded scatter plot, to allow users to easily link each suburb represented as lines in parallel coordinates with the one in the Google Maps view.

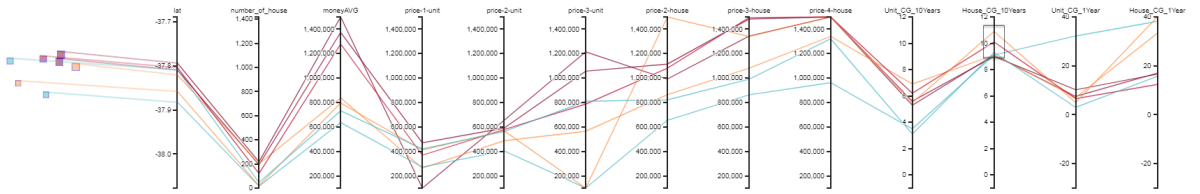


Figure 3.6: Illustration of VIS Design 7: PCs

VIS Design 8: PCs + Histograms. We have a novel design of drawing histograms on top of parallel coordinates to show the detailed statistics of a group of properties (e.g. 2-bedroom houses in the suburb) besides the original median number (Figure 3.4(e)). For example, we first show the median price of 2-bedroom houses for each suburb in parallel coordinates. When a user selects one suburb to highlight (from any views), we draw a histogram on top of parallel coordinates to show the detailed price distribution of all 2-bedroom houses in this suburb. By highlighting each suburb one by one, users could easily get the differences of property characteristics and the price distribution in each suburb.

Justification. One alternative method to visualize the multidimensional information in limited screen size is to use Heatmap [Wilkinson and Friendly, 2012]. We choose parallel coordinates since it is easy to have filtering on its own axes, and it is easy to be combined with other methods, as what we have designed, with histograms. Such design of drawing histograms on top of parallel coordinates is novel and effective to show the detailed information of the selected suburb. It is different from Hansen’s work [Hansen, 2013] which also combines parallel coordinates and histograms. In particular, Hansen has provided a histogram at each parallel coordinate axis to indicate the instances of data associated with line values, i.e., the histogram value is a summary of what you can see from the axis; while histogram in our design is an expansion of the existing median price and shows more detailed distribution information related to the axis.

3.5.3 Property-level visualization (T.3 & T.4)

Our property-level visualization (Figure 3.7) includes the Google Maps view, the multidimensional view, the Image Card view and the Word Cloud view. By default, we only show the Google Maps view. Users can select factors that they care about to visualize them in the multidimensional view, which will also enable the other views.

3.5.3.1 The Google maps view (T.3.1)

The Google Maps view provides users how properties distribute geographically and how attributes of properties vary across locations.

VIS Design 9: Glyphs on map 2. Each property is mapped as a regular polygon (Figure 3.7(b)). The color of the polygon is determined by the price of a property. The shape of a polygon is defined by bedroom numbers, i.e., a triangle and a square represent a property that has one bedroom and two bedrooms, respectively.

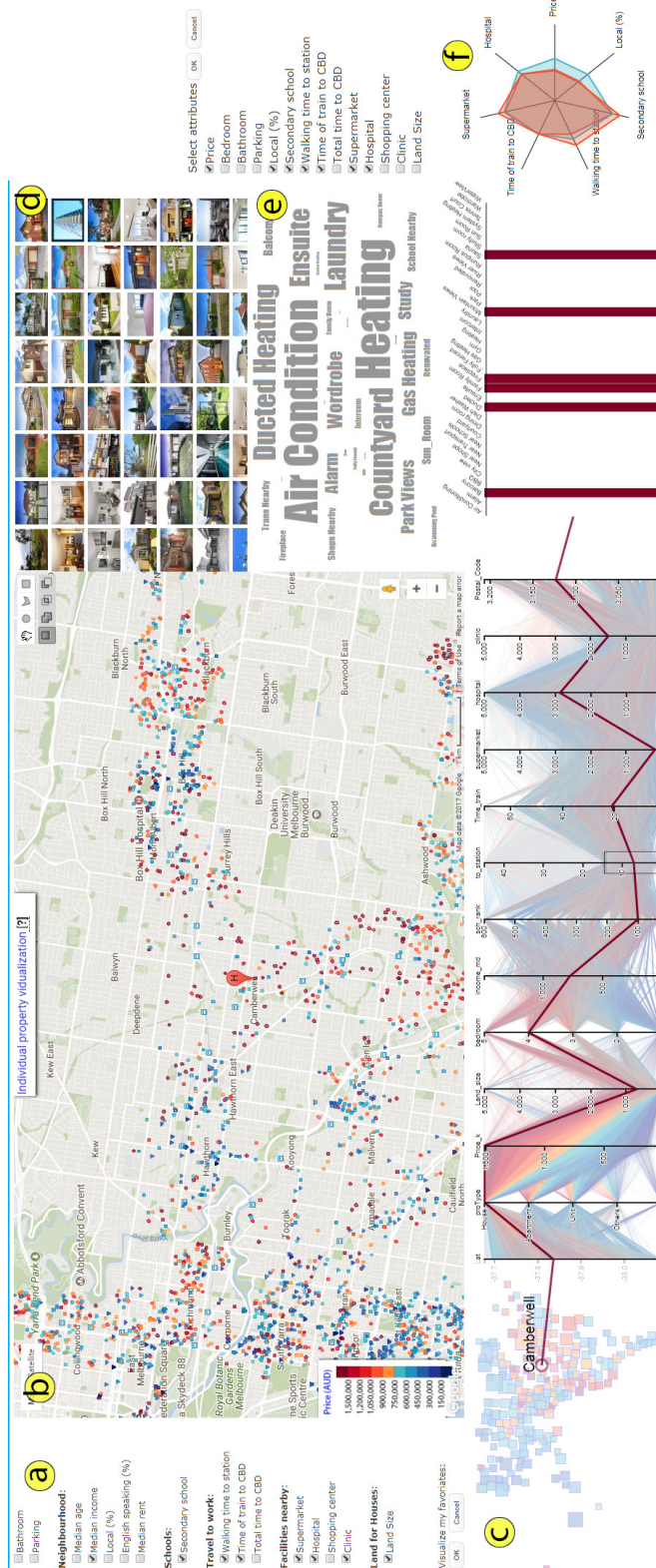


Figure 3.7: Design of property-level Visualization. (a) navigation bar, where users can select attributes to explore in the multidimensional view; (b) google maps view; (c) multidimensional view, filtered as properties within 15-min walk to the nearest train station; (d) image cloud; (e) word cloud; (f) spider chart.

We define a set of **selection tools** to allows users directly select regions that they are interested in from the Google Maps view (Figure 3.7(b) - right top corner). First, we allow users to draw polygons, circles and self-defined shapes (polygons) to select regions in the Google maps view. Secondly, inspired by Adobe Photoshop CC¹⁶, we define four selection operators as 1) *new*, 2) *add to*, 3) *intersect with* and 4) *subtract from*, so a user can select regions freely from the Google Maps view.

Justification. In this map view, we try to provide a multi-scale visual encoding scheme [Beecham et al., 2016] to visualize different levels of details with regard to different levels of user interactions. We choose the colour to represent the most important information since it is still recognizable even when the map is zoomed out. Using colour to represent price also makes sure the design is consistent with the suburb level (Section 5.2). We choose polygons instead of circles to represent each property, so we could present another dimension with the shape of polygons; also, drawing polygons is more efficient than drawing circles. We use shapes rather than the size of properties, since visual clutter is less likely to happen.

3.5.3.2 The multidimensional view (T.3.2)

The multidimensional view presents and compares all the numerical, categorical and boolean data linked to each property.

VIS Design 10: Parallel coordinates 2. As shown in Figure 3.7(c), parallel coordinates is used to display numeric and categorical attributes. It has two main functions in our system. On the one hand, it provides the comparison of different properties in multiple dimensions; by selecting and highlighting, users can also easily understand how one property is different from the others. On the other hand, it provides an interactive way for users to do filtering of properties of their own preferences based on the attributes that they are concerned about (which can be selected from the navigation bar) .

VIS Design 11: Geo-coded scatterplot. We map each property as a point in a 2D coordinate system based on its geographic location (Figure 3.7(c), left), similar to a dot map [Mackay, 1949]. This geo-coded scatter plot connects the Google Maps view and the multi-dimensional view. To avoid visual confusion, we set the first axis in parallel coordinates as *latitude*, which greatly reduces the chance of intersection of connected lines.

VIS Design 12: Coloured boolean table. As shown in the right side of Figure 3.7(c), it is a variation of Heatmap [Wilkinson and Friendly, 2012] to visualize boolean attributes (such

¹⁶<http://www.adobe.com/>

as whether the property has air conditioning or not). Each column represents a feature, and each row is corresponding to a property. The feature of a property that has a value of *yes* will be filled with the colour that is the same as that in parallel coordinates and the geo-coded scatterplot. This design is directly connected with parallel coordinates' right end.

Justification. We have directly linked three different designs to visualize multiple attributes associated with each property. First, we choose parallel coordinates out of a bunch of multidimensional visualization methods since it allows users to filter out properties based on different attributes in a straightforward way, which also helps our filtering function outperform existing commercial systems. Secondly, we use a geo-coded scatter plot directly connecting parallel coordinates. We believe that it makes users more easily link the multidimensional view with the Google Maps view. We have gained feedback from some users that, the geo-coded scatter plot makes them easier to understand the meaning of a particular polyline in parallel coordinate which represents different attributes of a property. Thirdly, with regard to visualizing boolean data, one alternation is using parallel sets [Kosara et al., 2006]. We choose the boolean table since users are more concerned with whether one property has one particular feature other than the relations among the features; and for the information of how many properties share the same features, we will show it in the word cloud view.

3.5.3.3 The image card view (T.3.3) & the word cloud view (T.3.4)

We provide the Image Card view and the Word Cloud view to display the additional image and textual information.

VIS Design 13: Image card. HomeSeeker by default displays 200 pictures of 200 properties (Figure 3.7(d)). The actual number of pictures depends on the screen resolution. Also, a vertical scroll bar is provided. After users select a property from the Image Card view or from other views, we display more pictures of the property.

VIS Design 14: Word cloud. We visualize the most common features of the properties with word cloud (Figure 3.7(e)) that the user has selected in the Google Maps view or in the multidimensional view. When users click on one feature (in the Word Cloud view), the properties that have the feature are highlighted on the map and in parallel coordinates, while the corresponding column in the coloured Boolean table is also highlighted. The Word Cloud view presents an overview of the general features of the properties that users are looking at, and help them understand a group of properties in a particular region (selected from the Google Maps view) or share the similar features (selected from the multidimensional view).

Justification. We have two additional views to visualize additional image and text information. For the images, we provide a basic function to display one image associated with each property. We have not provided any sorting function since the main purpose of this view is to provide users with some basic idea of how each property looks like. When the number of properties decreases, the image can be enlarged to increase the space utilization rate. To keep consistency with other views and avoid confusion, the colour of word cloud is set into the black.

3.5.3.4 The spider chart view (T.4)

The spider chart view provides users with a detailed comparison among a limited number of candidate properties.

VIS Design 15: Spider chart. We use a spider chart [Friendly, 1991] (Figure 3.7(f)) to compare the properties in the user’s favourite list based on the attributes that the user is concerned about. First, we have reversed the value in some axes so that the property that is placed more outside always means it is better in the corresponding dimension. Secondly, we have pre-defined the maximum and minimum value of each axis in the spider chart as the maximum and minimum value of each dimension in the whole data table. Such a design allows users to understand whether the differences between the two properties are huge or not.

Justification. The main purpose of the spider chart view is to compare the properties in the user’s favourite list. The first controversial part of the design is the duplicate information as presented in parallel coordinates. We argue that the spider chart augments the parallel coordinates to show a more clear comparison of properties in the aspects selected by users. Another problem of the spider chart is the limited number of properties it supports since we display a very limited number of properties in the spider chart. If there are more properties in the user’s favourite list, he/she can still compare those properties in parallel coordinates. The third problem is the limited number of attributes that the spider chart supports. Normally, the spider chart can display at most 12 to 16 properties, which is able to meet most users’ requirements.

3.6 System Implementation

We developed HomeSeeker¹⁷ in HTML5 and Javascript, with the library of d3.js and Google Maps JavaScript API. The database that we used is MySQL 5.6.17, and we used PHP to

¹⁷The demo is available at <http://115.146.89.158/>.

connect the database.

Particularly, in the Google Maps view, we sample properties if there are too many properties (>5000), and our index system based on the address enables that the sampled properties are in different areas. To guarantee efficiency, we use progressive rendering techniques [Turkay et al., 2017a] in some visualization designs (e.g., parallel coordinates).

3.7 Experiments and Discussion

In this section we present four case studies subsequent to the description of the experimental dataset. This is followed by domain expert interviews and a general discussion.

3.7.1 Dataset description

As discussed in Section 3.3, we have crawled the real estate data from different channels and integrated them to form a comprehensive location-aware dataset. We use Melbourne dataset to present case studies in this section.

3.7.2 Case studies

We present four typical cases of home buyers/investors can use HomeSeeker to find properties. Table 3.4 describes the budgets and requirements from the four users, and summarizes how users can use our basic tasks to finish different tasks and the involved visual designs.

3.7.2.1 Case I. From school zone to discover properties

User A had a budget of 0.8 million dollars to buy a 3-bedroom property in the Eastern Melbourne suburbs where most of her friends live. As a parent, she preferred the property within a good public secondary school and close to facilities like supermarkets, shopping centres and clinics.

She first went to profile-based region visualization (**T.1**) and selected secondary public school exploration. After discovering and comparing the school ratings, she selected those schools having a good academic performance (within the top 20% in Melbourne), including Mckinnon Secondary College, Balwyn High School, etc. (Figure 3.8(a)).

Table 3.4: User requirements of four use cases, corresponding tasks and visualization design choices.

User	Buget	Requirement	Task	T.1		T.2					T.3.&T.4							
				CM	DM1	VD2	VD3	VD4	VD5	VD6	VD7	VD8	VD9	VD10	VD11	VD12	VD13	VD14
			<div><div></div><div></div><div></div></div>															
			Sequence															
A	1,200,000	a 3-bedroom property within a good public secondary school zone and close to facilities	T.1-> T.3-> T.4	✓									✓	✓	✓	✓	✓	✓
B	900,000	a property in a suburb with more Chinese people, and might increase in value	T.2-> T.1-> T.3-> T.4	✓		✓			✓				✓	✓	✓	✓	✓	✓
C	800,000	invest a property that makes more profits	T.2-> T.3-> T.4				✓	✓	✓	✓			✓	✓	✓	✓	✓	✓
D	-	new to the city, want to explore the local real estate market	T.1-> T.2-> T.3	✓		✓	✓	✓	✓	✓								

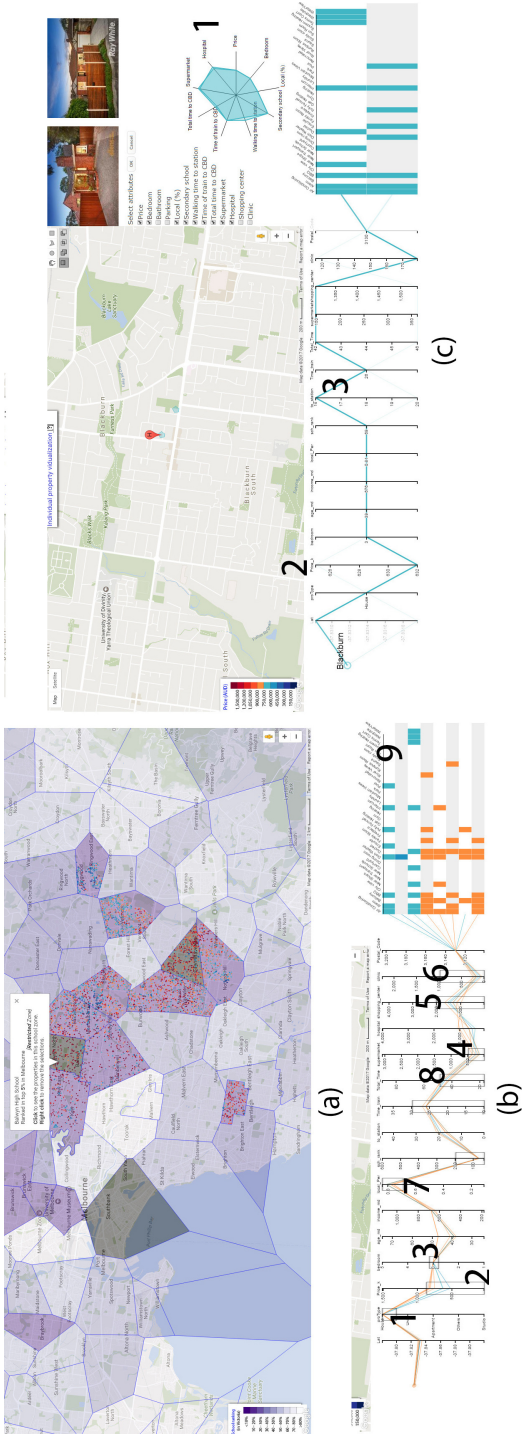


Figure 3.8: Illustration of Case I: from school zone to discover properties. (a) selecting properties within the zone of good public secondary schools; (b) selecting properties based on the user's requirements in the property-level exploration; (c) comparing candidate properties in detail.

She then clicked on property exploration (**T.3**) and found that there were too many available properties. After selecting those attributes that she was interested, she enabled the multidimensional view. She then filtered out the properties based on her original requirements (houses, below 0.8 million, 3-bedroom, within 500m to the nearest supermarket, 2km to the near shopping center and 1km to the nearest clinic (Figure 3.8(b)-[1-6]). After comparing the attributes of the remaining properties, she realized that she also wanted to live in a region with more local people (Figure 3.8(b)-7). Since her son might need to go to the city by train at weekends, the desired property needed to be near the train station as well (Figure 3.8(b)-8). After the filtering, there left ten properties all in the suburb of Blackburn. She picked two of the houses in lower prices (Figure 3.8(b)-2) and with most of the facilities nearby (Figure 3.8(b)-9).

After saving the two properties into her favourites, she compared those two houses in detail (**T.4**) (Figure 3.8(c)). She found that the two houses were very similar to each other (Figure 3.8(c)-1), except that House 1 had a slightly shorter walking time to the nearest train station from. Then, she further discovered from the multidimensional view that House 1 was 6k more expensive than House 2 (Figure 3.8(c)-2), but was closer to most of the facilities (Figure 3.8(c)-3). At last, she decided to inspect those two houses in person and had House 1 (Figure 3.8(c)-4) as a priority.

3.7.2.2 Case II. From understanding regions to discover properties

User B wanted to buy a house in a region where there were more Chinese residences so that his Chinese parents would have an easy life in the region. Since he might live there only for 3 or 4 years, he also preferred a region where the house value could increase in value over years.

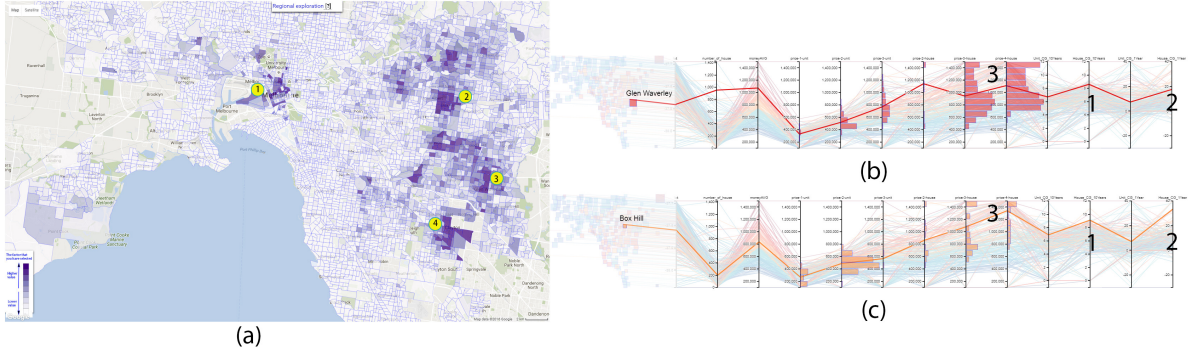


Figure 3.9: Illustration of Case II: from understanding regions to discover properties. (a) regional exploration (regions with more Chinese residents are discovered); (b & c) detailed information of two selected suburbs in Melbourne: Box Hill (b) and Glen Waverley (c).

He first went to regional exploration (**T.1**) and selected *Chinese_%* (percentage of Chinese residents) from the drop-down box (Figure 3.9(a)). He identified four regions where there were more Chinese residences, as Melbourne CBD (1), Box Hill (2), Glen Waverley (3), and Clayton (4). He then selected the median age at the regional exploration and found that Melbourne CBD and Clayton had more young people living there. Therefore, considering his parents' age, he chose Box Hill and Glen Waverley as two candidate suburbs to buy a house.

Then he went to suburb-level exploration (**T.2**), and selected Box Hill and Glen Waverley from the Google Maps view to add them to the historical view. By comparing price-related information in the two suburbs (Figure 3.9(b & c)), he noticed that 1) house price in both the two suburbs had an average capital growth of over 8% in the past ten years (Figure 3.9(b & c)-1); 2) house price in Box Hill (Figure 3.9(b)-2) increased over 30% in the past year, while the increasing rate for Glen Waverley (Figure 3.9(c)-2) was around 20%; 3) there were much more 3-/4-bedroom houses within his budget (1 million) in Glen Waverley (Figure 3.9(b)-3) comparing to Box Hill (Figure 3.9(c)-3). Considering the available properties and the possibility of price bubbles in Box Hill at the moment, he decided to first consider properties in Glen Waverley.

He double clicked the suburb of Glen Waverley and enabled to explore the individual properties in Glen Waverley. Since he did not care about the suburb boundaries, he drew rectangles with our region selection tools to add more candidate regions near Glen Waverley Community centre, and then filtered the properties and compared the properties based on his own requirement, similar to how User A discovered the individual properties and compared them (**T.3** & **T.4**), User B found three candidate properties and had a ranking among them.

3.7.2.3 Case III. Investment on properties

User C had 0.6 million dollars and wanted to invest a property from which he could make the most profits.

He went directly to suburb exploration (**T.2**), and selected those suburbs whose average capital growth of houses was above 8% in the past ten years (Figure 3.10(a)-1). Considering his budget, he further filtered the suburbs with limiting the average price below 0.7 million (Figure 3.10(a)-2). There were only four suburbs left, as St Kilda, Braybrook, Glen Huntly and Caulfield East (Figure 3.10(a)-[3-6]). After comparing the detailed information of those suburbs, he found that, there were more 3-/4-bedroom houses in Braybrook (Figure 3.10(a)-7) that were below the average price of this suburb (0.49 million, also certainly within his budget), which means that he could buy one of the properties, did some decoration if needed and then

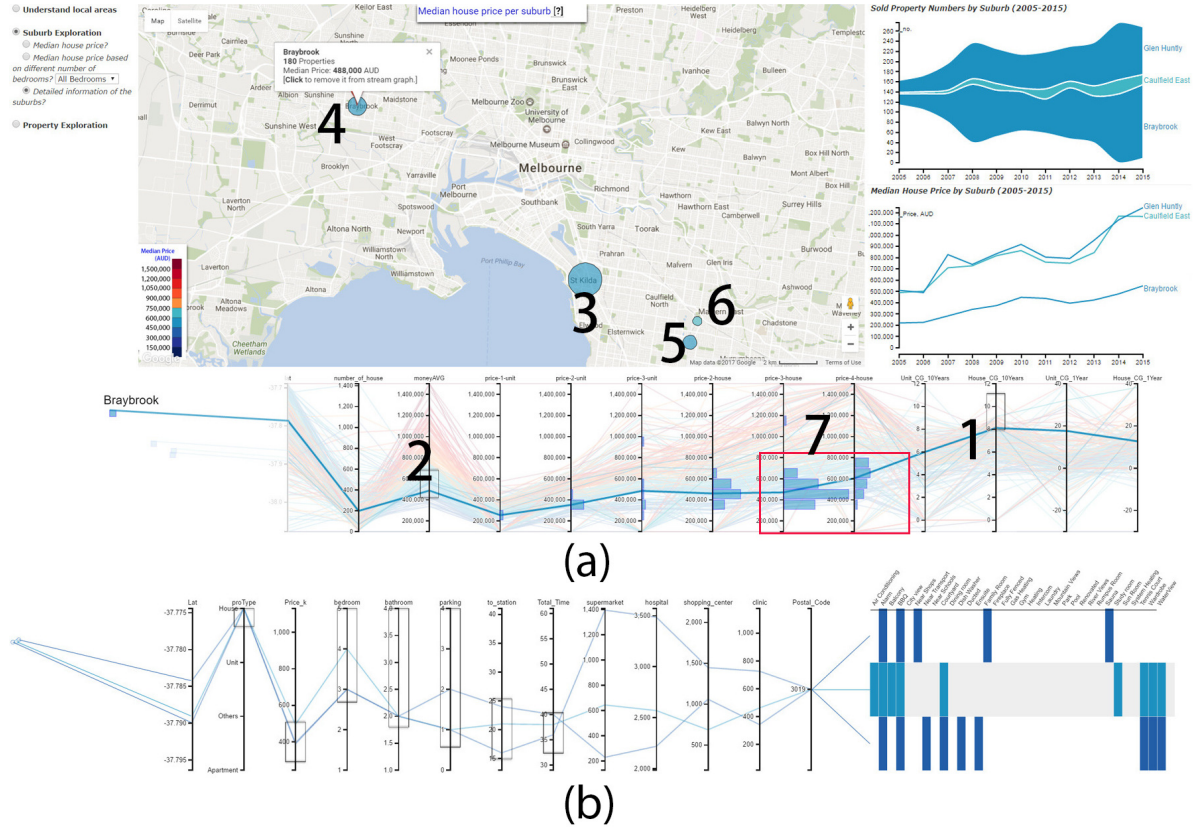


Figure 3.10: Illustration of Case III: investment on properties. (a) suburb exploration - selecting suburbs with higher house price growth; (b) comparing 3 candidate houses at the property-level exploration.

sold the property at a fair price. Therefore, he chose Braybrook as the target suburb, and clicked into the suburb to explore the individual houses.

At the individual property level (T.3 & T.4), he chose the attributes that he thought critical to resell the property. Then he filtered properties (Figure 3.10(b)) and had (3+)-bedroom houses which were below the average price in Braybrook, had at least two bathrooms and one packing, and within 40 minutes to the city by train. He had three houses left. After comparing the properties in detail, he chose House 1 as his primary target and the other two houses as candidates properties.

3.7.2.4 Case IV. Exploring the properties as beginners

User D was new to Melbourne, and wanted to explore the local real estate market, got to know different lifestyles in different regions, so that he could have a basic idea of what kind of

properties he would buy in the future.

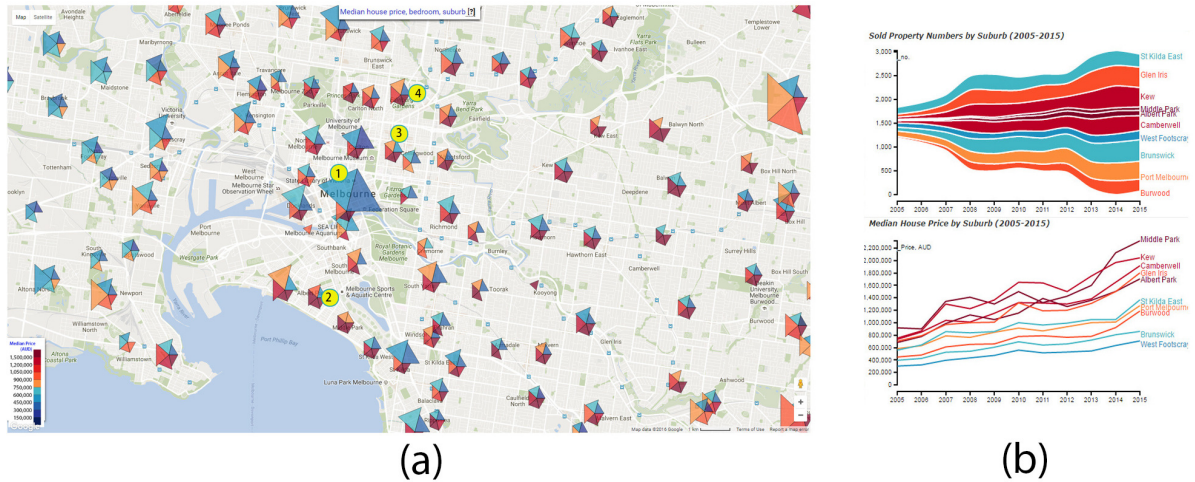


Figure 3.11: *Illustration of Case IV: exploring the properties as beginners. (a) visualizing property price based on different number of bedrooms; (b) changes of houses price and sold house property number over 10 years at different suburbs of Melbourne.*

He first selected suburb exploration (**T.2**) (Figure 3.4), and found that, in general, the prices of properties in the eastern suburbs and the southeastern suburbs were higher than that in the northern suburbs, and the prices of that in the western suburbs were the lowest; also, the price had a negative correlation with the distance from the Melbourne CBD (central business district) area. He was surprised to find that property prices in the CBD were not high, since he believed that renting a room in the CBD was quite expensive. He found the reason of this as he moved to another level to see how the prices of properties were different based on different numbers of bedrooms in different suburbs (Figure 3.11(a)). There were more 1-bedroom and 2-bedroom apartments in the CBD area (Figure 3.11(a)-1), which highly influenced the price; however, the regions around the CBD, such as Albert Park (Figure 3.11(a)-2), Fitzroy (Figure 3.11(a)-3) and Fitzroy North (Figure 3.11(a)-4) had more 3-bedroom houses, which made the price very high. The patterns showing how the prices of Melbourne properties were different along with different directions and distances to the CBD were more clear to him when only the 3-bedroom properties were selected. He also explored the individual properties, and checked the differences between the (individual) properties of the CBD and the surrounding areas.

He also noticed that median house prices were much higher than unit prices. From the Google Maps view, he selected several suburbs which locate in the western, northern, eastern and southeastern of Melbourne, and also had various distances to the CBD (Figure 3.11(b)).

Overall, the house price in 2015 had at least doubled comparing to ten years ago; while the unit price had increased too, but not significantly, especially in the latest five years. Further looking into the changes of house prices, he noticed that the price had been increased from 2005 and peaked at 2010/2011, and then decreased a little bit in the following two years and then increased at a much higher rate in the latest two years.

He then discovered the difference among regions in other aspects, such as people, public schools and facilities. in the profile-based region exploration (**T.1**). At last, he clicked on property-level exploration (Figure 3.7) and defined several kinds of filtering settings to see what kinds of properties he could afford in different regions (**T.3**).

3.7.3 Domain expert feedback

To attract users, we demonstrated our demo system at the 27th Australian Database Conference¹⁸ (for half a day) and the 2017 conference of Beyond Research - Pathway to Impact¹⁹ (for two days). We also advertised our system on social media with an introduction video and demo links. Besides general users and researchers from different fields, we particularly got the attention from two real estate agents, two researchers in Business IT (related to real estate) and a professor in urban science, which are all considered as domain experts.

One of the real estate agents commented on our system (based on an early version of implementation), saying that *[The System is] Incredible, what a great translation of property buyer needs. It's refreshing to see it built from authenticity, instead of commercialism.* The agent contacted us from social media and met us in person. The same with how we discussed with other domain experts, we introduced the system, asked them to try the system by themselves, and then sought feedback from them.

Most of the domain experts are fascinated about how we collected the data from different sources and integrated them together. One of the real estate agents commented that our tool could be very useful for him to recommend properties to home buyers, and help them illustrate why the property is worth to buy. He also suggested us to include the annual growth information which investors might be more interested (for example, they might search for the suburbs that have an annual growth larger than 10% in the last five years). Though our system mainly focuses on home buyers, we have included such information in our later implementation to suit for more user types.

¹⁸<https://adc2016.cse.unsw.edu.au/>

¹⁹<http://beyondresearchconference.com.au/>

3.7.4 Discussion

Based on the case studies, we have demonstrated that our system can be beneficial to both home buyers and investors; and users with different levels of knowledge (on the local real-estate market) can jump into a certain level of tasks to explore the local properties. Although users still need to inspect the properties before they make a final decision, our system provides rich sources for them to find candidate properties and understand the properties before the inspection. We have augmented existing commercial systems in the following aspects. First, based on a unique location-centred dataset that we provide, all the data (related to different profiles) and services (exploration in the suburb level and individual property level) in our system are interconnected. Secondly, we have provided a way to help users understand the local real estate market, which also helps them further understand what kinds of properties that they prefer and which they can afford. Thirdly, we allow users to explore the properties, find properties based on their requirements in a more interesting and effective way, since our system has provided instant feedback after they interact with the system. Last, we allow users to visually compare suburbs and properties, which the current commercial systems fail to do.

One of the limitations is related to the scalability. We have sampled the properties on top of the Google Maps view when there are too many (i.e., more than 5,000) properties at the selected region; and we have also used progressive rendering on parallel coordinates. Users have commented that the progressive rendering in a way makes them feel confused, and the massive lines in the parallel coordinates fail to provide much information. In our second and third research questions, we will discover more about the aggregation based on both the map view and the multidimensional view.

Another consideration is whether our framework works well in other cities/countries besides Melbourne/Australia. So far, we have also collected data from other major cities in Australia, such as Sydney, Brisbane, Perth and Adelaide. Since the data profiles that we have defined (Section 3.3.2) are general enough to cover all location information related to a real estate market, and new information can always be treated as one instance of one type of the profiles. Our framework works well with other cities, except that only some details in the process of data collecting need to be modified. For example, in Melbourne, most commuters travel by train or tram; while in Sydney, it could be by train or ferry. Certainly, there are different considerations across countries, we believe that our framework can inspire researchers from other countries to work on their real estate data.

3.8 Summary

In this chapter, we explored visual analytics of geo-related multidimensional data using the real estate domain as an example. We designed and developed HomeSeeker which augments existing commercial systems to help users understand the local real estate market, find preferred properties based on their personal preferences, and compare properties from the aspects that they are concerned about. We provided a location-centred real estate dataset, which is exclusive in this chapter. We presented a systematic visualization design study following with justifications. Case studies based on real-world datasets demonstrated the usefulness of our system, and users with different levels of knowledge on the market and different requirements (either general home buyers or investors) can benefit from our visual analytics system.

Chapter 4

ConcaveCubes: Supporting Large-scale Geographic Visualization

“Maps encourage boldness. They’re like cryptic love letters. They make anything seem possible.”

— Mark Jenkins

In this chapter, we study the problem of supporting effective visualization/visual analytics for large-scale geo-related multidimensional data. From an extensive literature study, we find that the existing solutions suffer from at least one of the drawbacks below: (i) loss of interesting structures/outliers due to sampling; (ii) supporting heatmaps only, which provides limited information; and (iii) no notion of real-world geography semantics (e.g., country, state, city) is captured in the visualization result as well as the underlying index. Therefore, we propose ConcaveCubes, a cluster-based data cube to support interactive visualization of large-scale multidimensional urban data. Specifically, we devise an appropriate visualization abstraction and visualization design based on clusters. We propose a novel concave hull construction method to support boundary based cluster map visualization, where real-world geographical semantics are well preserved. Instead of calculating the clusters on demand, ConcaveCubes (re)uses existing calculation and visualization results to efficiently support different kinds of user interactions. We conduct extensive experiments using real-world datasets and show the efficiency and effectiveness of ConcaveCubes by comparing with the state-of-the-art cube-based solutions.

4.1 Introduction

In the previous chapter, 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 individual data points on a map, suffer not only from large memory consumption, but also have perceptual and interactive scalability problems [Liu et al., 2013]. In particular, users’ perceptual and cognitive capacities are overwhelmed by data over-plotting, and users’ interaction with large-scale datasets can easily lead to high latency.

A straightforward approach is to employ sampling methods [Drosou and Pitoura, 2012] to reduce the data to be displayed. However, they possibly elide interesting structures or outliers, thereby preventing users from querying the data to cater to their own preference in the data exploration stage. Therefore, one recent research trend is to design data cubes and pre-compute possible data aggregations to support efficient visualization, such as imMens [Liu et al., 2013], Nanocubes [Lins et al., 2013], and Hashedcubes [Pahins et al., 2017]. As a result, they enable fast query processing for interactive visualizations of large and multidimensional data.

While these data cube structures are effective for solving the scalability problem, we observe that they only support heatmaps [Bojko, 2009] based on binned aggregation in terms of visualizing geographical features. Although Liu et al. [Liu et al., 2013] have shown that heatmaps have advantages vis-à-vis sampled symbol maps (e.g., heatmaps can show the overall geographical distribution while sampled symbol maps cannot), the information that heatmaps can provide is often very limited (Example 4.1 in Section 4.3.4).

To deal with the limitation of heatmaps, we describe a design space for geographical visualization based on the result of data clustering (rather than relational aggregation). Following Tobler’s First Law of Geography - *Everything is related to everything else, but near things are more related than distant things* [Tobler, 1970], we first cluster geographical data points that are close in distance (assuming Euclidean distance in contrast to other distances such as the Hausdorff distance [Nutanong et al., 2011]) and share similar features, and then represent each cluster with a geographical bubble on top of the map. It is called bubble-based cluster maps as shown in Figure 4.2(a). Cluster maps, which is hard to support by existing data cubes (e.g., imMens [Liu et al., 2013], Nanocubes [Lins et al., 2013], Hashedcubes [Pahins et al., 2017]), can provide additional information compared to heatmaps or binned plots. We compare existing

map designs and discuss why we chose cluster maps, rather than heatmaps or binned plots, etc. in Section 4.3.4.

One problem of the existing cluster map design is that each cluster is often visualized based on bubbles, with the size and the colour (sometimes the shape as well) of each bubble representing different features of the data. Although bubble-based cluster maps can present information that heatmaps and binned plots cannot, the locations of properties in each cluster could not be precisely represented, i.e., properties actually inside of the circle might not belong to the cluster (Example 4.2 in Section 4.3.4). To solve such a problem, we provide a new map design to represent the geographical boundary of each cluster, i.e., boundary-based cluster maps (similar but the same as choropleth maps [Brewer et al., 1997]).

A key question arising for the boundary-based cluster maps is “what is an appropriate choice of the polygons used to present the boundary of clusters?”. To answer this question, we reviewed existing methods (e.g., KNN-based method [Moreira and Santos, 2007], χ -shapes [Duckham et al., 2008] and α -Concave Hull [Asaeedi et al., 2017]) that generate different polygons (hulls) based on the included geographical points. However, all of these methods suffer from at least one of the following problems: (i) large empty areas inside the hull, (ii) too complex shapes, and (iii) overlaps among the polygons representing the different clusters (e.g., Figure 4.3(d)). The problem is explained in greater detail in Section 4.4, with a summary of envisioned features of an appropriate choice in Table 4.1.

We propose a new concave hull algorithm, which generates a polygon (hull) to present each cluster based on the geographical locations of all data points in the cluster. In particular, our algorithm is based on a global Delaunay triangulation [Samet, 2006], followed by a separation of points based on the different clusters; thus it is able to avoid all of the three aforementioned problems.

Following our proposed cluster-based visualization design, a subsequent challenge is: *what kind of a data structure is efficient to support such visualization design and user interactions on large-scale data?* Accordingly, we propose a tree-structured index called ConcaveCubes, which organizes the pre-computed hierarchical clustering result in data cubes. Compared to state-of-the-art visualization-driven cube structures, (i) ConcaveCubes supports cluster maps which existing structures cannot; (ii) ConcaveCubes exploits real-world geographic semantics (e.g., country, state, city) rather than using grid-based aggregations; (iii) instead of calculating the clusters on demand, ConcaveCubes utilizes existing calculation and visualization results to efficiently support different kinds of user interactions, such as zooming & panning, filtering, and granularity control.

To summarize, we make the following contributions in this chapter.

1. We present a cluster-based visualization abstraction of geographical data (Section 4.3).
2. We propose an algorithm to efficiently generate concave hulls that include geographical points in each cluster as compactly as possible. As a result, it is able to avoid large empty areas inside the hull, complex shapes and overlaps among different clusters (Section 4.4).
3. We propose a cluster-based data cube (ConcaveCubes) to efficiently support interactive response to users' visualized exploration on large-scale geographic multidimensional data (Section 4.5).
4. We conduct extensive experiments using real-world datasets, and compare ConcaveCubes with state-of-the-art cube-based data structures to verify the efficiency and effectiveness of ConcaveCubes (Section 4.7).

4.2 Related Work

There are two major types of methods to address the scalability issue of visualizing large-scale datasets. On the one hand, visualization techniques address the perceptual and interactive scalability problem at the visualization end, e.g., pixel-based visualization [Keim, 1996], alpha blending [Jerding and Stasko, 1998], spatial displacement methods [Trutschl et al., 2003], and dimension re-ordering [Peng et al., 2004] for parallel coordinates; however, they still need to scan each data item, which makes them difficult to scale. On the other hand, researchers have proposed various data reduction methods which reduce the size of the data before the data enters the visual rendering process. For example, sampling [Drosou and Pitoura, 2012] & filtering [Ahlberg and Shneiderman, 1994] are widely used to select a smaller subset of data before applying standard visualization techniques; however, they fail to provide an overview of the data distribution [Lins et al., 2013].

Data cube aggregation methods. As one of the data reduction methods, data cubes [Gray et al., 1997] have been intensively studied in the area of data warehouses. In this technique, aggregation results of the raw data are pre-computed on predefined dimensions to support data exploration. However, a complete data cube is often too large to fit in memory and query in real-time. One recent research direction is to design visualization-constrained data cubes that are constrained by the number of pixels used in visualization [Keim, 1996]. Different methods have been proposed to reduce the size of the data cubes for information visualization based on

binned aggregation [Carr et al., 1987], which conveys both global patterns and local features while enabling different levels of resolution by changing the bin size. Visualization researchers [Liu et al., 2013, Lins et al., 2013] suggested a design principle that *scalability should be limited by the chosen resolution of the visualization data, not the number of records*. Based on this principle, Liu et al. [2013] proposed imMens, which decomposes the full cube into sub-cubes to minimize data memory usage and thus reduce the query latency. However, imMens can only support brushing and linking [Keim, 2002] scenarios with at most four dimensions. Nonocubes [Lins et al., 2013], on the other hand, supports an arbitrary number of dimensions; but it might suffer from query latency and has a high memory cost, since it uses a large amount of binned aggregation. GaussianCubes [Wang et al., 2017b] and TopKube [Miranda et al., 2018] extend the idea of Nanocubes by supporting more types of queries, but still suffer from the high query latency as the number of dimensions increases. Hashedcubes [Pahins et al., 2017] sorts data in advance based on judiciously selected pivots, thereby on-the-fly aggregation computation can be supported and the storage cost is reduced significantly.

Our proposed ConcaveCubes is in line with the data cube methods. As compared to state-of-the-art imMens [Liu et al., 2013], Nanocubes [Lins et al., 2013], and Hashedcubes [Pahins et al., 2017], our ConcaveCubes distinguishes from them in three aspects. First, instead of storing individual geographic points, ConcaveCubes first clusters those data items that are close in distance and share similar features, and then builds indexes based on the clusters, i.e., cluster-based data cubes. Therefore, ConcaveCubes supports the visualization of clusters on top of the map, while existing methods only support heatmaps. Second, ConcaveCubes exploits real-world geographic semantics (e.g., country, state, city) rather than using grid-based aggregations. Third, instead of calculating everything on demand, ConcaveCubes utilizes existing calculation and visualization results to efficiently support different kinds of user interactions, such as zooming, panning and filtering.

4.3 Cluster-based Visualization Abstraction

Figure 4.1 presents the structure of this section, and how it is linked with Section 4.4 and Section 4.5.

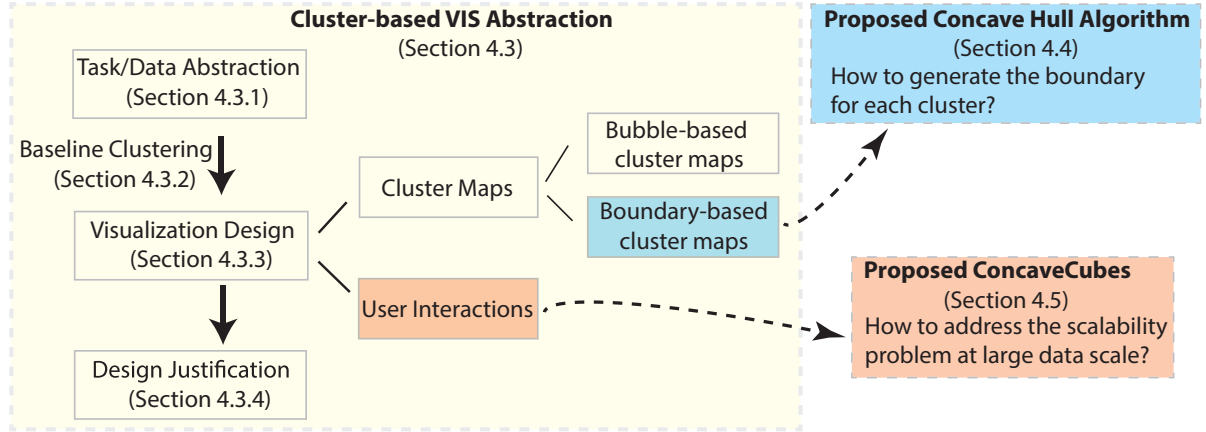


Figure 4.1: An overview of our solution in Sections 4.3, 4.4 & 4.5.

4.3.1 Data and task abstraction

For ease of explanation, we use the real estate dataset (from the last chapter) as an example of geo-related multidimensional data to explain this section. Recall Section 3.3, the real estate dataset include five profiles, i.e., basic profile, census profile, education profile, facility profile, and transportation profile (detailed in the previous chapter (Chapter 3). Each property is associated with its geo-spatial information (latitude and longitude), categorical dimensions (e.g., property type), and numerical dimensions (e.g., price).

Following Shneiderman’s visual information-seeking mantra [Shneiderman, 1996] to define our tasks, i.e., *overview first, zoom and filter, then details-on-demand*, we illustrate the abstraction of tasks based on a series of real-life questions: (1) *What is the difference of the property prices in different suburbs of Victoria (overview)?* (2) *What if the user only concerns about 3-bedroom houses (filtering)?* (3) *What is the difference between the houses in blocks of the same suburb (zooming & granularity control)?* (4) *Based on the user-selected attributes, how do the houses in a block differ from the rest of the houses in the same suburb, or the rest of the houses in that entire state (details-on-demand, highlighting & linking)?*

4.3.2 A baseline solution

Before describing our cluster-based visualization design, we introduce a baseline clustering solution. Here, we first subdivide different dimensions into four main types based on (i) whether they are directly linked to locations, and (ii) whether they directly affect the clustering result.

- **Geo-dependent dimensions:** including geographical locations represented by latitude

and longitude, and geo-semantics such as *state*, *suburb*. These dimensions are directly linked to geography, and will directly affect the clustering result.

- **Additional geo-dependent measures:** e.g., the distance to the nearest train station, the median income of the region. These dimensions are directly related to geography. However, once a group of real estate properties are close in distance, they share similar values of the other dimensions. Therefore, it is not necessary to consider those dimensions in the clustering process.
- **Geo-independent dimensions:** e.g., price, property type, bedroom number. They are major factors in the clustering process which are not directly linked to the geo-location.
- **Additional geo-independent measures:** e.g., number of parking spots, whether containing air conditioning. Users may want to check these factors for individual houses but they are not important enough to be considered in the clustering process.

Now we present the steps of a straightforward method to cluster the properties that are close in distance while also sharing similar features: **(i)** We group properties based on geo-independent dimensions, e.g., group all 3-bedroom houses with price between 0.9-1million. **(ii)** We further group the properties based on one level of geo-semantics (geo-dependent dimensions), e.g., all 3-bedroom houses in the same suburb will be grouped together. **(iii)** For each group, we apply DBScan [Ester et al., 1996] to divide the properties into different clusters based on the geo-location, e.g., the properties within the same group after step 2 and close in distance (e.g., <150m) will be in the same cluster.

4.3.3 Visualization design

To support the visualization tasks (Section 4.3.1), we design two map-based views and a linked multidimensional view. In the map-based views, we visualize the clusters on top of the map. When a cluster is highlighted in the map-based view, a linked multidimensional view presents the detailed information of the cluster based on user-selected measures. Justification of using cluster maps instead of other map designs (e.g., heatmaps that existing visualization-based data cubes support) will be presented in Section 4.3.4.

Map-based view 1: bubble-based cluster maps. A straightforward visual encoding to visualize clusters is to use bubble maps. Comparing to heatmaps that Nanocubes [Lins et al., 2013] and Hashedcubes [Pahins et al., 2017] support, bubble maps have at least one more retina variable (i.e., the size of the bubble) to encode an extra dimension (another variable is

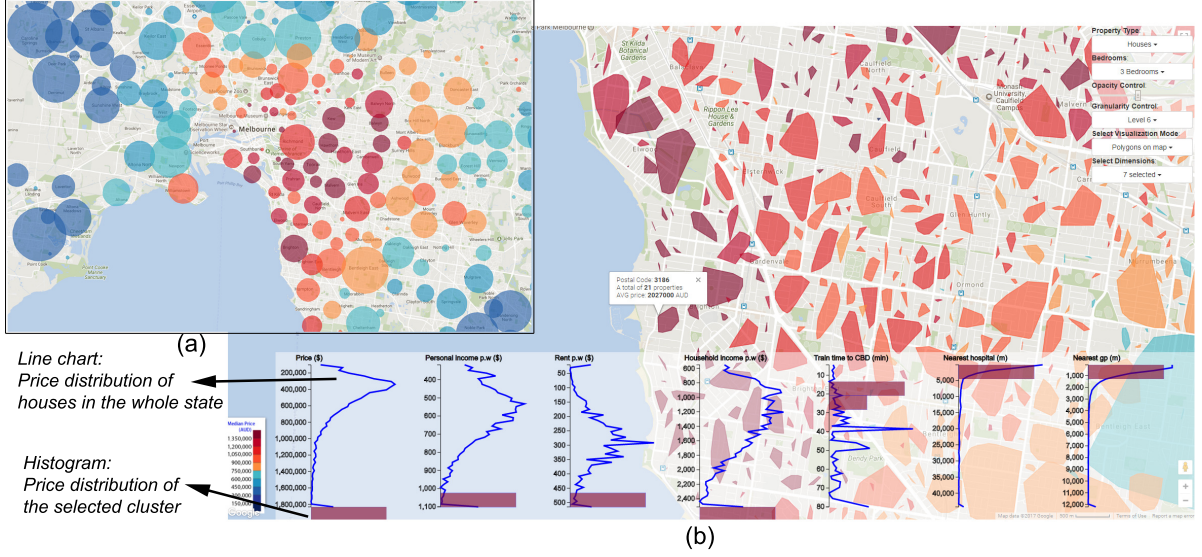


Figure 4.2: Illustration of our visualization design. (a) map view 1: bubble-based cluster map; (b) map view 2: boundary-based cluster map, and the multidimensional view with a cluster highlighted (properties in the selected cluster is compared to that of the entire state based on 7 user-selected measures (sold price, personal income, rent price, household income, distance to CBD, nearest hospital and nearest doctor).

the shape). An example is shown in Figure 4.2(a), where the colour of each bubble represents the average price of properties in each cluster, and the size of the bubble represents the total number of properties in the cluster. We also use alpha blending [Jerding and Stasko, 1998] to reduce visual cluttering for intersecting bubbles.

Map-based view 2: boundary-based cluster maps. As boundaries are important measurements of the geographic information, our second map-based view draws concave hull (a polygon that covers a group of points) [Moreira and Santos, 2007] on top of the map to illustrate positions of the properties in each cluster (Figure 4.2(b)). How to generate a concave hull based on a group of geographical points is presented in Section 4.4. The colour of the concave hull is based on an aggregated value of user-defined dimensions for all the properties in each cluster. This is done as follows: suppose that a user selects m dimensions, and there are n clusters. Let $\tilde{x}_{i,j}$ be the median value of all properties in cluster c_j ($j \in [0, n-1]$) at dimension i ($i \in [0, m-1]$). The aggregated value of a cluster c_j is computed as a weighted sum:

$$aggr_{c_j} = \sum_{i=0}^{m-1} w_i \cdot N(\tilde{x}_{i,j}) / \sum_{i=0}^{m-1} w_i \quad (4.1)$$

where, $N(\tilde{x}_i)$ corresponds to the normalized value of $\tilde{x}_{i,j}$ between $[0, 1]$ and computed as:

$N(\tilde{x}_{i,j}) = (\tilde{x}_{i,j} - \min_j \tilde{x}_{i,j}) / (\max_j \tilde{x}_{i,j} - \min_j \tilde{x}_{i,j})$, where w_i is the weight of the i -th dimension and is defined by the user. Various strategies [Wang et al., 2017a] could be used to assign the weight after the user gives the importance rank of all dimensions, including Fixed Weights (FW), Randomly Assigned Weights (RAW), Uniformly Distributed Weights (UDW), etc. In our implementation, we adopt one of the RAW methods named Random Weighted Genetic Algorithm (RWGA) [Murata and Ishibuchi, 1995]. In particular, when the user selects only one dimension (e.g., price), the colour is mapped based on the selected dimension.

Multidimensional view: line charts + histograms. We design a linked multidimensional view to visualize the statistical information of a selected (highlighted) cluster and compare it with others. For example, in Figure 4.2(b), after the user selects a cluster in Brighton, the statistical information of this cluster based on seven user-defined dimensions is shown as seven histograms in the multidimensional view; also, the information in the entire state (Victoria) on those seven dimensions are shown as seven line charts for comparison.

Interaction design. Based on our task abstraction, we support the following user interactions:

- **Zooming & panning:** we support zooming & panning on maps.
- **Filtering:** we support filtering data values on each dimension. The cluster will be recalculated as a result of filtering. Filtering is performed using the selection panel shown in the top right corner of Figure 4.2(b) that consists of sliders and drop-down menus.
- **Granularity control.** We support map-based visualization in multi-granularity scales. In our implementation, after users zoom in/out on top of the map, the granularity level will also be changed. However, users can also change to a finer or coarser level of granularity by selecting from the selection panel while staying at the same zooming level.
- **Other interactions,** such as highlighting & linking (i.e., selecting and highlighting a cluster on the map and linking the information in the multidimensional view).

4.3.4 Design justification: why cluster-based visualization?

There are various map designs to visualize geographical data, including symbol maps, heatmaps, bubble maps, cluster maps, etc. We observe that, existing visualization-driven data cube structures (e.g., imMens [Liu et al., 2013], Nanocubes [Lins et al., 2013], Hashedcubes [Pahins et al., 2017]) only support heatmaps or dot maps based on binned aggregation. Although Liu et al. [Liu et al., 2013] have shown that heatmaps have advantages in comparison to sampled symbol maps (e.g., heatmaps can show the overall geographical distribution while sampled symbol

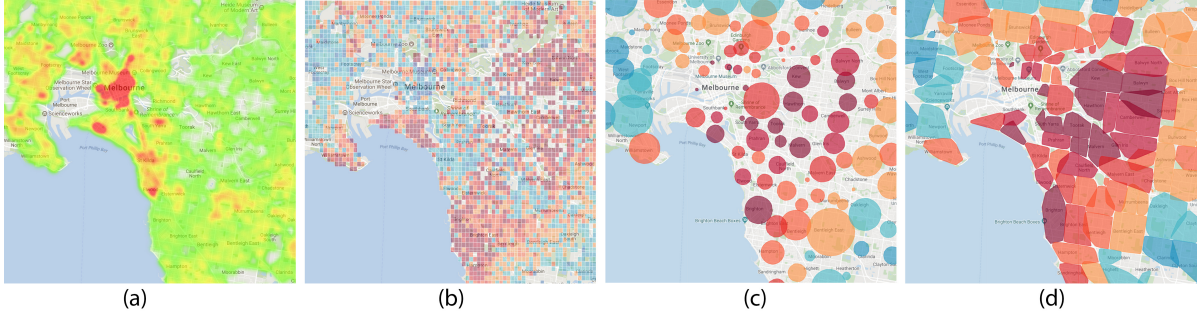


Figure 4.3: A comparison of four geographic visualizations based on the real estate data in Melbourne. (a) Heatmaps and (b) binned plots, which *imMens* [Liu et al., 2013], *Nanocubes* [Lins et al., 2013] and *Hashedcubes* [Pahins et al., 2017] support; (c) bubble-based cluster maps and (d) boundary-based cluster maps that our proposed *ConcaveCubes* supports.

maps cannot), the information that heatmaps can provide is still very limited, as illustrated in Example 4.1.

Example 4.1. Figures 4.3(a) and 4.3(b) show a visualization of Melbourne’s real estate data using heatmap and binned plots (a variation of heatmaps), respectively. Heatmap, which is the only form of visualization supported by existing cube structures, can only present the density of real estates in different regions (by colour). Binned plots, which can be supported by the existing methods by slight adjustments, can provide price distribution (by colour) (Figure 4.3(b)); however, the plots are not easy to interpret as each binned block does not have real-life semantic meanings. For example, the real estates in a binned block may not be in the same suburb and/or may have different key features (e.g., they may belong to different school zones).

On the other hand, cluster maps, which are difficult to support by existing data cubes, provide more information (with semantic meanings) in comparison to heatmaps or binned plots: Example 4.2.

Example 4.2. We visualize Melbourne’s real estate data using bubble-based cluster maps (Figure 4.3(c)). Each bubble represents a group of real estate properties which are clustered based on both the geo-location and two other features (corresponding to suburbs & secondary school zones). The colour and size of each bubble represent the median price and the total number of properties in each cluster, respectively. Compared to heatmaps and binned plots, bubble-based cluster maps (i) represent one more dimension, and (ii) have a real semantic meaning of the visual mark - each bubble represents a group of houses that are in the same suburb and school zone as well as close in distance.

Although traditional bubble-based cluster maps can present information that heatmaps and binned plots cannot, the locations of properties in each cluster are not precisely represented, i.e., properties actually inside a circle might not belong to the cluster. Our proposed boundary-based cluster maps aims to solve this problem, as shown in Example 4.3.

Example 4.3. *We visualize the same data using boundary-based cluster maps (Figure 4.3(d)). Each cluster is represented by a colour-encoded hull (boundary of the cluster). Compared to the heatmaps or binned plots in Figure 4.3(a & b), boundary-based cluster maps present geographical information with semantic meanings. Comparing to our baseline bubble-based cluster maps which present one more measure using the size of each bubble, the boundary-based cluster maps present more precise location information.*

4.4 Algorithm for Generating Boundary-based Cluster Maps

In this section, with the clusters as the input, we present our algorithm to generate the boundary of each cluster that best represents the group of geographical points that are physically located in it. In particular, we first outline the desired features (from the perspective of visual perception) of a boundary generation algorithm. Next, we show that the concave hull based representation is the most suitable choice. Third, we propose a novel concave hull construction algorithm that meets all of these desired features. Finally, we compare the visualization result and the time complexity of our proposed algorithm with the existing methods.

4.4.1 An outline of desired features

We envision four features for an ideal cluster map: (1) no overlap between the visualization of any two clusters, (2) complex shapes should be minimized to enhance user’s cognitive capability, (3) empty areas of each hull [Galton and Duckham, 2006] (i.e., the area inside a hull that does not contain any real estate properties) should be minimized, (4) parameter tuning for boundary generation should be avoided.

Generating hulls (polygons) to represent the geographical boundary of each cluster is one of the approaches to achieve these features. We first review existing hull generation algorithms. Ebert et al. [2015] classify hulls into five different categories: (i) *rectangular hull*, (ii) *orthogonal convex hull*, (iii) *convex hull*, (iv) *concave hull*, and (v) *concave hull for several groups*. Since our target is to represent each cluster with a single polygon, *concave hull for several groups* (representing one cluster) is not appropriate in our case. Among the other categories, the

concave hull is recognized as presenting the most precise shape recognition of an area [Ebert et al., 2015]. Therefore, we opt for a concave hull based representation for the positions of properties in each cluster.

Table 4.1 shows a comprehensive list of limitations of seven major hull construction algorithms. Although the convex hull algorithm does not generate complex shapes, it usually results in large empty areas and will easily cause overlaps. While concave hulls generated by χ -shapes [Duckham et al., 2008] and the KNN-based method [Moreira and Santos, 2007] can reduce empty areas, they either have overlaps or complex shapes. The Voronoi-based [Alani et al., 2001] and the triangulation-based [Arampatzis et al., 2006] concave hull method fully partition the whole space, thus a huge empty area can be generated inside each hull.

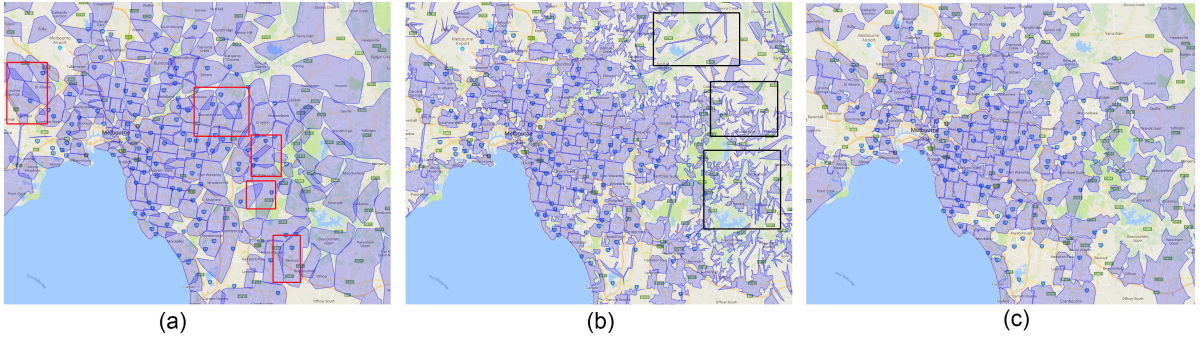


Figure 4.4: The result of an existing concave hull algorithm [Park and Oh, 2012b] (representing all the concave hull methods in Table 4.1) [Red - overlaps, black - complex shapes] and the result of using our proposed algorithm: (a) using the method in [Park and Oh, 2012b] with a threshold of 100m; (b) using the method in [Park and Oh, 2012b] with a threshold of 800m; (c) using our proposed concave hull construction approach.

Moreover, concave hulls are usually not unique, i.e., the constructed concave hulls can be different in different algorithms and/or with different parameters. Although different algorithms have been proposed to generate a concave hull, there is no standardization on the effectiveness of the concave hull, i.e., determining which concave hull algorithm generates a region that best represents a group of points is inconclusive [Galton and Duckham, 2006]. Even with the same algorithm, determining the parameter settings is a difficult problem.

Figures 4.4(a & b) illustrate the common limitations of the existing Concave hull methods with real-life data. We have implemented two existing methods [Park and Oh, 2012b, Moreira and Santos, 2007] and applied them to our real estate dataset, where each cluster includes properties in a postal area. The algorithm by Park and Oh [2012b] first constructs a convex

Table 4.1: *A comparison between our proposed algorithm and 7 existing hull construction algorithms.*

		Overlaps	Complex Shape	Empty Area	Parameter Needed	Complexity
Convex Hull	Graham Scan [Jarvis, 1973]	Yes	No	Yes	No	$O(n \log n)$
	χ -shapes [Duckham et al., 2008]	Yes	Yes	No	Yes	$O(n \log n)$
Convave Hull	KNN-based method [Moreira and Santos, 2007]	Yes	Acceptable	No	Yes	$O(n^3)$
	α -Concave Hull [Asaeedi et al., 2017]	Yes	Yes	No	Yes	$O(nh \log n)$
	Park and Oh [2012b]	Yes	Yes	No	Yes	$O(nh \log n)$
Others	Alani et al. [2001]	No	No	Yes	No	$O(n \log n)$
	Arampatzis et al. [2006]	No	No	Yes	No	$O(n \log n)$
Our proposed concave hull algorithm		No	No	No	No	$O(n \ln n)$

hull, and then execute a “digging” process (progressively replace an outer edge with two close inner edges) to get a concave hull. The only parameter here is a threshold that controls when to terminate. We visualize the results on Google Maps, and observe that: a larger threshold as shown in Figure 4.4(a), may cause lots of overlaps in dense areas (e.g., CBD areas which have more properties); while a smaller threshold will result in very complex shapes in the less dense area and users may fail to recognize the shapes (Figure 4.4(b)). We also implemented the KNN-based method [Moreira and Santos, 2007], which does not generate too complex shapes, but cause significant overlaps similar to Figure 4.4(a).

4.4.2 Our concave hull construction approach

To address the limitation of existing concave hull generation methods, we propose a novel algorithm to meet all the four envisioned features. The pseudocode of our proposed approach is presented in Algorithm 4.1 and illustrated in Figure 4.5. The input of the algorithm is a set of geographical points, P . Each point $p \in P$ has three attributes, where latitude and longitude jointly represent its geographical location, and cluster $p.cluster$ indicates the cluster in which the point belongs. The output of the algorithm is a set of concave hulls. Each concave hull corresponds to a unique cluster ID, and its boundary should be a Jordan curve. The Jordan

Algorithm 4.1: CONCAVE_HULL (P)

```

1 Input: A set  $P$  of geographical points, each  $p \in P$  has attributes {latitude, longitude, cluster}.
2 Output: A list of concave hull boundaries for each cluster.
3 DELAUNEY_TRIANGULATION( $P$ )
4  $TE \leftarrow$  Set of resulting triangulation edges
5 for each edge  $e = \langle p_1, p_2 \rangle \in TE$  do
6   | if  $p_1.\text{cluster} \neq p_2.\text{cluster}$  then
7   |   | delete  $e$  from  $TE$ 
8 Delete all inner edges from  $TE$ 
9 while  $\exists p : \text{degree}(p) = 1$  do
10  |  $p_1 \leftarrow$  the node that  $p$  connects;
11  | if  $\text{degree}(p_1) = 1$  then
12  |   | delete  $\langle p, p_1 \rangle$  from  $TE$ ;
13 while  $\exists p : \exists \angle p'pp''$  where  $p'$  and  $p''$  do not form a loop do
14  |  $A_p \leftarrow$  All outer angles created at  $p$ ;
15  |  $\angle p'pp'' \leftarrow$  The smallest angle in  $A_p$ ;
16  | if  $\langle p, p' \rangle$  is not an exposed line then
17  |   | delete  $\langle p, p' \rangle$  from  $TE$ ;
18  | if  $\langle p, p'' \rangle$  is not an exposed line then
19  |   | delete  $\langle p, p'' \rangle$  from  $TE$ ;
20  | add  $\langle p', p'' \rangle$  to  $TE$ ;
21  $c\_edges \leftarrow$  BOUNDARY_EDGES_GROUP_BY_CLUSTERS( $TE$ )
22 RETURN  $c\_edges$ 

```

curve restriction is commonly used in all the methods in Table 4.1. The reason for adopting this restriction in our cases is quite straightforward: the boundary for each cluster should be a plane simple closed curve that divides the whole space into exactly two regions - the one inside the cluster, and the one outside it.

At a high level, the steps of the algorithm are: (i) Generate a Delaunay triangulation for P . (ii) To reduce overlaps, separate clusters by deleting edges that connect two nodes in different clusters. (iii) Remove the inner edges to get the outer boundary of the cluster, which ensures no additional empty area is included. (iv) If a non-Jordan curve is generated in this process, then apply our proposed novel edge replacement technique to remove such complex shapes.

We now describe the details of our algorithm. The terms ‘point’ and ‘node’ are used interchangeably for the rest of the algorithm.

As shown in Line 3 of Algorithm 4.1, we first generate a Delaunay triangulation for all the points in P . Next, we delete the edges that connect two nodes in different clusters to separate the clusters (Lines 5 - 7). We further detect the outer edges by deleting all edges that appear twice in the triangulation result (Line 8). As illustrated in Figure 4.5 (a), a concave hull would be generated for each cluster in most cases. However, since the triangulation is generated based on all points, after we separate the clusters, there could exist exposed lines (explained later),

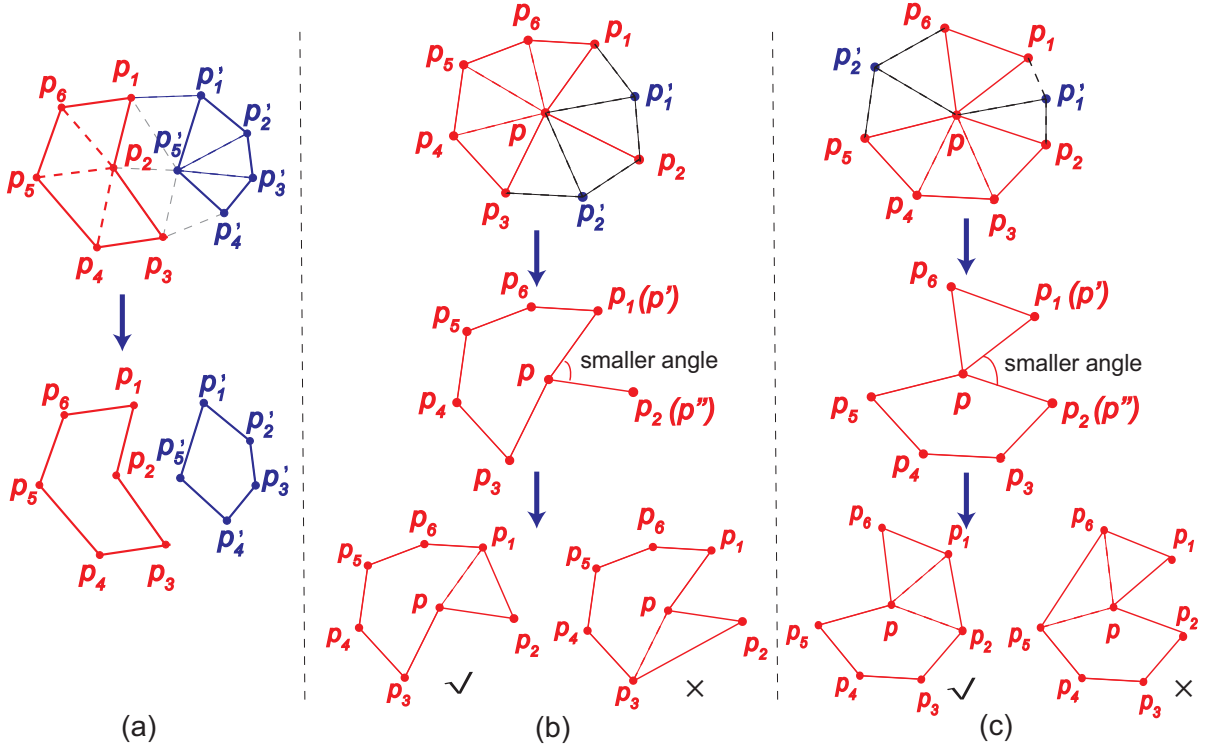


Figure 4.5: Illustration of the proposed Concave hull algorithm. (a) the process of separating clusters after a global triangulation; (b-c), two examples of non-Jordan boundaries: how the boundary is formed and how we eliminate it.

e.g., $\langle p, p_2 \rangle$ in Figure 4.5(b) or point connections [Galton and Duckham, 2006] (e.g., p in Figure 4.5(c)), which make the hull a non-Jordan boundary.

Eliminating non-Jordan boundaries. The degree of each node of a Jordan boundary must be equal to 2 where the nodes form a closed loop. Therefore, if the resulting concave hull of a cluster in the previous step is a non-Jordan boundary, that implies there must be at least two nodes with a degree not equal to 2.

We denote an edge as an ‘exposed line’ if the edge is not part of any loop (e.g., $\langle p, p_2 \rangle$ in Figure 4.5(b)). Let an angle $\angle p_i p p_j$ created at node p be denoted as an ‘outer angle’ where the other points p_i and p_j do not form a closed loop with each other (i.e., they are either involved in two different loops - $\angle p_1 p p_2$ or $\angle p_5 p p_6$ in Figure 4.5(c), or at least one of them is connected with an exposed line - $\angle p_1 p p_2$ or $\angle p_2 p p_3$ in Figure 4.5(b)). According to the properties of a Jordan curve, where the nodes of a cluster should form a single loop, there must not be any such outer angle. Thus we transform the problem of obtaining a Jordan curve from a non-Jordan

one to the problem of eliminating any outer angles. We present the steps of our technique in the following:

Let p be a node such that at least one outer angle is created at p . We find the set A of all the outer angles created at p . We find the smallest angle from A (we will explain why we deal with the smallest angle in Example 4.4). Let that angle be $\angle p'pp''$ (Lines 14 - 15). As the nodes p' and p'' of an outer angle do not form a closed loop with each other, we create an edge by adding p' and p'' (Line 20). Then, if the edge $\langle p', p \rangle$ was not an exposed line, we remove $\langle p', p \rangle$ (as the degree of p' was at least 2, and its degree got increased by 1 because of the new edge). Similarly, if the edge $\langle p, p'' \rangle$ was not an exposed line, we remove $\langle p, p'' \rangle$ as well (Lines 16 - 19). We continue this process until there is no node left with any outer angle, i.e., the Jordan curve is obtained. We present the steps of eliminating non-Jordan boundaries in Lines 9 - 20 in Algorithm 4.1. There is a special case where there are only two nodes in a cluster. In such case, we disconnect them by removing the edge (Lines 9 - 12).

Example 4.4. *Figures 4.5(b & c) present two examples of how the outer angles are formed and how we eliminate them. In Figure 4.5(b), since the red points and the blue points belong to different clusters, after we remove the grey edges to separate the two clusters (Lines 5 - 7), $\langle p, p_2 \rangle$ becomes an exposed line. There are two ways to eliminate the exposed line by dealing with either of the two outer angles $\angle p_1pp_2$ or $\angle p_2pp_3$. We choose to deal with the smaller angle $\angle p_1pp_2$ (rather than $\angle p_2pp_3$) by replacing pp_1 with p_1p_2 , which will generate a smaller interior angle (the interior angle $\angle p_2pp_3 < \angle p_1pp_2$ in the final hull). Based on the complexity function defined in [Brinkhoff et al., 1995], a smaller interior angle will reduce the chance of notches, which will also reduce the complexity of the concave hull. Similarly in Figure 4.5(c), dealing with the smaller outer angle $\angle p_1pp_2$ will also generate a less complex hull.*

Time complexity. The time complexity of the proposed algorithm is mainly affected by the triangulation process, which is $O(n \log n)$ [Duckham et al., 2008], where n is the number of points in P . The process of separating clusters and deleting all inner edges (Lines 5 - 8) depends on the number of edges in the triangulation result which is proved to be no larger than $3n$ [Seidel, 1995]. The complexity of the core algorithm loop (Lines 9 - 20) is based on the number of special cases - outer angles (s). In the real case, $s \ll h \ll n$, where h is the number of points that lie on the final hull boundaries. Therefore, the overall complexity of the algorithm is $O(n \log n)$.

The result. Figure 4.4(c) presents the result of concave hulls using our proposed algorithm (with the same real estate data as in Figure 4.4(a & b)). Our proposed algorithm, which does

not need any parameters, is able to avoid all three problems: overlaps, complex shapes, and empty areas.

4.5 ConcaveCubes: Cluster-based Data Cubes index

In this section, we propose ConcaveCubes to support the visualization design (Section 4.3.3) on large-scale data from a data reduction perspective. We first give an overview of ConcaveCubes followed by an example, and then describe how we construct ConcaveCubes. Finally, we discuss how different user interactions are supported.

ConcaveCubes is a data cube structure that precomputes clusters to support visualization and user interactions. ConcaveCubes addresses the scalability issue from the data reduction perspective as: (i) the hierarchical structure supports accessing only the data points necessary for the current view during visualization and user interactions (explained in Section 4.5.3); (ii) the precomputed aggregations are used to reduce the on-the-fly computation without accessing the actual data points.

4.5.1 An example of ConcaveCubes

ConcaveCubes has a hierarchical structure with multiple levels, and each level corresponds a specific dimension type: recall the baseline clustering method defined in Section 4.3.2, we have subdivided dimensions into four types based on whether they are directly linked to geo-location and whether they are going to affect the clustering result. To generate ConcaveCubes, each level of ConcaveCubes corresponds to one type of dimensions: as geo-independent dimensions, geo-semantics and geo-location.

Figure 4.6 presents an example of ConcaveCubes with Australia’s real estate data. The data is first sorted in the first 4 levels. The nodes at the last geo-semantic level will be split if they belong to different clusters at the next level. For example, the highlighted node (in orange) at level 4 contains all 2-bedroom units in Melbourne, Victoria. It is split into two nodes ($C1$ & $C2$) based on the clustering function at level 5, and $C1$ is further split into two nodes based on a finer clustering function at level 6. For each node in level 5 & 6, we store the statistical information of the properties in each cluster (shown as $[i,j]$ in Figure 4.6, where i and j indicate the start and end index of properties, respectively).

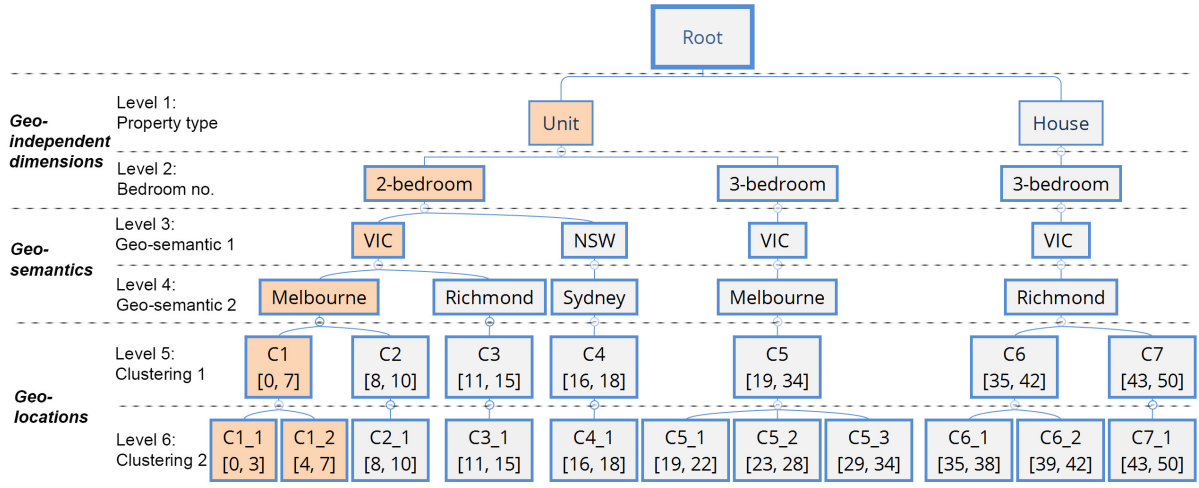


Figure 4.6: *An example of ConcaveCubes based on real estate data.*

4.5.2 The construction of ConcaveCubes

As shown in Figure 4.7, ConcaveCubes is constructed in four main steps: (1) data pre-processing, (2) a multi-criteria sorting on the categorical and geo-semantic dimensions, (3) hierarchical clustering, and (4) cluster-based calculations.

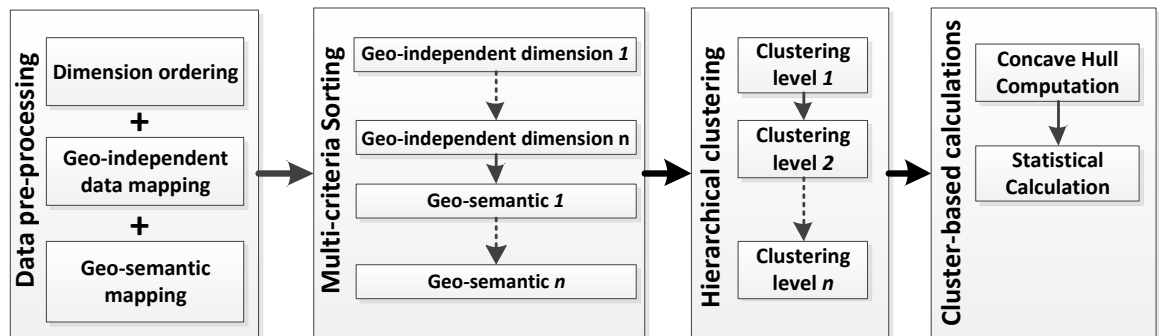


Figure 4.7: *Illustration of the ConcaveCubes construction procedure.*

Data pre-processing. Before generating ConcaveCubes, we pre-process the raw data as follows: (i) for each geo-independent dimension and each geo-semantic level, we set an ordering of the dimensions; (ii) we divide the value of geo-independent dimensions based on specific values or ranges, which convert them into categorical values; (iii) we map each data point to its corresponding geo-semantic region (e.g., a property might be mapped to Australia \rightarrow Victoria

→ Melbourne).

Multi-criteria sorting. Similar to Hashedcubes, our ConcaveCubes also needs a multi-criteria sorting. Note that, our geographical level is different from that in state-of-the-art data cubes. In Nanocubes and Hashedcubes, the geographical features are processed based on the Quadtree, a grid-based hierarchical spatial data structure that recursively divides the space into four regions based on either its interior [Samet and Tamminen, 1986] or its boundary [Samet and Webber, 1984]. Such a structure is 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-location of properties using a Quadtree, in one of the middle levels, 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 ConcaveCubes based on real-world geographical hierarchy, such as country, state, city.

Hierarchical clustering. After the multi-criteria sorting, all properties having the same values in geo-independent dimensions and in the same geo-semantic area will be grouped together. However, properties in the same group can still be quite different. For example, in Figure 4.2(b), houses that are close to the beach are more expensive than others even when they are in the same suburbs. Therefore, we apply hierarchical clustering [Murtagh, 1983] after the multi-criteria sorting to further divide the data into different groups. In our implementation, we use a hierarchical DBScan [Campello et al., 2013] by decreasing the threshold of the algorithm at a lower level. For example, at level 5 in Figure 4.6, we apply DBScan [Ester et al., 1996] with a threshold of 400m to cluster the properties based on Euclidean distance. Then at 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.

Cluster-based concave hull calculation. We calculate a concave hull for each cluster in ConcaveCubes based on the algorithm defined in Section 4.4 to support the boundary-based cluster maps. Since our proposed concave hull method is based on a global triangulation, we apply the proposed algorithm on a group of clusters that share a parent-node. Instead of storing the locations for all data points, we store the concave hull boundaries. At last, for each cluster that has a concave hull (clusters with less than three points will not have a hull), we calculate the statistical information based on each geo-dependent and geo-independent measures (Section 4.3.2) and store them in the database. The statistical (aggregated) information is stored as views in the database, so that when a cluster is highlighted, the information of the selected dimension measures will be efficiently loaded to visualize.

4.5.3 Index operations

In this subsection, we illustrate the operations that ConcaveCubes provides to support the visualization design and user interactions defined in Section 4.3.3.

4.5.3.1 Initialization

The initialization of the visualization is a query on ConcaveCubes with four parameters, a default map window, geo-independent dimensions (e.g., 3-bedroom houses), a default clustering granularity (e.g., a 400m DBScan), and a focused measure (e.g., price). Based on the default settings, ConcaveCubes is traversed from the root and the following steps are executed: (i) the nodes matching the geo-independent dimensions are scanned, (ii) the nodes in geographic semantic levels within the map window are accessed, and (iii) the clusters at the default granularity level are obtained.

4.5.3.2 User interactions

The key idea for ConcaveCubes to support different user interactions is to reuse existing calculations and visualizations while updating for user interactions.

Interaction 1: zoom-in, zoom-out, panning. Zooming and panning directly influence the map window (w). The change of w corresponds to the level of geographic semantics in ConcaveCubes. Therefore, after users apply zooming in/out or panning, we calculate a set subtraction between the current (w_c) and previous (w_p) map windows, and then refresh the visualization based on the subtraction result. Since the boundary of a semantic geographical node often contains many geo-points, we store a 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.

Interaction 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 at level 5 will be added to the visualization result while the previous result remains. Note that, having multiple selections might result in some intersecting concave hulls. For each concave hull, we store a minimum bounding rectangle that covers it, so that we can effectively detect intersections and compute the union of the intersected concave hulls at runtime.

Interaction 3: granularity control. For example, in Figure 4.6, if the user changes the current granularity level from Level 5 to Level 6, the clusters that are not split will stay in the

map window, but they will fall into multiple smaller clusters. For example, $C1$, $C5$ and $C6$ at level 5 will be replaced by their child clusters at level 6.

4.6 A demonstration system

We have designed and implemented a demonstration system¹ based on real estate data (Section 3.3). In this section, we illustrate how we use ConcaveCubes to support our visualization and user interactions. Each method is followed by a demonstration example based on the real estate dataset.

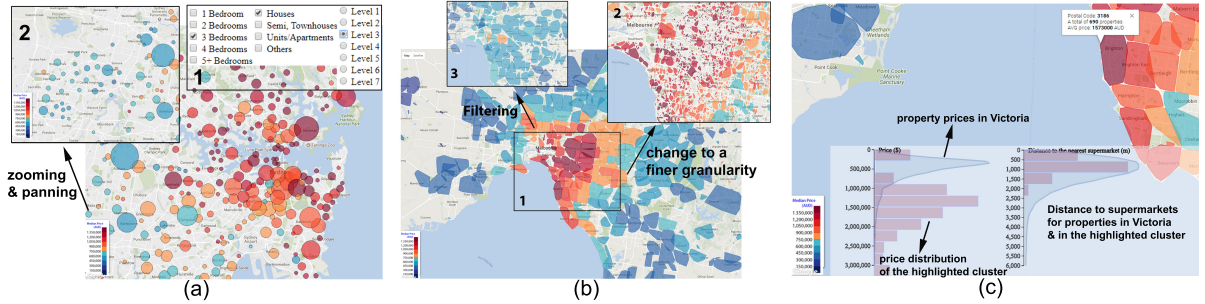


Figure 4.8: *Demonstration examples: (a) map view 1: dot map (Sydney), with zooming & panning illustrated; (b) map view 2: polygons on top of the map (Melbourne), with filtering and granularity control illustrated; (c) the multidimensional view when a cluster is highlighted (properties in the cluster are compared to that of the entire state based on 2 user-selected measures (the price & the distance to supermarkets)).*

4.6.1 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 ConcaveCubes 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 Figure 4.6, only the highlighted clusters in level 5 will be visualized for the default initialization

¹<http://115.146.89.158/ConvexCubes/>

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 example 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 4.8 (a)-1), either of the two map views will be shown to the user (Figure 4.8 (a) or (b)) based on the stored information. For example, in Figure 4.8 (b), the user can observe how the prices of 3-bedroom houses are distributed in the geographical space.*

4.6.2 Supporting user interactions

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

4.6.2.1 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 ConcaveCubes. 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 example 2: *The user can zoom in or zoom out, or pan the map just as they do on Google Maps. ConcaveCubes re-use existing visualization results and partially refresh the interface. An example is shown in Figure 4.8(a).*

4.6.2.2 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 example 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 4.8(b)-3, we change the filtering setting from 3-bedroom houses to 2-/3- bedroom units in Melbourne.*

4.6.2.3 Granularity control

This will affect the level of numerical dimensions. For example, in Figure 4.6, 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 example 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 4.8(a)-1). As shown in Figure 4.8(b)-2, the user can change to a finer granularity and see how the clusters in 4.8(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.*

4.6.2.4 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 example 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 4.8(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*

Table 4.2: *A summary of the relevant information for building ConcaveCubes.*

Dataset	Objects (#)	Clusters (#)	Memory (MB)	Time	Geo-scope	Geo- semantics	Categorical Dimensions	H- clustering	measures (#)
Real_estate _VIC	667k	50k	23.4	4m:16s	Victoria	3 levels	pro_type, bedroom	6 levels	36
Real_estate _AU	1.41m	104k	49.9	9m:8s	Australia	4 levels	pro_type, bedroom, year	6 levels	36
Brightkite	4.5m	489k	98.4	58s	World	3 levels	hour, day, week ¹	-	1
		570k	117.3	1m:27s	World	3 levels	month, hour, day ²	-	1
		33k	4.2	25m	World	6 levels	-	3 levels	1
Gowalla	6.4m	37k	4.5	48m:3s	World	6 levels	-	3 levels	1
Census _AU	23.4m	57k	7.9	35s	Australia	4 levels	-	-	100

¹ categorical dimensions used in Nanocubes and Hashedcubes; ² categorical dimensions used in imMens and Hashedcubes.

entire Victoria based on two user-defined measures (the price and the distance to the nearest supermarket).

4.7 Experiments

In this section, we present an extensive experimental evaluation of ConcaveCubes using several real-life datasets. We first describe the datasets and schemas, and report the storage and construction time of ConcaveCubes. We then present several visualization results, and compare our result with the result generated based on existing visualization-based data cubes. Finally we report the query time based on a set of real-world queries.

4.7.1 Datasets and schemas

We have collected five datasets that include two real estate datasets, a census dataset, and two location-based social network datasets. We summarize all of the schema variations and datasets in Table 4.2.

- **Real estate dataset.** We have crawled and cleaned the real estate data in Australia using the data profiles defined in the previous chapter (Chapter 3). We include two

schemas in our experiments as shown in Table 4.2. The geo-semantic levels are based on ASGS (Australian Statistical Geography Standard [Australian Bureau of Statistics, 2016a]). Both the schemas include six levels of a hierarchical DBScan.

- **Location-based social network datasets.** Brightkite and Gowalla are two social network datasets with users' check-in locations [Cho et al., 2011]. For Brightkite, we build ConcaveCubes using three different schemas: two of them share the same categorical dimensions with imMens [Liu et al., 2013], Nanocubes [Lins et al., 2013], and Hashedcubes [Pahins et al., 2017]. The geo-semantics are all mapped based on GADM (Database of Global Administrative Areas [Areas, 2012]).
- **Australian census dataset.** It includes four levels of geo-semantics based on ASGS. We simulated n data points which belong to each of the 57,523 SA1s (the smallest area for the release of census data [Australian Bureau of Statistics, 2016a]), where n is the population in each area. As a result, we generated 24.5 million data (the total number of the Australian population at the 2016 census report [Australian Bureau of Statistics, 2016b]). Then, we select 100 features (including median age, income, etc.) from the 2016 census data of Australia.

We report the memory usage and construction time of ConcaveCubes in Table 4.2. Although the construction time of ConcaveCubes is much higher than that of Nanocubes and Hashedcubes, ConcaveCubes is much more memory efficient than the other two methods. On the one hand, for the construction time, the bottleneck of our construction process includes (i) geo-semantic mapping, (ii) hierarchical DBScan computation, and (iii) Concave hull calculation. The construction time of ConcaveCubes is proportional to the number of objects, and the number of hierarchical clustering levels. However, with more categorical dimensions or geo-semantics levels, the data entering the hierarchical DBScan process will be divided into more sub-groups, which results that the construction time of ConcaveCubes could possibly decrease. On the other hand, as examples of the memory efficiency of ConcaveCubes, it requires less than 30MB of memory for both the Brightkite and Gowalla datasets, where HashedCubes requires more than 300MB [Pahins et al., 2017] and Nanocubes needs more than 1GB [Lins et al., 2013]. The reason behind this is that, we store the geo-semantic information for the geographic dimensions which are different with Nanocubes and Hashcubes, which use Quad-trees; as a result, much fewer data aggregations will be needed in the bottom layers of ConcaveCubes.

4.7.2 Visualization results

In addition to the visualization of real estate data shown in the previous section (Figure 4.8 & 4.6) here we present visualization results based on the other datasets and compare them with existing methods.

Location-based social network datasets. Based on the Brightkite dataset, Figure 4.9 compares the visualization results supported by ConcaveCubes with the visualization supported by Nanocubes, Hashedcubes, and imMens. The two visualizations present both global patterns (e.g., an overview of the check-ins across the United States) and local features (e.g., inter-state highway travel). However, the boundary-based cluster maps, which ConcaveCubes supports, present important geo-semantic information which are missed in Heatmaps. For example, each polygon (cluster) in Figure 4.9(a) represents a group of data points within the same county and close in distance.

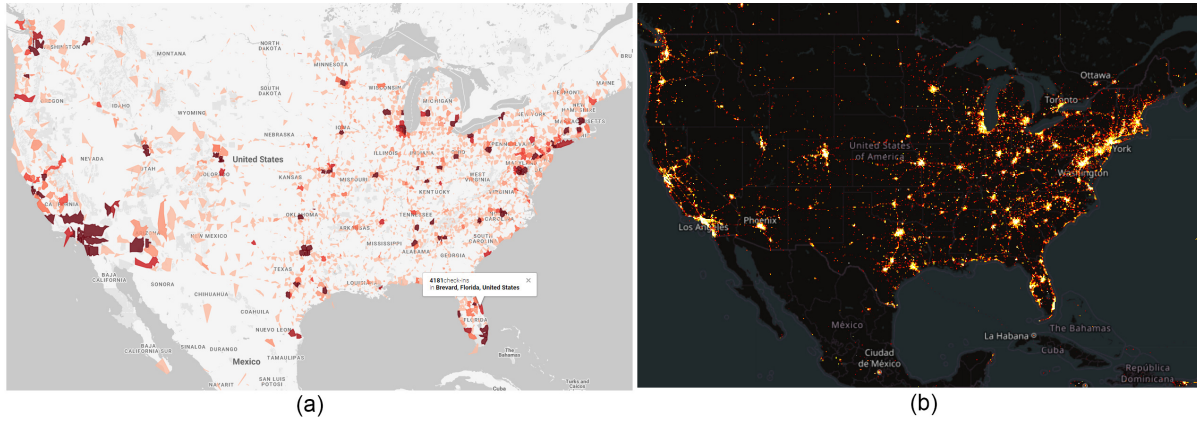


Figure 4.9: Visualization of the Brightkite dataset. (a) cluster maps (with a cluster selected) supported by our proposed ConcaveCubes: colour represents no. of check-ins; (b) heatmaps supported by Nanocubes, Hashedcubes, and imMens (screenshot is taken from the online demo of Nanocubes).

As shown in Figure 4.10(a), we generate a cluster map based on a group of simulated data points at each geo-semantic level in the Australia census data. ConcaveCubes is able to provide an efficient approximate solution of choropleth maps [Brewer et al., 1997] by simulating geographic points in each region and calculating approximate boundaries based on the simulated points: comparing to the original dataset which takes more than 200MB to store the geographic boundaries, ConcaveCubes needs only 7.9MB of memory.

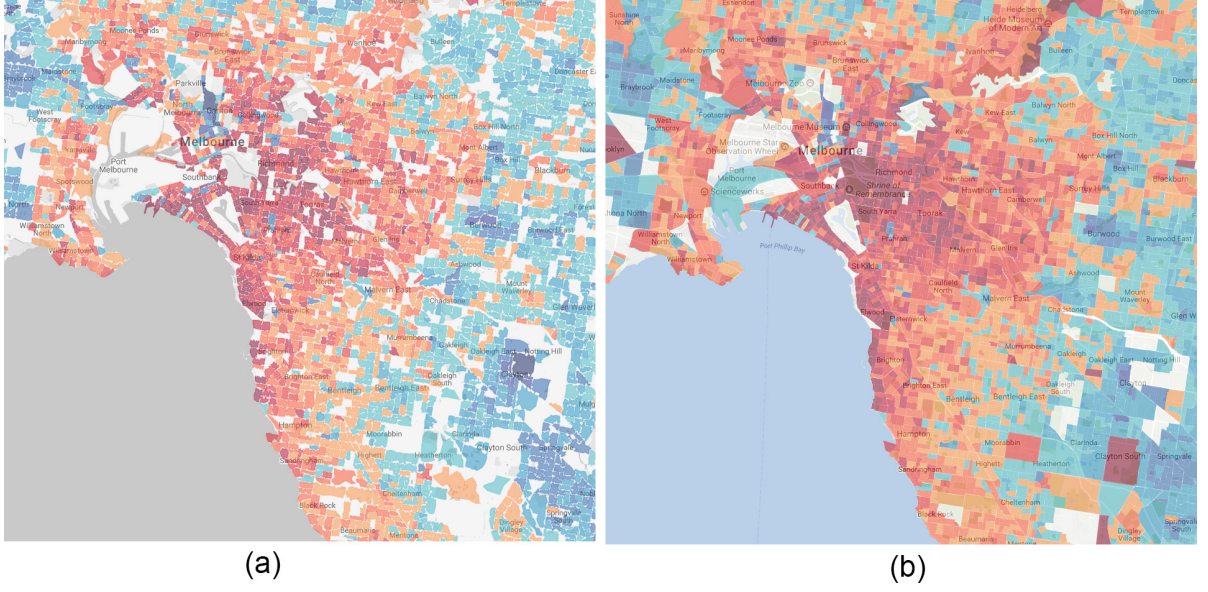


Figure 4.10: Visualization of the Australian census data: *ConcaveCubes* is efficient to support cluster maps (a), which could be an approximate solution of the traditional choropleth maps (b).

4.7.3 Query time

We report the query time based on the real usage of the system. A total of 6,317 query requests were collected on the public *ConcaveCubes* web site², in which users performed different user interactions based on the real estate data in Victoria. To evaluate the query efficiency in a fixed environment, we recorded the query related to each user interaction and re-executed it in Google chrome on a quad-core i7-5500U CPU (2.4GHz) with 8GB RAM. Figure 4.11 shows the result based on six different types of user interactions.

Comparing to *Hashedcubes* which have only about 2% of the queries taking longer than 40ms (as stated in [Pahins et al., 2017]), most of the queries in *ConcaveCubes* are responded within 25-50ms. The reason of why *ConcaveCubes* might take a slightly longer time is that: we exploit geo-semantic information in *ConcaveCubes* and display the boundaries for each cluster, while *Hashedcubes* do not need to deal with the boundaries. Nevertheless, all of our queries (from the experiments) are returned within 0.1s, which satisfies the perceptual processing time constant defined by Card et al. [Card et al., 1991]. Among all different types of user interactions, highlighting & linking is the most efficient operation: it retrieves the aggregation information of

²<http://115.146.89.158/ConvexCubes/>

that cluster from ConcaveCubes directly and then displays them, so no additional calculation is required; granularity control takes slightly longer time than the other operations, since users might select a finer granularity in a high zoomed level, which means several levels of hierarchical clustering levels might need to be accessed.

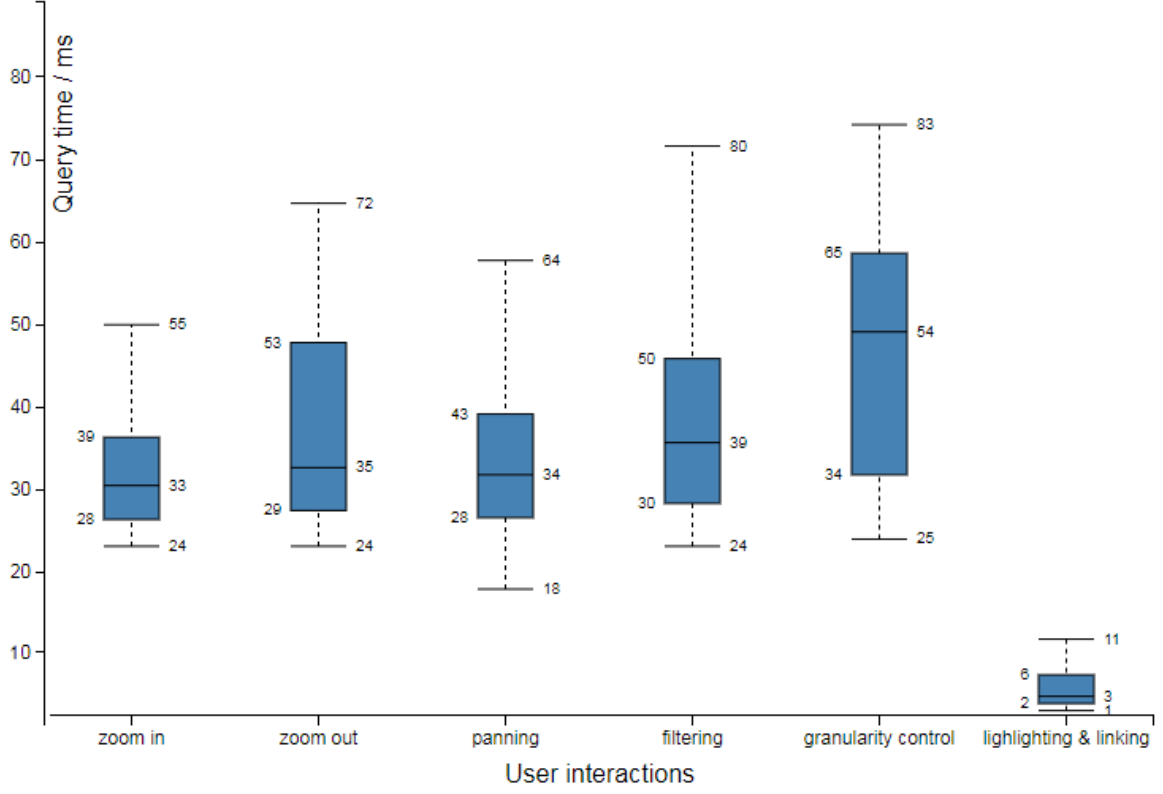


Figure 4.11: Query latency per type of user interactions based on the real estate dataset.

4.7.4 Discussion

Based on the experimental results, we have demonstrated that ConcaveCubes supports both bubble-based and boundary-based cluster maps, which is not straightforward to be supported by existing methods (e.g., imMens, Nanocubes, Hashedcubes). Compared to heatmaps and binned plots, bubble-based cluster maps present additional information (with the color and the size of each bubble), and our proposed boundary-based cluster maps present more precise geographical information with semantic meanings.

Situations where cluster maps are most applicable. Our proposed boundary-based

cluster maps can be regarded as a generalization of choropleth maps: the surface of each single color element is not directly the representation of an explicit surface delimitation (district limits for instance), but the result of the aggregation/clustering of data. Our methods are most suitable to apply when geo-referencing is just point-based or when the query needs to go beyond known space delimitations such as districts (real estate and social network examples). From the example of Census data, we also demonstrate that, even when the boundary of geo-semantics is given, ConcaveCubes can provide an approximate yet more efficient solution of the traditional choropleth maps.

Limitation of cluster maps. While cluster maps that the proposed ConcaveCubes support have advantages over other map designs, the method also has its limitations. For examples, bubble-based cluster maps lack precise geographic information, and boundary-based ones might not present density information. One future research direction is to conduct a proper user study to compare different types of map designs (including the above-mentioned ones and some other map designs such as dot maps and cartogram maps) based on different visualization tasks.

Limitation of dimensions supported by ConcaveCubes. Apart from the disadvantages of the supported cluster maps, another main limitation of the proposed ConcaveCubes is the number of dimensions that it supports, which is a common problem for cube-based data structures. Similar to Nanocubes and Hashedcubes, ConcaveCubes can support at most 5-10 categorical dimensions. It is worth noting that, besides the geographic dimensions and categorical dimensions, ConcaveCubes is able to support temporal dimensions using the same method defined in [Pahins et al., 2017] by converting the temporal dimensions into categorical dimensions.

Supporting dynamic data. Although ConcaveCubes is pre-defined in all the experiments, one possible extension of our work is to adapt the method with dynamic data. To insert, update or delete a data item in ConcaveCubes after it has already been constructed, we only need to change the affected nodes in the hierarchical structure. For example, to insert a new 2-bedroom unit at Melbourne to the ConcaveCubes shown in Figure 4.6, we need to find the corresponding nodes in Levels 1-4 that the new property belongs to (the highlighted nodes); then based on whether the new property is within the concave hull boundary of any existing clusters ($C1$ or $C2$), we update the corresponding node, or add a new node (if the new property does not belong to either of the clusters).

4.8 Summary

In this chapter, we proposed ConcaveCubes, a data cube based scalable approach to support interactive visualization of large-scale multidimensional data with geographic semantic captured. In particular, we first presented a visualization abstraction for the geographical data based on clusters and proposed a visualization design of two map-based views (bubble-based as a baseline and boundary-based cluster maps as our desired map view) and a linked multidimensional view. For an effective boundary-based cluster map view, we proposed a novel concave hull construction algorithm, which addresses the limitation of existing methods by eliminating empty areas, complex shapes, and overlaps among different clusters. The ConcaveCubes utilizes the existing calculations based on the categorical and geo-semantic dimensions of the clusters and visualization results to efficiently support various types of user interactions. We designed and developed a demo system to illustrate how ConcaveCubes supports large-scale geo-related multidimensional data. Extensive experiments on real-world datasets verified the efficiency and effectiveness of our proposed ConcaveCubes.

Chapter 5

AOI-shapes: Interactive Visualization of Urban Areas of Interest

“The trick of it...is to be courageous and bold and make a difference. Not change the world exactly, just the bit around you...Change lives through art maybe. Write beautifully. Cherish your friends, stay true to your principles, live passionately and fully and well. Experience new things. Love and be loved if at all possible.”

— David Nicholls, *One Day*

In this chapter, we study the problem of efficient and effective visualization of user-defined urban areas of interest. In Particular, we first describe a design space for visualizing urban AOIs based on points of interest. After extensively reviewing existing “footprint” methods, we propose a novel parameter-free footprint method, named AOI-shapes, to capture the boundary information of a user-defined urban AOI. Next, to allow interactive query refinements by the user, we propose two efficient and scalable algorithms that reuse existing visualization results to incrementally generate urban AOIs. Finally, we conduct extensive experiments with both synthetic and real-world datasets to demonstrate the effectiveness and efficiency of the proposed methods.

5.1 Introduction

One of the most important issues related to geo-related multidimensional data is to understand urban areas of interest (AOIs). An urban area of interest (AOI) refers to the area within an

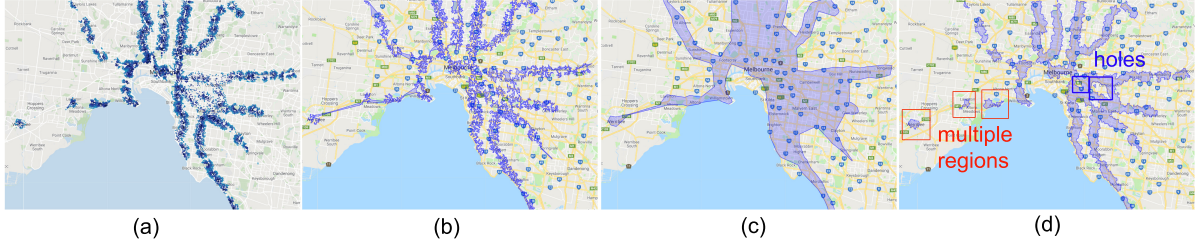


Figure 5.1: An example of user-defined AOIs (properties within a 15-min walk to the nearest train station) presented as: (a) individual property points (method defined in Chapter 3), (b) a polygon-based region generated using χ -shapes [Duckham et al., 2008] ($\chi=0.05$); (c) a polygon-based region generated using χ -shapes [Duckham et al., 2008] ($\chi=0.2$); (d) polygon-based regions generated using our proposed AOI-shapes.

urban environment which attracts people’s attention [Hu et al., 2015]. Taking house seeking as an example, location is a key factor, based on which various AOIs can be defined by users to cater for their respective concerns/interest. The following example motivates us to investigate the AOI visualization problem.

Example 5.1. In Chapter 3, we propose a visual analytics system for Australia’s real estate data to help users understand the local real estate market and find their preferred housing properties. At the property level exploration, users can filter out properties based on their own requirements over the multidimensional view. After that, all properties that satisfy the user query will be shown on top of the map. As shown in Figure 5.1(a), the user prefers to live in a place within a 15-minute walk to the nearest train station. Properties that satisfy such a query will result in a user-defined urban AOI. Understanding such AOIs will assist home buyers in understanding different lifestyles in different regions, and help the government make policies.

AOIs could be represented either as polygon-based regions or as a set of individual points (e.g., real estate properties). Previous studies have shown that the polygon-based representation performs better from both cognitive and computational perspectives [Akdag et al., 2014]. However, unlike most well-defined administrative districts (e.g., suburbs, cities), there are normally no defined boundaries for AOIs. The region (shape) of an urban AOI could be vague [Hu et al., 2015] and often depends on users’ preference.

One of the most popular methods to characterize the shape of an AOI is to generate the region boundaries, which are called “footprints”, based on the points of interest (POIs) that satisfy the user query. For example, Polisciuc et al. [2015] generate the footprints based on

people’s visited places and visualize the footprints to help understand the land use for urban planning and management. In Example 5.1, the AOIs shown in Figure 5.1(b-d) are generated based on the POIs in Figure 5.1(a) (the housing properties satisfying the user query).

Different footprint algorithms have been proposed, including Convex hulls [Graham, 1972], concave hulls [Park and Oh, 2012a], KNN-based concave hulls [Moreira and Santos, 2007], χ -shapes [Duckham et al., 2008], etc. However, footprints from a definite set of points are usually not unique [Ebert et al., 2015], i.e., the constructed footprints can be different in different algorithms and/or different parameters (further illustrated in Example 5.2); and there is no consensus on the effectiveness of the footprint algorithms.

After an extensive literature review (Section 5.2.2), we summarize the state-of-the-art footprint algorithms in Table 5.1. We find that, existing footprint algorithms suffer from at least one of the following drawbacks: (1) additional parameter tuning is needed from the user; (2) the method cannot recognize multiple regions and/or outliers; (3) the method cannot detect inner holes; and (4) the method is not extendable to support incremental generation of the footprint. The following example highlights the drawbacks of the existing methods.

Example 5.2. *We generate different footprints for the properties shown in Figure 5.1(a) using χ -shapes [Zhong and Duckham, 2017]. The algorithm first constructs a convex hull, and then executes a “digging” process (progressively replace an outer edge with two adjacent inner edges) to get a concave hull (i.e., footprint). The only parameter is a length-related threshold which decides whether each boundary edge is too “long” or not; and the algorithm will terminate if all boundary edges are shorter than the threshold. We use two different parameters to generate the footprints and present both the results on Google Maps (Figure 5.1(b & c)). The results illustrate the first three drawbacks of existing footprint algorithms: (i) a smaller threshold may make the result too “concave” for users to recognize the region (Figure 5.1(b)), and a larger threshold may cause the region to be too “big” to characterize the shape of the region (Figure 5.1(c)); unfortunately, it is hard to find an appropriate threshold (which does not have intuitive real-life meanings) for an unknown set of data points. (ii) since the algorithm cannot recognize multiple regions, those regions (as highlighted in red in Figure 5.1(d)) are unexpectedly connected in Figure 5.1(b & c); (iii) there are regions in the middle of two train lines (as highlighted in Figure 5.1(d)) that do not satisfy the user query; however, since χ -shapes algorithm might not detect those holes (Figure 5.1(c)).*

Different from other footprint-related applications, in our application, an AOI is generated by a user query, and the user query will return two separate sets of points: those that satisfy

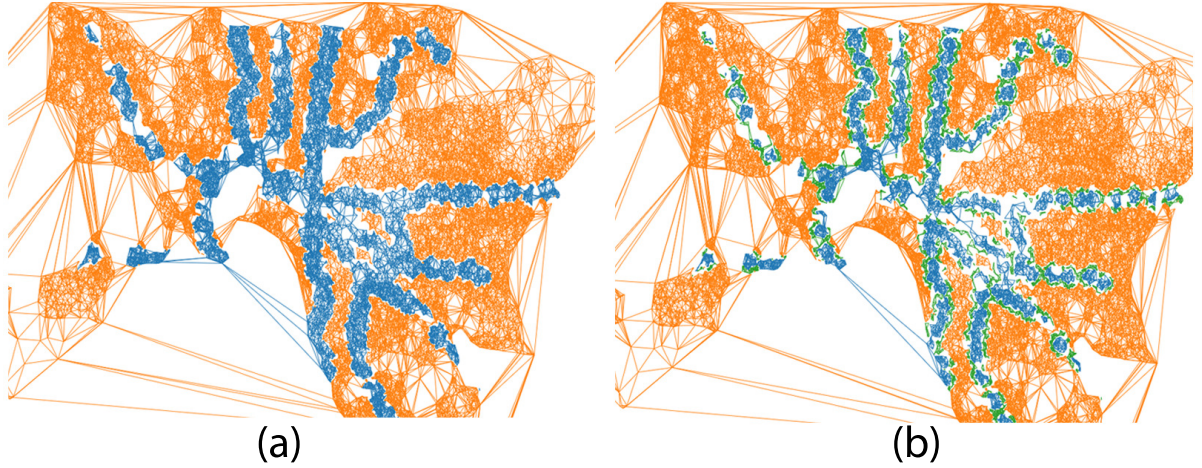


Figure 5.2: Illustration of the proposed incremental AOI-shapes based on a global Delaunay triangulation. (a) user query 1: 15-min walk to the nearest train station (blue points satisfying the query, orange points not satisfying the query); (b) a subsequent user query 2: less than 15-min walk to the nearest train station, and 30-min train to the CBD (green points: those satisfying query 1 but not satisfying query 2).

the user query and those do not. Based on the intuition that, “the points that do not satisfy the user query should not be in the generated AOI”, we propose a parameter-free footprint method, named AOI-shapes, to describe the boundary of the user-defined urban AOI. The proposed algorithm is based on a global Delaunay triangulation [Samet, 2006] (Figure 5.2(a)), followed by a separation of points/regions considering whether the data points meet the user query (inside the AOI, blue points in Figure 5.2(a)) or not (outside the AOI, orange points in Figure 5.2(a)). As a result, AOI-shapes is able to overcome all four drawbacks of existing methods. First, based on whether or not each point satisfies the user query, the region will be naturally separated; we further define boundary edge conditions based on triangle properties (details at Section 5) to make the AOI-shapes more concisely represent the region without additional parameter tuning needed (1st drawback). Second, multiple regions, outliers and holes are detected based on the points that do not satisfy the query (2nd and 3rd drawbacks). Third, we extend AOI-shapes by exploiting the common result points between two consecutive rounds of user queries, and propose two incremental AOI-shapes algorithms to support efficient visualizations of AOIs (the last drawback).

Example 5.3. Suppose that, after visualizing the result for the properties within 15-min walking distance, the user now wants to refine the query to visualize the properties within 15-min walk

to the nearest train station and also within a total of 40-min travel time (by train+walk) to his workplace in the CBD. Such a query will result in a new AOI (Figure 5.2(b)), which is a sub-region of the previous AOI (Figure 5.2(a)). If an incremental update is not supported, the AOI and the boundaries of the AOI need to be re-calculated from scratch for this visualization. As many properties are shared between the examples in Figure 5.2(a) and Figure 5.2(b), a method that updates the visualization by incrementally considering only the necessary changes is highly important for efficiency and scalability, especially for a large-scale dataset.

To summarize, we make the following contributions in this chapter:

- We present a comprehensive literature review of the existing footprint methods (Section 5.2.2) and summarize them based on the construction process and the characteristics of the result (Table 5.1).
- We propose a novel footprint method, named AOI-shapes, to effectively capture the region of an urban AOI based on POIs that satisfy the user query and those that do not (Section 5.5).
- We propose two efficient and scalable solutions to incrementally generate the AOI-shapes by using existing result after the query is refined by user interactions (Section 5.6).
- We conduct extensive experiments based on both synthetic and real-world datasets to evaluate the effectiveness and efficiency of the proposed methods (Section 5.8).

5.2 Related Work

In this section, we review the closely related work on the visualization of AOIs and the footprint methods for calculating the AOI.

5.2.1 Visualization of urban areas of interest (AOIs)

The concept of AOIs [Hu et al., 2015] is closely related to points of interest (POIs). While a POI represents an individual location (e.g., a property, or a landmark), an AOI usually contains multiple geographic points, such as the restaurants on a pedestrian street [Elias, 2003], nearby landmarks [Raubal and Winter, 2002], or properties within a 15-min walk to the nearest train station (Figure 5.1).

In terms of geographic visualization, AOIs are often represented as polygon-based areas [Celikten et al., 2017, Xing et al., 2017, Gao et al., 2017, Cao et al., 2013, Hu et al., 2015, Jung et al., 2010, Packer et al., 2013] instead of as individual points. For example, Hu et al. [2015] extracted the urban AOIs using geo-tagged photos and represented the AOI as a polygon, which is generated based on the POIs using the χ -shape algorithm; Jung et al. [2010] also used the χ -shape algorithm to generate the “footprint” based on a group of geo-referenced photos (e.g., all the photos with a ‘park’ tag in Manhattan, New York), and then visualized the “footprint” as polygons for its represented AOI on the map. Previous studies have shown the advantages of using polygons to represent AOIs compared to using individual points. First, from the cognitive perspective, polygons can provide simple and accessible representations for areas compared with clustered points [Hu et al., 2015]. Second, from the computational perspective, it is generally more efficient to perform operations on a polygon than on a set of points [Akdag et al., 2014]. Third, polygons convey information about the shape of an area and can be employed for shape-based analysis [Li et al., 2014]. In this chapter, we also adopt such a polygon-based representation of the AOI.

5.2.2 Footprint methods: generating boundaries for AOIs

Unlike most well-defined administrative districts (e.g., suburbs, states), the boundaries of an urban AOI are often vague [McKenzie et al., 2015, Hu et al., 2015]. To define its boundary, several algorithms have been proposed to generate a polygonal “footprint” that characterizes the distribution of a set of points P in the two-dimensional plane, i.e., $P \subset R^2$. A trivial approach is to use the minimum bounding rectangle [Leonard et al., 1992] to represent an AOI, as shown in Figure 5.3(a). There have been some non-trivial footprint methods proposed, such as convex hulls, simple concave hulls, and non-simple concave hulls (see Fig. 5.3(b), (c) and (d) respectively).

5.2.2.1 Convex hull

A more sophisticated version of a rectangular box is the rectilinear convex hull, also known as the orthogonal convex hull. A convex hull (CH) of a point set P is the smallest convex set, which contains P . Different convex hull algorithms have been proposed to generate the convex hull for a set of points. One well-known method is the Graham Scan [Graham, 1972], which finds all vertices of the convex hull ordered along its boundary. Another method, called Gift wrapping [Jarvis, 1973], finds the next point which will form the widest angle with the

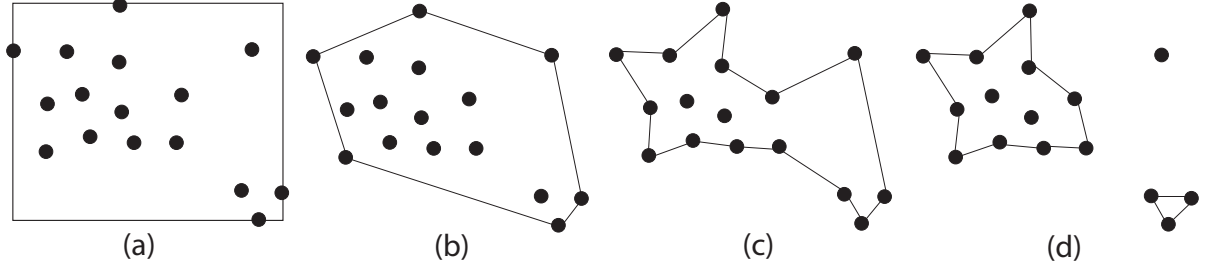


Figure 5.3: Examples of different kinds of hulls: (a) rectangular box, (b) the convex hull, (c) an example of simple concave hulls, and (d) concave hulls for multiple groups, an example of non-simple concave hulls.

previous two points. Some other notable algorithms include the Divide and Conquer algorithm [Preparata and Hong, 1977], Andrew’s Monotone chain algorithm [Andrew, 1979], Quickhull [Eddy, 1977, Bykat, 1978], and Chan’s method [Chan, 1996]. Since the convex hull for a set of points P is unique, all the aforementioned methods will generate the same result. Chadnov and Skvortsov [2004] conducted experiments to compare the efficiency of some of the methods.

5.2.2.2 Simple concave hull

A simple concave hull is a generalization of the convex hull, which allows any angle between two edges. Same as the convex hull, a simple concave hull is a Jordan-boundary hull that covers all the input points [Galton and Duckham, 2006]. Any simple shape, which is between the convex hull and the hull that completely minimizes the area, is regarded as a concave hull [Rosén et al., 2014]. Unlike convex hull, concave hulls from a definite set of points are not unique. Concave hulls can vary with different algorithms, and/or different parameters. More than 30 concave hull methods have been proposed in the past decade. To our best knowledge, there is no existing study that categorizes different concave hull methods. Here, we divide them based on how the concave hull is generated.

(1) Convex hull + “digging”. This group of methods generate the convex hull for the given data points and then progressively replace a “longer” outer edge with two inner edges (an iterative “digging” process) until the concave hull meets the requirement. On the one hand, some works focus on the “digging” process of convex hulls. For example, Park and Oh [2012a] proposed a method, which identifies a convex hull first, and then “digs” the hull to produce a concave hull with an appropriate depth, and the digging process is determined by a threshold value N which corresponds to the edge length. Polisciuc et al. [2015] proposed

a similar method which constructs a convex hull first but then pushes “long” edges inside to generate new boundary edges. Rosén et al. [2014] proposed a parallelized algorithm to further improve the performance. Simple-shape [Gheibi et al., 2011] starts reconstruction from the convex hull and makes it concave step by step based on a new hybrid selection criterion which is built on human beings’ visual perception. On the other hand, many researchers explored the “digging” process subsequent to the convex hull generated based on Delaunay triangulation (DT) [Samet, 2006]. One of the most popular and widely adopted methods is χ -shapes [Duckham et al., 2008] which starts with the DT and iteratively removes the longest exterior edge of the triangulation that does not cause “irregularity” (defined in [Duckham et al., 2008]). Also, a parameter-free method called *ec*-shapes [Saalima and Shajahan, 2016] constructs the DT of a given data set, and then successively remove exterior edges subject to the proposed “circle” and regularity constraints to compute a resultant boundary.

(2) Boundary detection. Another set of methods try to directly detect the boundary. Galton and Duckham [2006] proposed the swinging arm method, which generalises the gift wrapping algorithm [Jarvis, 1973] for concave hull generation. While finding the next point that will form the widest angle with the previous two points, it only finds the point within a distance of r (a fixed parameter) of the current point, instead of finding it in the entire set of points. The K-nearest neighbour (kNN) method [Moreira and Santos, 2007] also generalises the gift wrapping algorithm [Jarvis, 1973]. At each iteration, the kNN-based algorithm finds the next point from the kNN of the current point, instead of processing the entire point set P used in the gift-wrapping algorithm. Chaudhuri et al. [1997] defined an s -shape of a point set P similar to the process of pixelization, i.e., P is partitioned into a grid of square cells of side-length s . Then, an r -shape of P is defined to smooth the s -shape. Based on the observation that boundary points tend to have fewer reverse k-nearest neighbors (RkNNs, where the RkNN of an object p is the set of points that have p as one of their k-nearest neighbors), Xia et al. proposed the BORDER method [Xia et al., 2006] to detect boundary points in a point set. However, it is beneficial only for data sets that are well-clustered.

(3) Exploiting auxiliary points. In addition to the original point set P , several methods take a set of auxiliary points A to construct the concave hull. A -shapes [Melkemi, 1997, 2000] first computes the Voronoi Diagram (VD) of $P \cup A$, then the A -shape of P is defined by joining any pair of points $p_i, p_j \in P$ whose Voronoi cells share at least a border with some other Voronoi cells containing a point of A . Arampatzis et al. [2006] used a similar method to generate the region boundaries based on DT instead of VD. Alani et al. [2001] introduced a Voronoi diagram method for generating approximate regional extents from sets of centroids that are respectively

inside and external to a region.

(4) Based on mathematical definitions. One of the most famous methods is called α -shapes [Radke, 1983]: the α -hull of P is the intersection of all closed discs with radius $1/\alpha$ that contains all the points of P ($\alpha > 0$). If $\alpha < 0$, the α -hull of P is defined as the intersection of all closed complements of discs (where these discs have radius $-1/\alpha$) that contain all the points of P . The α -shape of P is the straight line version of the α -hull. When $\alpha = 0$, the α -shape of P is its convex hull. Asaeedi et al. [2017] defined α -Concave Hull which is a generalization of the convex hull. The parameter α determines the smoothness level of the constructed hull on a set of points, where $\alpha + \pi$ is the largest inner angle of the calculated concave hull. After proving that computing an α -Concave hull with a given area on a set of points is NP-complete for any $\alpha \in (0, \pi)$, the authors present an approximation algorithm to compute an α -Concave hull. ω -hull [Xu et al., 2010] is similar to α -Concave Hull, where ω refers to the smallest outer angle of the hull. Based on the definition, a convex hull is a class ω -hull of $\omega \geq \pi$. The authors proposed an algorithm to calculate ω -hull by extending the Grantham Scan algorithm [Graham, 1972] for computing the convex hull.

5.2.2.3 Non-simple concave hull

In this thesis, a non-simple concave hull refers to the concave hull that has non-Jordan boundaries, or allows multiple regions and outliers.

(1) Concave hull with hole detection. Several methods have been proposed which can effectively detect holes inside a hull. One of the most famous examples is the α -shape of a point set P ($\alpha < 0$). Variants of α -shapes such as k -order α -hulls and LDA- α -shapes [Maillot et al., 2010] effectively use the local density variation found in the point sets for detecting the hollow regions. Kolingerová and Žalik [2006] also defined a method based on DT to detect the domain boundaries of a given set P . It first constructs a DT for P and then removes those “unsatisfactory shapes of triangles” which are defined as triangles with too “big” or too “small” angles and triangles with too “long” or too “short” edges. As a result, the algorithm can detect both outer boundaries and inner holes. Pohl and Feldmann [2016] presented two methods based on the KNN-based Concave Hull Algorithm [Moreira and Santos, 2007] to create straight, angular and non-convex outlines of 2D point sets and possibly contained holes. Peethambaran and Muthuganapathy [2015] proposed a fully automatic Delaunay based sculpting algorithm to approximate the shape of a finite set of points P . It uses a combination of a local measure (circumcenter of Delaunay triangles) and a global measure (size of the Delaunay triangles

Table 5.1: *A summary of footprint methods*

	convex hull + "digging"	boundary detection	auxiliary points	mathematical definition	based on DT	based on VD	holes	outliers	multiple regions	parameter free	incremental method
	Construction process						Characteristics				
Rectangular box [Leonard et al., 1992]				✓					✓		
Rectilinear convex hull [Rawlins and Wood, 1987]				✓					✓		
Convex hull [Graham, 1972]				✓					✓	✓	
Concaveman* [Park and Oh, 2012a]	✓										
Polisciuc et al. [2015]	✓										
Rosén et al. [2014]	✓										
Simple-shape [Gheibi et al., 2011]	✓										
χ -shapes* [Duckham et al., 2008]	✓				✓						
ec-shapes [Methirumangalath et al., 2015]	✓				✓						
Swinging arm [Galton and Duckham, 2006]		✓									
kNN-based method* [Moreira and Santos, 2007]		✓									
s-shape [Chaudhuri et al., 1997]		✓									
r-shape [Chaudhuri et al., 1997]		✓									
BORDER [Xia et al., 2006]		✓									
A-shapes [Melkemi, 1997, 2000]			✓			✓	✓		✓		
Arampatzis et al. [2006]			✓		✓					✓	
Alani et al. [2001]			✓			✓				✓	
α -shapes* [Radke, 1983]				✓			✓				
α -Concave Hull [Asaeedi et al., 2017]				✓							
ω -hull [Xu et al., 2010]		✓		✓							
LDA- α -shapes [Maillot et al., 2010]				✓							
Kolingerová and Žalik [2006]					✓		✓				
Pohl and Feldmann [2016]							✓				
Peethambaran and Muthuganapathy [2015]					✓		✓			✓	
Methirumangalath et al. [2015]					✓					✓	
Zhu et al. [2017]							✓				
DBScan + χ -shapes [Zhong and Duckham, 2016]	✓				✓		✓		✓		
χ -outlines* [Zhong and Duckham, 2016]	✓				✓			✓	✓		
Incremental χ -shapes [Zhong and Duckham, 2017]	✓				✓						✓
Our proposed AOI-shapes			✓		✓		✓	✓	✓	✓	✓

* Methods in gray are used as alternatives in the experiment (Section 5.8)

quantified by the circumradius) for the filtration of the DT. As a result, the algorithm is able to generate a simplicial complex, and effectively detect holes inside the shape. Zhu et al. [2017] proposed an algorithm to detect cavities and holes from planar points based on density. The

algorithm conceptualizes cavity and hole as “a structure with a low-density region surrounded by the high-density region”.

(2) Concave hulls for multiple groups. Different algorithms have been proposed to characterize more complicated point distributions with disconnected shapes and outliers. One straightforward approach is to pre-process a spatial clustering algorithm such as DBScan [Ester et al., 1996], followed by any simple concave hull method (presented at Section 5.2.2.2). One main disadvantage of such a two-step method is that, any pre-clustering algorithm necessarily requires additional parameterization. Zhong and Duckham [2016] proposed χ -outlines, which extends χ -shapes by dealing with situations of disconnected shapes and excluding outliers.

5.2.2.4 A summary of existing footprint methods

We summarize different footprint methods in Table 5.1, where Columns 2-5 refer to the main construction process of the method. Since many methods are based on Delaunay Triangulation (DT) or Voronoi Diagram (VD), we summarize such information in Columns 6-7. We highlight the characteristics of the generated hull in Columns 8-11, as whether the boundary is a Jordan curve (Column 8), whether holes are allowed (Column 9), whether outliers are allowed (Column 10), and whether multiple regions are allowed (Column 11). Column 12 indicates whether the construction process is parameter-free, and the last column indicates whether an incremental method has been proposed.

To summarize, existing footprint methods suffer from at least one of the following drawbacks when we use them to generate user-defined urban AOIs based on the points that satisfy the user query: (1) the method cannot recognize multiple regions and/or outliers; (2) the method cannot detect holes; (3) the algorithm needs additional parameter tuning, and (4) the method does not support incremental generation of the footprint.

5.3 Problem illustration

In this section, we illustrate our problem based on a real-world application related to user-defined urban AOIs. In particular, we first define the visualization tasks using a real estate dataset as an example. After describing the visualization idioms, we illustrate why a novel footprint method is needed to support the visualization.

5.3.1 Visualization tasks

For ease of explanation, we describe our work using Australia’s real estate data (details in Chapter 3) as an example. The data lists 1.42 million properties sold between 2007 and 2016. Each property is associated with 72 dimensions including its geo-spatial information (latitude and longitude), categorical dimensions (e.g., property type), and numerical dimensions (e.g., price, distance to the nearest train station or supermarket).

Recall Example 5.1, each user query results in a user-defined urban AOI, and such AOI will change when the user refines her individual requirements. Therefore, our main visualization task is to visually present the user-defined urban AOI and allow users to gradually refine their requirements, in which case the visualization will need to be updated without latency.

5.3.2 Visualization idioms

We use a straightforward composite visualization design to visualize the user-defined AOIs. As shown in Figure 5.4, the design includes two coordinated views: a multidimensional view and a Google maps view.

Multidimensional view. As shown in Figure 5.4(a-1), a multidimensional view allows users to define the urban areas of their interest in an interactive way. Users can first select attributes (i.e., dimensions) of interest from the selection box on top of the view. Suppose that a user selects m attributes, then m coordinated histograms will be initialized to present the data distribution on the m selected dimensions. Users can filter out the dataset by directly brushing on top of each histogram. After the user conducts filtering (selection) on top of one dimension, all histograms of the remaining dimensions will be refreshed accordingly.

Google maps view. As discussed in Section 5.2.1, we adopt the most straightforward polygonal representation to visualize the urban AOI. As shown in Figure 5.4(a-2), the AOI can include multiple regions, and each region is visualized as a polygon on the Google maps. It is worth noting that, holes could also exist in real-life applications, and such region is visualized as empty areas inside the polygon (Figure 5.4(a-2)).

5.3.3 Why a novel footprint method is needed?

After users filter out the data from the multidimensional view, the following steps are sequentially executed: (i) the data (POIs) that satisfy user’s query are returned, (ii) the region (AOI) is calculated using a footprint method based on the updated data, and (iii) the calculated region is updated on the Google maps.

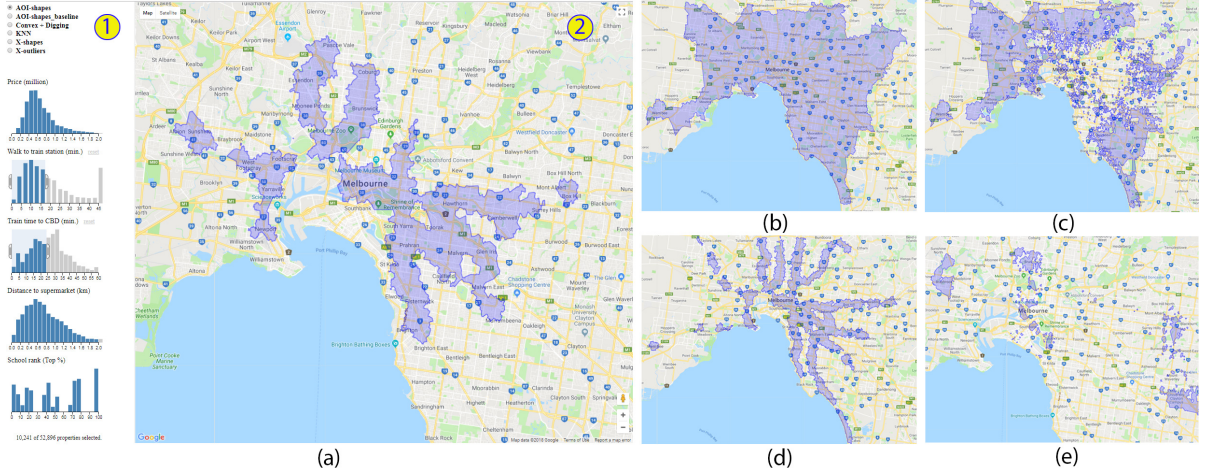


Figure 5.4: The interface and some examples of the visualization of user-defined urban areas of interest. (a) The interface, which includes (1) the multidimensional view, where users can define their queries based on multiple attributes; and (2) the Google maps view, where the user-defined AOI is shown on top of the map. Examples of the results: (a) the area in Melbourne within a 15-minute walk to the nearest train station and a 25-minute train to the CBD; (b) the Melbourne metropolitan area; (c) the Melbourne metropolitan area with house prices less than 1 million dollars; (d) the Melbourne metropolitan area within a 15-minute walk to the nearest train station; (e) the Melbourne metropolitan area with house prices less than 1 million dollars and in a top-10% public secondary school zone.

While the first step can be effectively implemented using relational database management system (RDBMS), and the last step should not be a problem if we only need to draw a limited number of regions on Google maps, so the main bottleneck is at the second step, which is the main technical contribution of our work.

Both the effectiveness and efficiency could be a problem when we use existing footprint methods to calculate the urban AOIs. On the one hand, as we have summarized in Table 5.1, existing methods suffer from at least one of the drawbacks related to detecting multiple regions, outliers and holes, as well as parameter-tuning. On the other hand, although existing algorithms (e.g., χ -shapes [Duckham et al., 2008]) can achieve the time complexity of $O(n \log n)$, it will take more than one second if the data size is larger than 200k (for more details, see Section 5.8 in the experiment part), which is considered as high latency for an interactive visualization application [Card et al., 1991]. Therefore, a novel footprint method is needed to solve both the effectiveness and efficiency problems of existing methods.

5.4 A baseline method

In this section, we propose a baseline method to overcome the drawbacks of existing footprint methods (as summarized in Table 5.1). Specifically, the proposed baseline method is (i) parameter-free: users do not need to tune any parameter for the AOI, (ii) separable: both the data points that satisfy the user query and those that do not are considered to generate the AOI, and (iii) multiple regions, outliers and holes can be recognized.

5.4.1 Preliminaries

Given a set of data points D , let $P \subseteq D$ be the points satisfying the user query (i.e., the points inside the user-defined AOI), and $\bar{P} = D \setminus P$ be the points outside the AOI. Our aim is to compute (i) the boundary of the regions that are formed by the points in P and (ii) the outliers (i.e., isolated neglecting points Zhong and Duckham [2016] that are outside the region boundaries).

5.4.1.1 Main idea of the proposed baseline method

The main idea of our proposed baseline method is to separate the regions that satisfy the user query from those that do not (Figure 5.1(d)). The method consists of an offline preprocessing step and a query time processing step. In the preprocessing part, we build a Delaunay triangulation (DT) [Samet, 2006] based on the whole dataset (D). In the query processing part, once the user issues a query, the set (P) of points that satisfy the query will be obtained. Then, the following steps will be executed to find the region boundary of P .

Step 1. Finding boundary edges based on the DT result.

Step 2. Recognizing different regions based on the boundary edges.

Step 3. Detecting holes.

5.4.1.2 Labels of vertices and edges

To better illustrate the algorithm, we first define the following labels of vertices (points) and edges with the examples shown in Figure 5.5.

For points (i.e., vertices), we divide them into the following types based on whether they are in P and whether they will form a boundary:

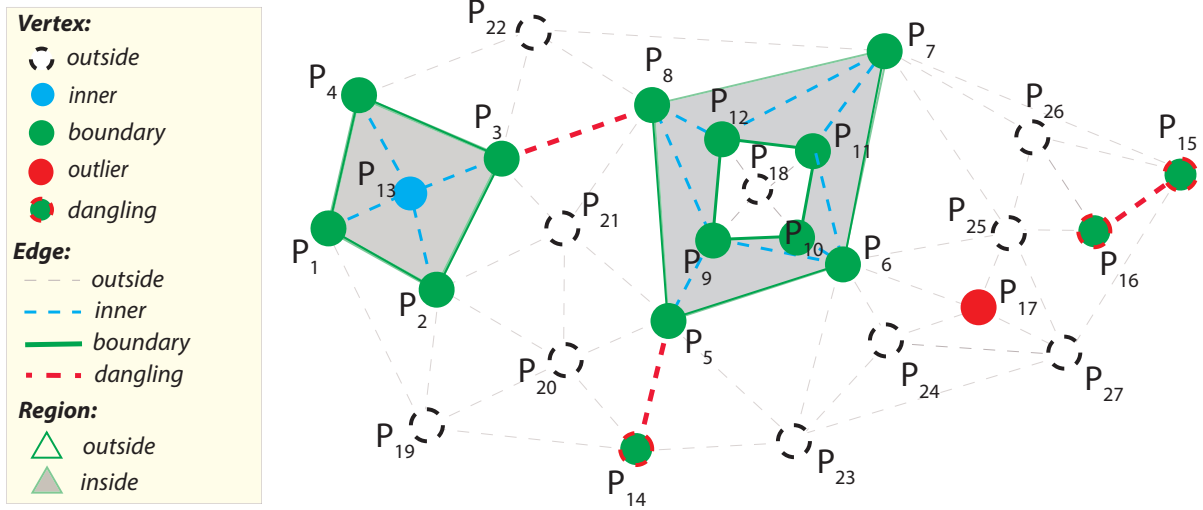


Figure 5.5: Illustration of different types of points, edges and regions. (1) p_1 – $p_{17} \in P$: p_1 – p_{12} are boundary points, where p_9 – p_{12} form a hole boundary; p_{13} is an inner point; p_{14} – p_{16} are dangling points; p_{17} is an outlier point; (2) p_{18} – $p_{27} \in \bar{P}$ (i.e., they are outside points).

- **Outside/inside.** p is an *outside* point if $p \in \overline{P}$. p is an *inside* point if $p \in P$.
- **Boundary.** A *boundary* point p satisfies the user query and is on the final boundary of a region (i.e., the resulting footprint). For example, p_1 – p_4 are *boundary* points which form a disclosed region as $p_1p_2p_3p_4$ in Figure 5.5.
- **Inner.** An *inner* point p satisfies the user query (i.e., $p \in P$) but is inside the final boundary of a region. E.g., p_{13} is an *inner* point.
- **Outlier.** A point p is an *outlier* if $p \in P$ but p is not connected with any other points in P . E.g., p_{17} is an *outlier* point.
- **Dangling.** A point p is a *dangling* point if (1) $p \in P$, and (2) p is not a *inner*, *boundary*, or *outlier* point; i.e., p connects to at least one point p' , where $p' \in P$ but any third points p'' that form a triangle with p and p' are *outside* points.

It is worth noting that, a point can either be an *outside* or an *inside* point. *Inside* points will be further recognized as *boundary*, *inner*, *outlier* points or *dangling* points during the algorithm process. Similar to the labels of vertices, edges are also divided into the following labels:

- **Outside/Inner.** pp' is an *outside* (*inner*) edge if either p or p' is an *outside* (*inner*) point. E.g., p_1p_{19} , $p_{23}p_{24}$ and $p_{12}p_{18}$ are *outside* edges; p_1p_{13} is an *inner* edge.
- **Boundary.** pp' is a *boundary* edge if both p and p' are *boundary* points. E.g., p_1p_4 and p_9p_{10} are *boundary* edges. It is worth noting that, a *boundary* edge always connects two different regions: an *outside* and an *inside* region.
- **Dangling.** Suppose that p_m forms a triangle with pp' , and p_n forms another triangle with pp' . pp' is a *dangling* edge if and only if $p \in P$, $p' \in P$, and $p_m \in \bar{P}$, $p_n \in \bar{P}$. E.g., p_3p_8 , p_5p_{14} and $p_{15}p_{16}$ are *dangling* edges.

5.4.2 The proposed baseline algorithm

Algorithm 5.1 presents the whole process of the baseline method. At the pre-processing step, we build a Delaunay triangulation based on the whole dataset (D) (Lines 3 - 4), and then initialize the labels of all triangulation edges as *outside* (Lines 5 - 6). At the query processing step, the set of points P is obtained based on the user query. We initialize the lists of *boundary* edges, *dangling* edges, and *outlier* points (Line 7); and for each point (p) satisfying the user query ($p \in P$), we label it as *inside*. Then the following steps are executed.

Baseline step 1: finding boundary edges. We traverse all the triangles in the DT result to find the boundary edges (Lines 10 - 30). On the one hand, an edge of a triangle will be labelled as a *boundary* edge (Line 23) if two of the three points of the triangle are in P , and the third point in \bar{P} . It is worth noting that, if the edge to be labelled has already been labelled as a *boundary*, then it becomes a *dangling* edge (Lines 19 - 21). On the other hand, a point (p) will be labelled as *outlier* if p is the only point in a triangle that is in P and p is not a *boundary* point (Lines 28 - 30). Figure 5.6 illustrates how we recognize *boundary* edges, *dangling* edges, and *outlier* points from the triangulation result.

Baseline step 2: recognizing multiple regions. For *dangling* and *outlier* points, they are all treated as *outliers* (individual points) in the final visualization (Line 46). For *boundary* edges, they will form multiple disclosed regions.

Lemma 5.1. Let G be the (undirected) graph formed by all the *boundary* edges. Then, every vertex of G has even degree.

Proof 5.1. For any vertex v in G , if $\text{DEGREE}(v)=1$, as shown in Figure 5.7(a), then vv' becomes a *dangling* edge; so, $\text{DEGREE}(v) \geq 2$. Now, suppose that the degree of v was an odd

Algorithm 5.1: BASELINE (D, P)

```

1 Input: A set  $D$  of geographical points, and a set  $P$  which are within the user-defined urban AOI (i.e., satisfy the
   user query). Each  $d \in D$  has attributes {latitude, longitude, label}, and  $d.label = outside$ .
2 Output: A list of regions (Regions) and a list of outliers (Outliers)
3 DELAUNEY_TRIANGULATION( $D$ ) ; ▷ Start of pre-processing
4  $S_T \leftarrow$  Triangle set of the DT result;
5 for each edge  $e \in S_T$  do
6   |  $e.label \leftarrow outside$ ;
7  $BE, DE, OP \leftarrow \emptyset$  ; ▷ Start of query processing
8 for each point  $p \in P$  do
9   |  $p.label \leftarrow inside$ ;
10 for each triangle  $t$  in  $S_T$  do
11   |  $tp_{in} \leftarrow \emptyset$ ; // triangle points in  $P$ 
12   | for each point  $p$  in  $t$  do
13   |   | if  $p.label \neq outside$  then
14   |   |   | Add  $p$  to  $tp_{in}$ ;
15   |   |   | if  $p.label = outlier$  then
16   |   |   |   | Remove  $p$  from  $OP$ ;
17   | if  $tp_{in}.length = 2$  then
18   |   |  $e \leftarrow Edge(tp_{in}[0], tp_{in}[1])$ ;
19   |   | if  $e.label = boundary$  then
20   |   |   | Delete  $e$  from  $BE$ ;
21   |   |   | Add  $e$  to  $DE$ ;
22   |   | else
23   |   |   | Add  $e$  to  $BE$ ;
24   |   |   |  $e.hole\_check\_point \leftarrow$  the incenter of  $t$ ;
25   |   |   |  $e.label \leftarrow boundary$ ;
26   |   |   |  $tp_{in}[0].label \leftarrow boundary$ ;
27   |   |   |  $tp_{in}[1].label \leftarrow boundary$ ;
28   | else if  $tp_{in}.length = 1$  &  $tp_{in}[0].label \neq boundary$  then
29   |   |  $tp_{in}[0].label \leftarrow outlier$ ;
30   |   | Add  $tp_{in}[0]$  to  $OP$ ;
31  $Regions, edges \leftarrow \emptyset$ ;
32 while  $BE \neq \emptyset$  do
33   |  $e \leftarrow BE[0]$ ;
34   | while  $e \neq null$  do
35   |   | Add  $e$  to  $edges$ ;
36   |   | Remove  $e$  from  $BE$ ;
37   |   |  $e \leftarrow ADJACENT(e, BE)$ ;
38   |   |  $position \leftarrow INTERSECT(e, edges)$ ;
39   |   | if  $position \neq -1$  then
40   |   |   |  $loop \leftarrow edges[position, edges.length-1]$ ;
41   |   |   |  $edges \leftarrow edges - loop$ ;
42   |   |   | Add  $loop$  to  $Regions$ ;
43 for each  $r$  in  $Regions$  do
44   | if  $POINT\_IN\_POLYGON(r[0].hole\_check\_point, r) = True$  then
45   |   |  $r.label \leftarrow hole$ ;
46  $Outliers \rightarrow GET\_OUTLIERS(OP, DE)$ ;
47 RETURN  $Regions, Outliers$ ;

```

number, as shown in Figure 5.7(b), there would be at least one edge in G becoming a *dangling* edge, which contradicts the assumption. Therefore, all vertices of G have even degree.

Lemma 5.2. All the *boundary* edges will form multiple edge-disjoint cycles.

Proof 5.2. Based on the properties of Eulerian circuit, that “an undirected graph can be

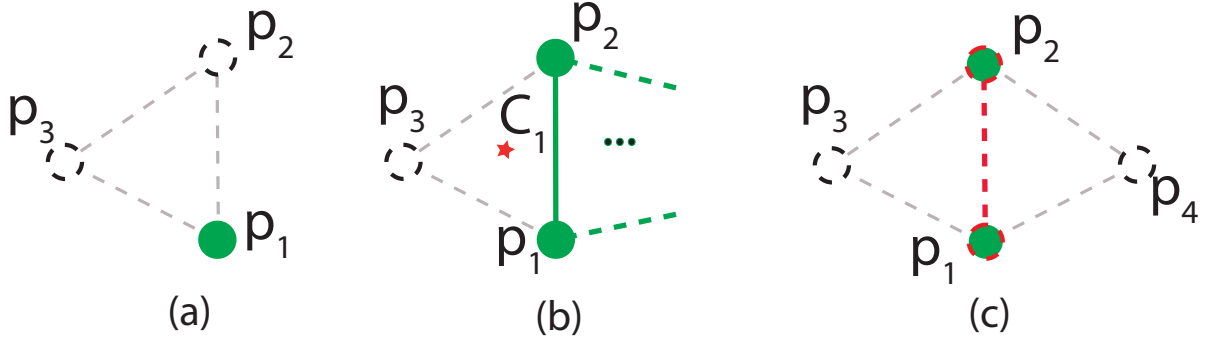


Figure 5.6: Illustration of finding boundary edges using the baseline method. (a) p_1 is labelled as an outlier point since it is not connected with any inside point; (b) p_1p_2 is labelled as a boundary edge since p_3 is an outside point; (c) the label of p_1p_2 is changed from boundary to dangling when p_4 is detected as an outside point.

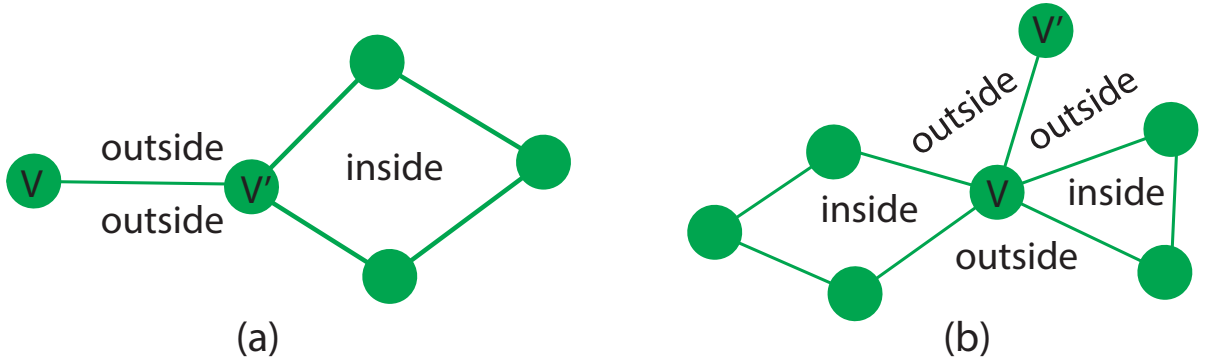


Figure 5.7: Illustration of Proof 5.1: (a) the edge vv' will become a dangling edge if the degree of v is 1, or (b) the degree is any odd number greater than 1.

decomposed into edge-disjoint cycles if and only if all of its vertices have even degree” [West, 2001], Lemma 5.2 is proved.

Definition 1. e' is an **adjacent edge** of e if both e' and e are *boundary edges* which intersect at p .

Based on Lemma 2, we can recognize multiple regions by sequentially connected *boundary edges* via the intersection points. As shown in Lines 32 - 42, we traverse all the boundary edges stored in BE . We first get an edge (e) from BE , and add it to *edges* (Lines 35 - 36). We then find an adjacent edge of e in BE , and store it as the new e (Line 37). We use the function INTERSECT (Line 38) to determine whether the new e intersects with any other edges in BE

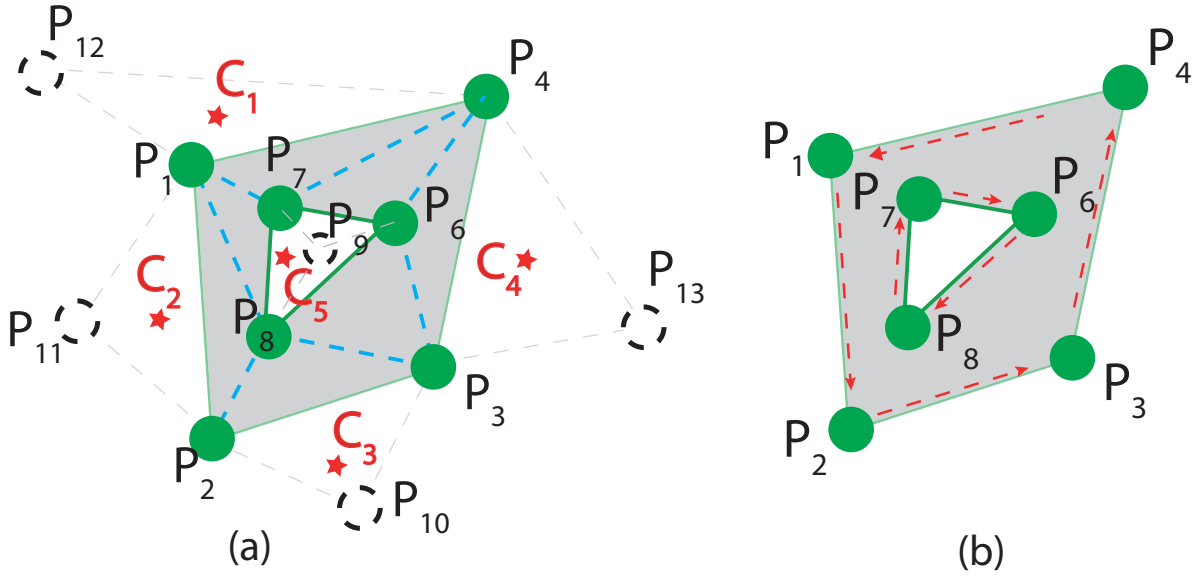


Figure 5.8: Illustration of detecting holes: (a) the baseline method which uses the *hole_check_point*; (b) the AOI-shapes algorithm which is based on the direction of the boundary darts.

(except the last edge). If there is an intersection, then there is a loop. We will remove such loop from the edge list, and add the loop to the final region list (Lines 39 - 42).

Baseline step 3: detecting holes. Until now, we have recognized multiple regions, each of which is presented as a single loop formed by boundary edges. For example, Figure 5.6 will be recognized as three loops (regions) $(p_1p_2p_3p_4, p_5p_6p_7p_8$ and $p_9p_{10}p_{11}p_{12})$. It is worth noting that, the loop might also form a negative region. For example, the loop $p_9p_{10}p_{11}p_{12}$ in Figure 5.6 is a hole. To recognize such hole regions, when we traverse each triangle (t) in DT to recognize *boundary* edges, we store the incenter (i.e., the point forming the origin of a circle inscribed inside the triangle t , which is always inside the triangle t) of t as a *hole_check_point* (Line 23), as shown in Figure 5.6(b). Finally, to detect whether a region is a hole, we only need to check if the *hole_check_point* of one *boundary* edge of the region is inside the region or not. If it is inside, then we label the region as a hole (Lines 43 - 45). For example, in Figure 5.8, when we recognize p_7p_8 as a boundary point based on triangle $p_7p_8p_9$ ($p_7, p_8 \in P$ but $p_9 \in \overline{P}$), we store the incenter of $p_7p_8p_9$ (c_5) as the *hole_check_point* of boundary edge p_7p_8 which is part of the region $p_6p_7p_8$. Since c_5 is inside the region $p_6p_7p_8$, so the region is a hole region.

5.4.3 Time complexity of the baseline method

Suppose that the number of points in D and P are N and n , respectively. For the pre-processing part, the Delauney triangulation costs $O(N \log N)$; since the number of edges in the triangulation result, which is proved to be no larger than $3N$ [Seidel, 1995], the initialization of the labels of points and edges will cost $O(N)$. Therefore, the pre-processing part is $O(N \log N)$. For the query processing part (Algorithm 5.1), finding boundary edges (step 1) needs to traverse all triangles, which is $O(2N)$; recognizing multiple regions need to traverse all boundary edges (and finding an adjacent edge for each boundary edge): it costs $O(s \log s)$, where s is the number of *boundary* edges, and $s \ll n$; detecting holes (step 2) takes a linear $O(m)$, where m is the total number of regions, and $m \ll s$. Therefore, the complexity of the query processing part is $O(N)$.

5.5 The proposed AOI-shapes

In this section, we propose our AOI-shapes method to improve the efficiency of the baseline method by using a modified DCEL structure to store the triangulation result. In this subsection, we first give a brief description of the DCEL data structure, and then explain how DCEL could possibly improve the efficiency of the baseline method. Finally, we describe the proposed AOI-shapes algorithm and analyse the complexity in the end.

5.5.1 The DCEL data structure

Our proposed AOI-shapes algorithm is based on the Doubly Connected Edge List (DCEL) data structure [Preparata and Shamos, 2012], which is widely adopted in computational geometry and solid modelling. DCEL links three sets of records, i.e., vertices (points), darts (edges), and faces (triangles): each point is a vertex, the line connecting two points in the Delaunay triangulation is an edge, and a triangle is a face which consists of three darts. Each edge is represented as two ‘darts’ (half edges) with a counter-clockwise direction.

The following terms will be helpful in explaining our algorithm.

- **Twin dart.** Since an edge is represented by two darts, each one of the two darts will have a twin dart in DCEL. As shown in Fig. 5.9, the twin dart of $\langle p_1, p_5 \rangle$ in face $f_{1,5,6}$ is $\langle p_5, p_1 \rangle$ in face $f_{1,2,5}$.

Algorithm 5.2: AOI-SHAPES(D, P)

```

1 Input & Output: Same as Algorithm 5.1.
2 Lines 3 - 6 in Algorithm 5.1 ; ▷ Start of pre-processing
3 Build the DCEL structure based on  $S_T$ ;
4 Same as Lines 9 - 11 in Algorithm 5.1. ; ▷ Start of query processing
5 for each point  $p \in P$  do
6    $Darts_p \leftarrow \text{DARTS\_START\_WITH}(p)$ ;
7    $outlier\_flag \leftarrow \text{True}$ ;
8   for each dart  $dt \in Darts_p$  do
9     if  $\text{END\_VERTEX}(dt).label = \text{inside}$  then
10       $outlier\_flag = \text{False}$ ;
11       $dt' \leftarrow \text{TWIN\_DART}(dt)$ ;
12      if  $\text{OPPOSITE\_VERTEX}(dt).label = \text{inside}$  then
13        if  $\text{OPPOSITE\_VERTEX}(dt').label = \text{inside}$  then
14           $dt.label = \text{inner}$ ;
15        else
16           $dt.label = \text{in\_boundary}$ ;
17          Add  $dt$  to  $BE$ ;
18      else
19        if  $\text{OPPOSITE\_VERTEX}(dt').label = \text{inside}$  then
20           $dt.label = \text{out\_boundary}$ ;
21        else
22           $dt.label = \text{dangling}$ ;
23          Add  $dt$  to  $DE$ ;
24      if  $outlier\_flag = \text{True}$  then
25        Add  $p$  to  $OP$ ;
26 for each edge  $e \in BE$  do
27   if  $e.obtuse = \text{True}$  then
28      $BE \leftarrow \text{DIGGING}(e, BE)$ ;
29 Same as Lines 33 - 38 in Algorithm 5.1;
30  $e \leftarrow \text{ADJACENT\_DCEL}(e, BE)$ ;
31 Same as Lines 40 - 44 in Algorithm 5.1;
32 for each  $r$  in  $Regions$  do
33   if  $\text{DIRECTION}(r) = \text{clockwise}$  then
34      $r.label \leftarrow \text{hole}$ ;
35 RETURN  $Regions, Outliers$ ;

```

- **Opposite vertex.** The opposite vertex of a dart is the other vertex that is within the same face with it. The opposite vertex of $\langle p_5, p_1 \rangle$ is p_2 (Fig. 5.9). They share face $f_{1,2,5}$.
- **Start/end vertex.** A dart is a half-edge from a start vertex to an end vertex. As shown in Fig. 5.9, the start vertex of $\langle p_5, p_1 \rangle$ is p_5 , and its end vertex is p_1 .

The following functions can be effectively implemented using the basic DCEL data structure.

- $\text{TWIN_DART}(d)$, which returns the twin dart of dart d .
- $\text{OPPOSITE_VERTEX}(d)$, which returns the opposite vertex of a given dart d .

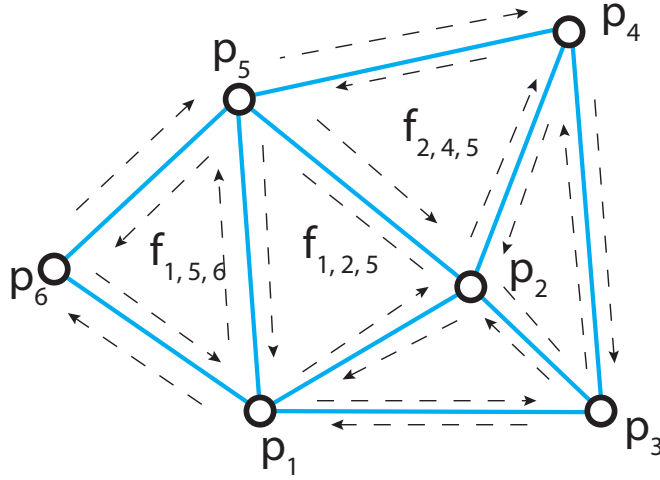


Figure 5.9: Illustration of the DCEL structure, where there are 5 faces, 20 darts (half edges), and 6 vertices.

- $\text{START_VERTEX}(d)$, which returns the start vertex of d .
- $\text{END_VERTEX}(d)$, which returns the end vertex of d .

We also implement the following functions based on DCEL for efficiency purposes (details will be explained in later sections).

- $\text{DARTS_START_WITH}(v)$, which returns all the darts starting from a given vertex v .
- $\text{OBTUSE}(d)$, which returns *True* if the opposite angle of d is obtuse.

5.5.2 Why DCEL structure?

We adopt the DCEL structure to store the DT result in our proposed AOI-shapes because DCEL can link all three sets of records, i.e., edges, vertices, and faces. DCEL could possibly improve the efficiency of the baseline method (i.e., Algorithm 5.1) from the following aspects:

- For step 1 of Algorithm 5.1, where we find the *boundary* edges by traversing all the triangles (Lines 10 - 30): since DCEL links vertices and triangles (faces), for a give P , we only need to access those triangles that are connected with p ($p \in P$). Also, since DCEL links edges (darts) and faces, we can directly determine whether a candidate *boundary* edge is a *dangling* edge or not (Section 5.5.3.1).

- Since DCEL links vertices and edges (darts), the INTERSECT function (Line 38) can be implemented in a more efficient way (Section 5.5.3.2).
- Since darts (edges) have direction information in DCEL, a hole can be more easily detected without storing *hole_check_point* (Section 5.5.3.3).
- With DCEL, a DIGGING function (which makes the final shape more “fit”) would be efficient to implement (more details, see Section 5.5.3.4).

5.5.3 Constructing the AOI-shapes

Algorithm 5.2 presents the complete process of the proposed AOI-shapes method. Its input and output are the same as the baseline, and it also includes two steps: an offline pre-processing part and an online query processing part. The pre-processing part is the same as the baseline except that we build the DCEL structure after calculating the DT (Line 3). At the query processing step, besides those three steps in the baseline method, the AOI-shapes method has an extra step: a “digging” process which makes the final boundary more “fit” to the region. Next, we illustrate how we implement each step of the AOI-shapes method using DCEL.

5.5.3.1 Step 1: finding boundary edges

In this first step, we try to find all the boundaries, i.e., *boundary* edges, *dangling* edges and outliers. In our baseline method (Algorithm 5.1), we traverse all the triangles in the resulting DT to find such boundaries. It is worth noting that, all the *boundary* edges and *dangling* edges connect two vertices within the user query (P), and the outliers are also within (P). Since DCEL could connect the vertex set and the edge (dart) set, we could possibly obtain all the *boundary* edges, *dangling* edges, and outliers, by traversing only those triangles that are connected with p ($p \in P$), instead of all the triangles in DT.

As shown in Algorithm 5.2, we traverse all the points that satisfy the user query (P) (Line 5). For each point (p), we first find all the darts ($Darts_p$) which have p as the origin point (Line 6). For each dart dt ($dt \in Darts_p$), we check whether its end point is in P (to find candidate *boundary* edges and *dangling* edges). On the one hand, if there is no such dart (i.e., the end point of all darts in $Darts_p$ is in \overline{P}), then p is an *outlier* point (Lines 7, 10, 24 - 25). On the other hand, for dt whose end point is also in P , suppose that the twin dart of dt is dt' . We then determine whether d is a *boundary*, *dangling* or *inner* dart based on the opposite points of both dt and dt' . As shown in Figure 5.10, there are a total of four situations:

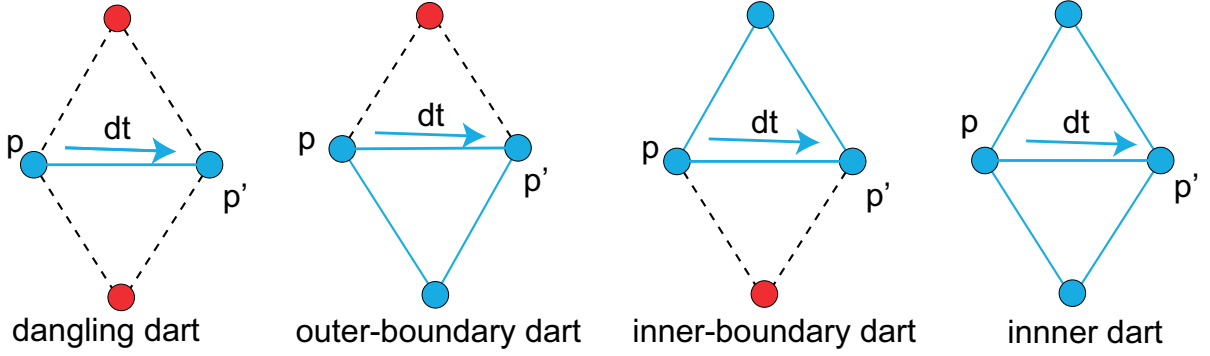


Figure 5.10: Illustration of step 1 (finding boundary edges) of the AOI-shapes algorithm.

- If both the opposite points of dt and dt' are in P , then dt is an *inner* dart (Lines 12 -14, Figure 5.10(a)).
- If only one of the the opposite points of dt and dt' is in P and the other one is in \bar{P} , then dt is a *boudary* dart (Lines 15 - 20, Figure 5.10(b, c)).
- If the opposite points of both dt and dt' are in \bar{P} , then dt is a *dangling* dart (Lines 21 - 23, Figure 5.10(d)).

It is worth noting that the twin dart of an *inner-boundary* dart is an *outer-boundary* dart, and vice versa. For storage efficiency purposes, we only need to store one of them. In our algorithm, we store the *inner-boundary* darts (Line 17); all *boundary* darts refer to *inner-boundary* darts in the rest of the chapter, unless specified otherwise.

5.5.3.2 Step 2: recognizing multiple regions

The second step is to recognize multiple regions from the *boundary* darts. The whole process is similar to the baseline method: we connect those *boundary* darts via the intersection vertices. Our AOI-shapes method improves the efficiency of the baseline in the following aspect. In the baseline method (Line 39, Algorithm 5.1), we find one of the *adjacent* edges as the next connected edge using the function `ADJACENT()`, which traverses all the edges in BE to find an adjacent edge. In the AOI-shapes method (Line 30), with DCEL, we first find the end vertex (v_1) of the current edge (dart) e , and then use the function `DARTS_START_WITH()` to find all the darts (*darts*) starting from v_1 . Then, for each dart (dt) in *darts*, if it is in BE , we will use it as the next edge e .

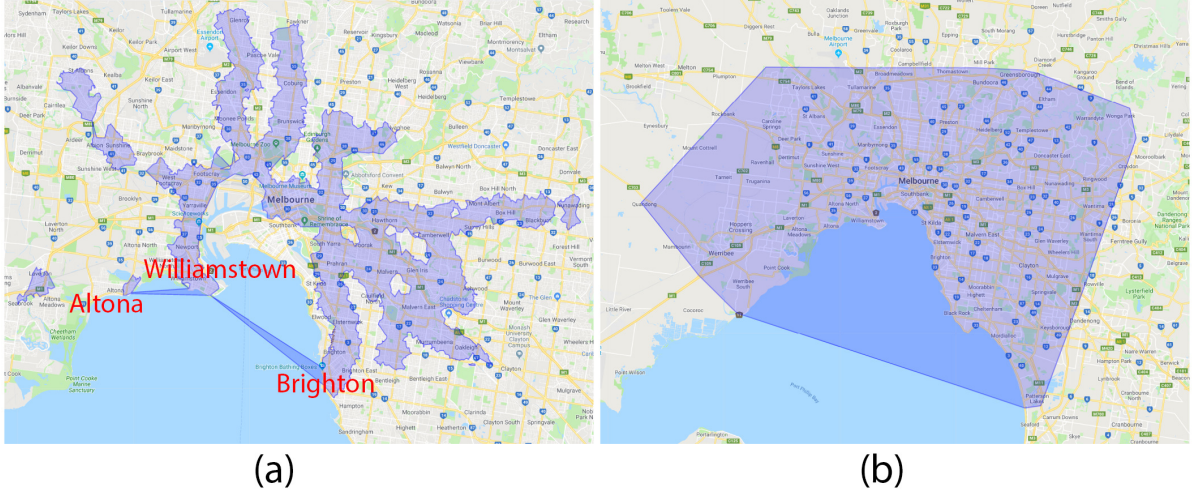


Figure 5.11: Illustration of the problem of the shapes generated from the baseline method. (1) The AOI of properties within a 15-min walk to train stations and a 30-min train ride to the city; (2) the AOI of all properties within the Melbourne Metropolitan area.

5.5.3.3 Step 3: detecting inner holes

In the baseline method, we detect whether or not a region is a hole based on the *hole_check_point*. In the AOI-shapes method, since darts in DCEL have directions, each region formed by the *boundary* darts will be either clockwise or counter-clockwise. As mentioned in Section 5.5.3.1, all the *boundary* darts stored from step 1 are *inner-boundary* darts. Therefore, all the non-hole regions should have the same direction as the DCEL face (counter-clockwise). Apparently, hole regions will have an opposite direction (clockwise), as shown in Figure 5.8(b). Lines 32 - 34 in Algorithm 5.2 demonstrate how we recognize the regions as hole regions based on the direction of the region (which is formed by *boundary* darts).

5.5.3.4 A “digging” process

Until now, the proposed AOI-shapes will return the same results as the baseline method. One problem of the resulting shape is: the region boundaries are formed at the process where we separate those points in \bar{P} , so if the set of \bar{P} is very “small”, or at the worst case, $\bar{P} = \emptyset$, then the resulting shape from the baseline method could be too “big” to characterize the boundary of the region. Example 5.4 illustrates such a problem.

Example 5.4. We present two examples in Figure 5.11 using the baseline method. In Figure 5.11(a), we can see that, since there are no data points in \bar{P} in the water area, the region between

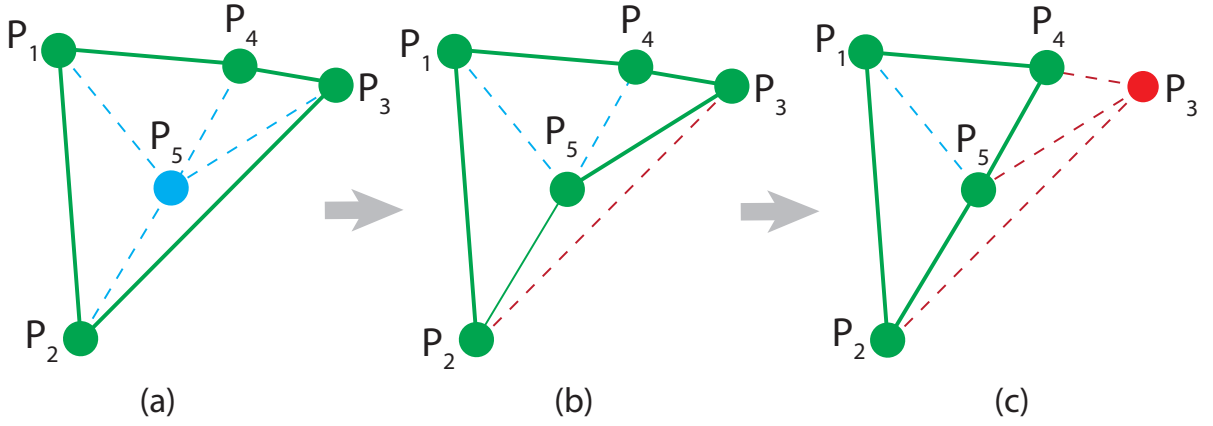


Figure 5.12: Illustration of the “digging” process to construct the AOI-shapes. (a) The original boundary darts p_1p_2 , p_2p_3 , p_3p_4 , p_4p_1 , and an inner point p_5 . (b) Since $\angle p_2p_5p_3$ is an obtuse angle, the “digging” process replaces the boundary dart p_2p_3 with p_2p_5 and p_5p_3 , and p_2p_3 becomes a dangling dart. (c) Since $\angle p_5p_4p_3$ is obtuse, the boundary dart p_5p_3 will be replaced by p_5p_4 and p_4p_3 ; however, since p_4p_3 was originally an outer-boundary dart, both p_5p_3 and p_4p_3 will become dangling darts and p_3 becomes an outlier point.

Altona and Williamstown, and the region between Williamstown and Brighton are unexpectedly connected. In Figure 5.11(b), where \bar{P} is \emptyset , the resulting shape is the convex hull of $D(P)$.

Existing work solves the above problem from two aspects. On the one hand, χ -shapes [Duckham et al., 2008] and Park and Oh [2012a] use a “digging” process which progressively replaces a “long” boundary edge with two “shorter” inner edges (as we have introduced in Example 5.2); however, this needs a parameter (which is hard to define) to determine whether a boundary edge is “short” enough to terminate the “digging” process. On the other hand, Peethambaran and Muthuganapathy [2015] introduce a parameter-free method which defines complex regularity and circle constraints to construct the “footprint” shape; however, the process will introduce an extra $O(N \log N)$ time complexity.

Here, we combine the advantages of those two kinds of methods, and introduce a simple “digging” process. We consider a boundary dart is too “long” if the opposite angle of the dart is obtuse; then we will replace the dart with another two darts (Lines 26 - 28). Fig. 5.12 illustrates the “digging” processing. Example 5.5 presents the results of the AOI-shapes method compared to the baseline.

Example 5.5. Figure 5.4(a) presents the AOI-shapes result using the same user input as that in Figure 5.11(a) (baseline). We can see that those unexpected connections are removed in the

“digging” process. Figure 5.4(b) presents the AOI-shapes result with the same empty \overline{P} as that in Figure 5.11(b), we can see that, the AOI-shapes is more accurate.

5.5.4 Time complexity of the AOI-shapes method

Compared to the baseline, the AOI-shapes method has an extra process to build the DCEL structure in the pre-processing part. Since the time of building the DCEL index depends on the number of edges in the triangulation result ($O(3N)$), the time complexity of the whole pre-processing part is still $O(N \log N)$, where N is the number of points in D . For the query processing part in Algorithm 5.2, we only need to traverse the points in P , whose complexity depends on the darts being visited: since each edge corresponds to two darts, finding boundary edges (step 1) has a complexity of $O(6n)$, where n is the number of points in P . In the worst case, if $\overline{P} = \emptyset$, $n = N$, the cost of step 1 is $O(6N)$. Similar to the baseline, the second and third steps cost $O(s \log s)$ and $O(m)$, respectively. The digging process needs to traverse all *boundary* darts, and in average it has a time complexity of $O(s)$; in the worst case, if all *inner* darts are changed to *boundary* darts, it will cost $O(6n)$. Therefore, the average time complexity of the query processing part is $O(n)$, and in the worst case, the complexity is $O(N)$.

5.6 The incremental AOI-shapes

AOI-shapes improves the efficiency of existing footprint methods by building a global DT at the pre-processing part. In this section, we propose two incremental methods to further improve the efficiency of the AOI-shapes by reusing the calculations at the query process part of the previous user query. The first incremental method reuses the labels of boundary edges (result of Step 1, Section 5.5.3.1), and the second method reuses the final region results from the previous AOI-shapes.

5.6.1 Incremental method I

Algorithm 5.3 presents the pseudocode for the first incremental method. We first calculate a difference set, and store the points set as P_{delete} and P_{new} . We then update the labels of all points in P_{delete} as *outside* and all points in P_{new} as *inside* (Line 8). Then, for each point in P_{delete} and P_{new} (Lines 9 - 12), we store all the following darts whose labels will be possibly changed: (1) all the darts connected with p (whose end point is also an *inside* point), and (2) each dart (dt) whose opposite point is p , and its twin dart (dt').

Algorithm 5.3: INCRE_AOI-SHAPES_1 (P_{new}, P_{pre})

```

1 Input: the new point set  $P_{new}$ , the point set  $P_{pre}$  from the previous user query.
2 Output: A list of footprint regions  $Regions$ .
3  $P_{delete} \leftarrow P_{pre} - P_{new}$ ;
4  $P_{add} \leftarrow P_{new} - P_{pre}$ ;
5 if  $P_{new}.size \leq P_{delete}.size + P_{add}.size$  then
6   | RETURN AOI-SHAPES( $D, P_{new}$ );
7  $Dt \leftarrow \emptyset$ ;
8 Update the label of points in  $P_{delete}$  and  $P_{add}$ ;
9 for each point  $p \in P_{delete} \cup P_{add}$  do
10  | for each dart  $dt \in (OPPOSITE\_DARTS(p) \cup START\_VERTEX(p))$  do
11  |   |  $dt' \leftarrow TWIN\_DART(dt)$ ;
12  |   | Add  $dt$  and  $dt'$  to  $Dt$ ;
13 for each dart  $dt \in Dt$  do
14  | Lines 12 - 23 in Algorithm 5.2;
15 CHECK_OUTLIERS();
16 Lines 26 - 34 in Algorithm 5.2;
17 RETURN  $Regions, Outliers$ ;
```

For each dart in $Darts$, we use the same condition check (shown in Figure 5.10) to label them as *dangling*, *boundary*, *inner* or *outside* (Lines 13 - 14). Specifically, (1) if a dart d is changing from *dangling* to *outside*, we check whether one of the points connected with d becomes an *outlier*; (2) if a new point does not connect with any darts whose end point is also in P_{new} , then the new point becomes an *outlier* point (Line 15).

After finding the new boundary edges, the incremental algorithm uses the same steps to dig the boundary edges, recognize multiple regions and detect inner holes as the AOI-shapes algorithm (Line 16), and finally gets the region and outlier lists (Line 17).

It is worth noting that, If the number of points in the new point set (n) is greater than the summation of the number of points to be added and the number of points to be deleted, then we will directly consider the new point set to generate a new footprint using the AOI-shapes method (Lines 5-6).

5.6.2 Incremental method II

Here, we reuse the final boundary and outlier information from the previous user query, and update each individual point one by one. Algorithm 5.4 presents the whole process. Same as Algorithm 5.3, we first calculate a difference set, and store the point set as P_{delete} and P_{new} (Line 2). Adding a new point might result in multiple existing separated regions connected, and deleting a point might result in a region splitting into multiple regions. To avoid a newly merged region being split in the removal process (Lines 3 - 10), we remove points first before

Algorithm 5.4: INCRE_AOI-SHAPES_2 (P_{new}, P_{pre})

```

1 Input & output: Same as algorithm 5.3.
2 Same as Lines 3 - 6 in algorithm 5.3.
3 while  $P_{delete} \neq \emptyset$  do
4   while  $\exists p : p \in P_{delete} \& p.label = boundary$  do
5     REMOVE_BOUNDARY_POINT( $p$ );
6   while  $\exists p : p \in P_{delete} \& p.label = dangling$  do
7     REMOVE_LINE_POINT( $p$ );
8   while  $\exists p : p \in P_{delete} \& p.label = outlier$  do
9     REMOVE_OUTLIER( $p$ );
10  REMOVE_INSIDE_POINT( $p$ );
11 while  $P_{add} \neq \emptyset$  do
12   while  $\exists p : p \in P_{add} \& p.label = transition$  do
13     while  $\exists p : TRANSITION(p).label = outlier$  do
14       ADD_OUTLIER( $p$ );
15     while  $\exists p : TRANSITION(p).label = dangling$  do
16       ADD_DANGLING_POINT( $p$ );
17     while  $\exists p : TRANSITION(p).label = boundary$  do
18       ADD_BOUNDARY_POINT( $p$ );
19 RETURN  $Regions, Outliers$ ;
```

we add new points (Lines 11 - 18).

5.6.2.1 Removing existing points

Removing an *outlier* point (e.g., p_{15} in Figure 5.13) does not need further operations. Removing a *dangling* point (e.g., p_{13} in Figure 5.13) might result in the other connected *dangling* point (p_{14}) becoming an *outlier* point.

To remove a *boundary* point p , we only need to find the connected vertices of p , and consider the following: (1) if no connected vertices are *inside* points, and there are more than three points in the hull boundary, then we only need to remove p , and connect the other two vertices as a new *boundary* edge (e.g., removing p_2 from Figure 5.13(a)); (2) if p connects with an *inside* point, then the *inside* point is labelled as *boundary*, and will be connected with the other two vertices to make two new *boundary* edges (e.g., removing p_7 from Figure 5.13(a)); (3) if there are only two *boundary* vertices left for the affected hull after p is removed, the two *boundary* vertices will become *dangling* points (e.g., removing p_{10} from Figure 5.13(a)).

It is worth noting that, removing a *boundary* point (p) may split a hull into multiple disconnected components. In such a case, we re-label all the affected darts (those darts connect p and those darts that have p as opposite points), then we connect those *boundary* edges (related to p) and get the region boundary.

Removing an *inner* point p will result in either holes or *dangling* edges. On the one hand,

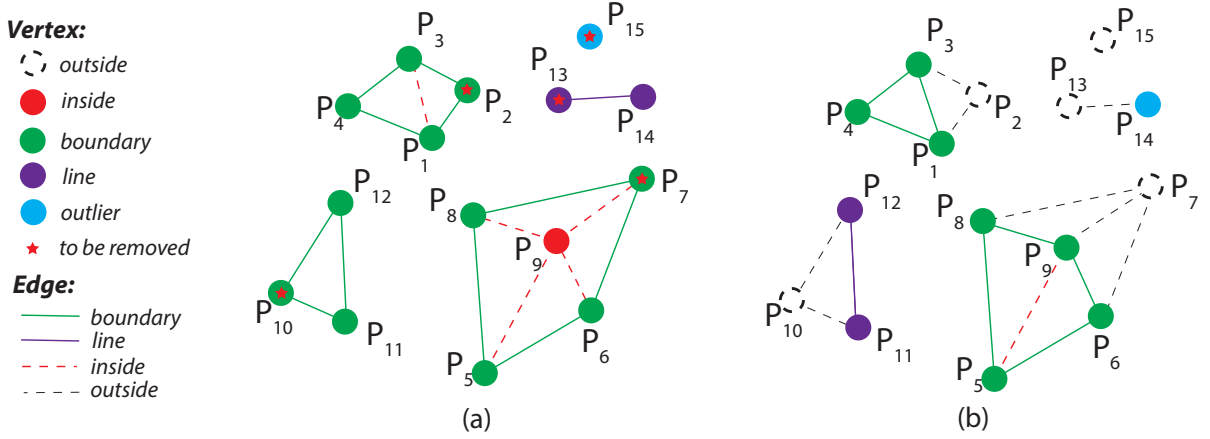


Figure 5.13: Example of removing boundary points (p_2 , p_7 , and p_{10}), line points (p_{13}) and outlier points (p_{15}): (a) the original footprints, with the points to be removed labelled with stars; (b) the result after removing the starred points.

if all the opposite darts of p are *inside* darts, there will be a hole formed by those opposite darts. On the other hand, if one or more of the opposite darts of p are *boundary* darts, those *boundary* darts will become *dangling* darts, and there might be new regions formed by other darts.

5.6.2.2 Adding new points

When adding a new point to the boundary, we need to traverse the current boundary and detect whether there are *boundary* darts connecting with the edge. In order to avoid such a traversal, we add another label for *outside* points: *transition*, and consider those *transition* points first when updating the AOI-shapes.

- **Transition.** A *transition* point is a point in \overline{P} , but that shares a triangle with a *boundary* edge, a *dangling* edge or an *outlier* point (i.e., it is the opposite vertex of the twin dart of a *boundary* or a *dangling* edge, or it directly connects with an *outlier* point).

Since *transition* points are the closest to existing *boundary* edges, *dangling* edges or *outlier* points, we first consider such points. It is obvious that: (1) adding a transition point p' to an *outlier* point p results in a *dangling* edge pp' ; (2) adding a transition point p'' to a *dangling* edge pp' results in a new triangle $pp'p''$ (if the three points are non-collinear): p , p' and p'' will become three *boundary* points.

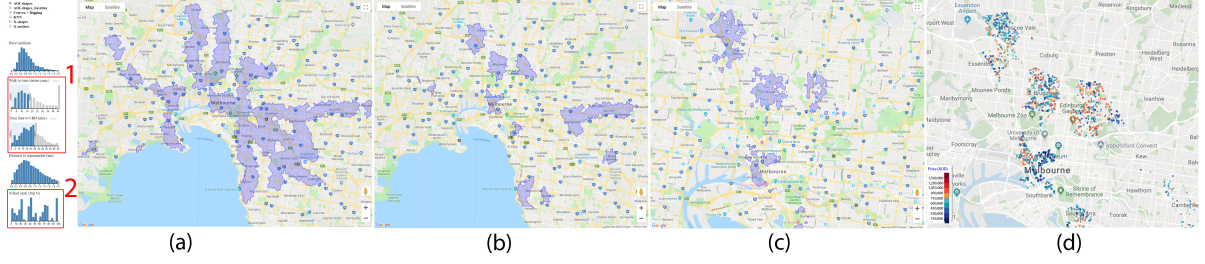


Figure 5.14: *Demonstration of the incremental AOI scenario: (a) the user refines his query from Figure 5.1 by further selecting those regions within 30-min train to the city, (b) the user continues to do filtering on the dimension “school_rank”; (c) the user only wants those areas that have properties within 1 million dollars; (d) individual properties in the AOI are shown to the user.*

A major challenge while adding new points is that multiple disconnected components may become connected, and we need to merge such components. In such a case (i.e., a new point p connects with multiple regions), we re-label all the affected darts (those darts connect p and those darts that have p as opposite points), connect those *boundary* edges (related to p) and obtain the region boundary.

5.6.3 Time complexity of the incremental algorithms

Suppose that, the number of points in the new point set is n , the number of points to be added and deleted are a and d , respectively. Then the time complexity of both two incremental AOI-shapes methods is $O(\text{MIN}(n, a + d))$.

5.7 A demonstration system

We have implemented an online demo system¹ using Australia’s real estate data (Chapter 3) to demonstrate the usefulness of the proposed AOI-shapes for interactive visualizations of urban AOIs. Besides the AOI-shapes, we have also implemented several state-of-the-art alternatives. In the demo, we allow users to switch freely among different methods and compare both the quality and efficiency of them. We also provide a slider for users to change the parameter if the alternative method (e.g., χ -shapes [Duckham et al., 2008]) needs one.

¹<http://aoishapes.com>

Our demonstration is based on the scenario that a user, *John*, tries to find a place to live in Melbourne. Our system helps him explore and understand different urban areas to cater to his own preferences on various dimensions before he tries to find a particular house.

Example 5.6. *John first plays with the system to check where he could buy a house within a budget of one million dollars. He gradually filters the properties on the “price” dimension (Figure 5.14), and the regions on the map view keep changing as he moves the slider. He finds that most of the regions in North and West Melbourne satisfy his query (i.e., he could afford houses in those regions); but if he wants to buy houses in the South and East Melbourne, he might need to live quite far away from the city (where he works).*

Example 5.7. *Living far away from the city is not a big problem since John could take the train. He then issues a new query and selects the region within 15-minute walk to the nearest train station. There are a large number of regions satisfying his requirement, so he refines his requirements by limiting the travel time to the city within half an hour (Figure 5.14(a-1)). As he keeps checking the changes of the region, he notices that the dimension “school_rank” does not follow a Normal distribution (Figure 5.14(a-2)). Considering his child’s education, he continues to refine his query and selects those regions within a top-20% secondary schools (Figure 5.14(b)). After that, he applies his budget-limitation (1 million dollars) again and checks those regions in Figure 5.14(c). Finally, John checks the individual properties within the urban area of his interest (Figure 5.14(d)).*

5.8 Experiments

In this section, we conduct experiments to evaluate the effectiveness and efficiency of our proposed AOI-shapes based on both synthetic and real-world datasets and compared it with the state-of-the-art alternatives shown in Table 5.2. All experiments have been done in JavaScript on a quad-core i7-5500U CPU (2.4GHz) with 8GB RAM.

5.8.1 Quality of the AOI-shapes

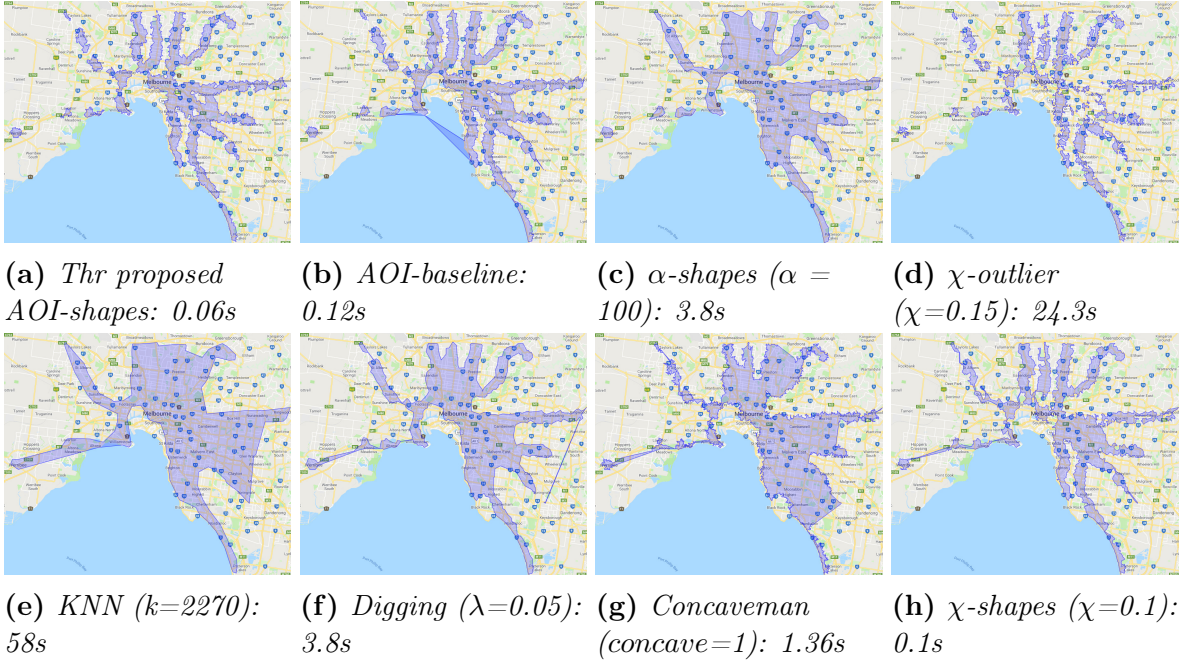
We first evaluate the quality of our proposed AOI-shapes and compare it with the state-of-the-art alternatives (Table 5.2). We present both the qualitative and quantitative comparisons in this subsection.

Since most alternative methods need one parameter, for each method, we have either tried 5-10 different values within the range or chosen the optimal value suggested by the original

Table 5.2: *Alternative methods compared in the experiment.*

	Short name	Parameter range	Default value
KNN-based concave hull [Moreira and Santos, 2007]	kNN	$[3, n]$	$\lceil n/10 \rceil$
Convex+“digging” [Rosén et al., 2014]	digging	$[0,1]$	0.2
ConcaveMan [Park and Oh, 2012a]	ConcaveMan	$[0,]$	1
α -shapes [Radke, 1983]	α -shapes	$[-1,1]$	0.2
χ -shapes [Duckham et al., 2008]	χ -shapes	$[0,1]$	0.1
χ -outliers [Zhong and Duckham, 2016]	χ -outliers	$[0,2]$	0.2
baseline method (Section 5.4)	AOI-baseline	-	-

n : number of points in P

**Figure 5.15:** *Results of the “footprint” generated based on each method and the construction time (P is properties within 15min to the nearest train stations in Melbourne).*

authors. For example, for χ -shapes, the authors suggested that a parameter in $[0.05, 0.2]$ out of the range $[0, 1]$ often provides optimal or near-optimal characteristic shapes; in our experiments, we have used different χ values within the suggested range.

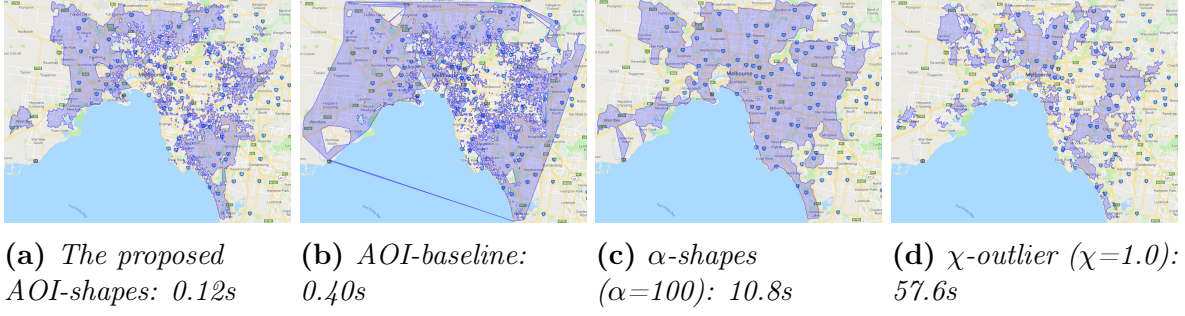


Figure 5.16: Results and construction time of the “footprint” generated based on four methods that is able to recognize multiple regions (P is properties sold within 1 million dollars in Melbourne).

5.8.1.1 Qualitative evaluation

Besides those real-world AOI-visualization scenarios supported by AOI-shapes presented in Chapter 5.7, here we present the footprint results and the construction time for eight different methods, using a user query (properties within a 15-minute walk to the nearest train station) as an example.

As shown in Figure 5.15, the proposed AOI-shapes can recognize multiple regions and inner holes, and is one of the most efficient ones. On the one hand, only those four methods on the top row (AOI-shapes, AOI-baseline, α -shapes and χ -outliers) can recognize multiple regions and outliers. While those four methods present similar results, the construction of AOI-shapes and the AOI-baseline are much more efficient compared to the other two methods (more details are given in section 5.8.1.2). On the other hand, the AOI-shapes and AOI-baseline can detect holes inside the region (e.g., those regions in the middle of two train lines as shown in Figure 5.1(d)); α -shapes with a negative α are also possible for hole detection, however, the sign of α needs to be pre-defined based on whether the final shape has a hole or not (which might be unknown to system designers); all the other five methods cannot detect holes.

We further compare all the methods based on another example, where the AOI query is defined as “regions with properties sold below 1 million dollars”. The result of AOI-shapes (Figure 5.16(a)) reveals that: house prices in most of the areas in South and East Melbourne are above 1 million dollars. Figure 5.15(b) presents similar patterns, but there are unexpected region connections on top of the water, and the region boundary is not optimal. χ -outliers (Figure 5.16(d)) also presents similar results but it takes much longer time (1 minute vs 0.1 seconds) to construct. χ -shapes (Figure 5.16(c)) is not able to recognize the empty areas. Similar to that in Figure 5.15, all the four other methods present similar results to χ -shapes; readers can

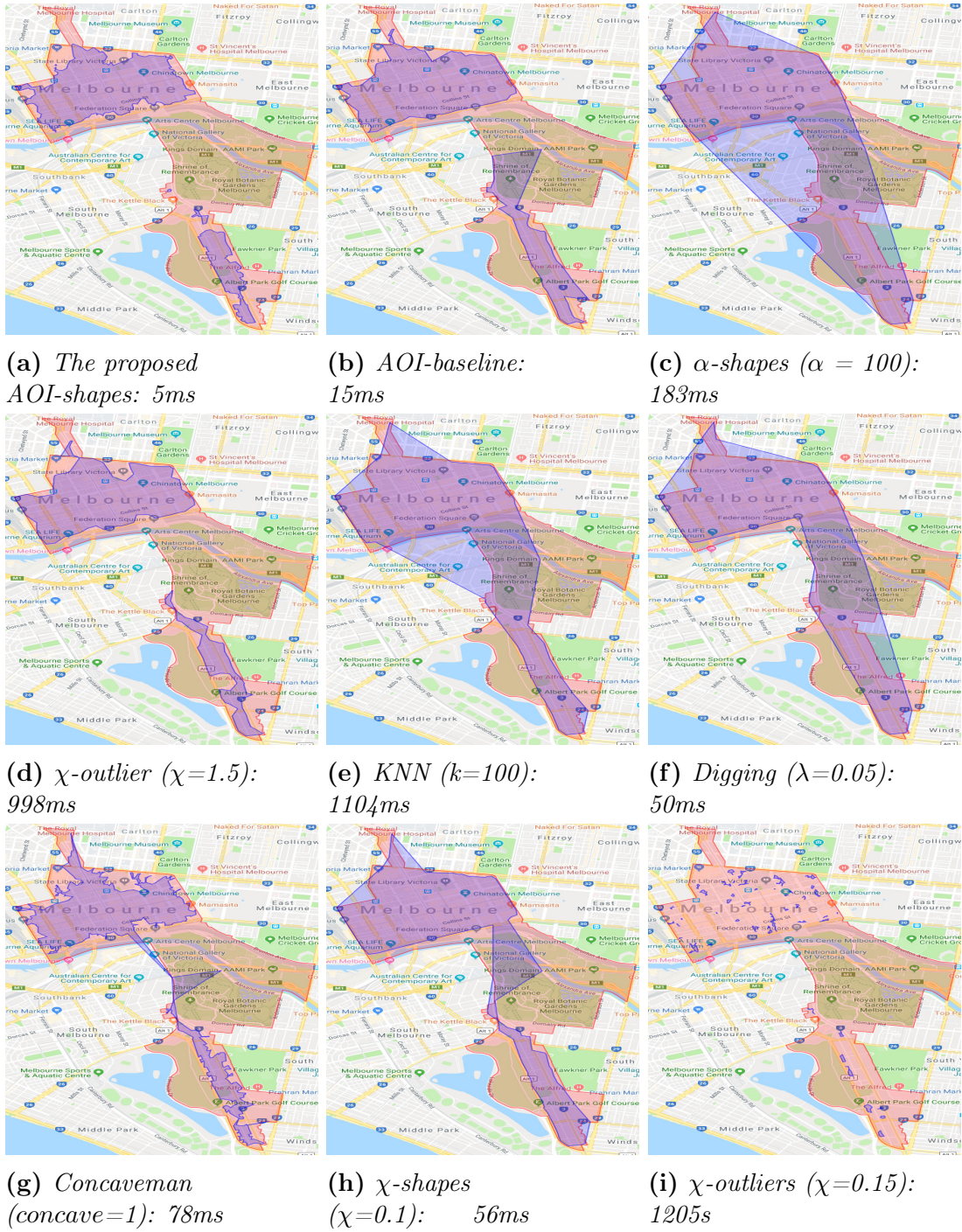


Figure 5.17: Properties in Melbourne CBD (blue: generated footprint; red: ground truth shape).

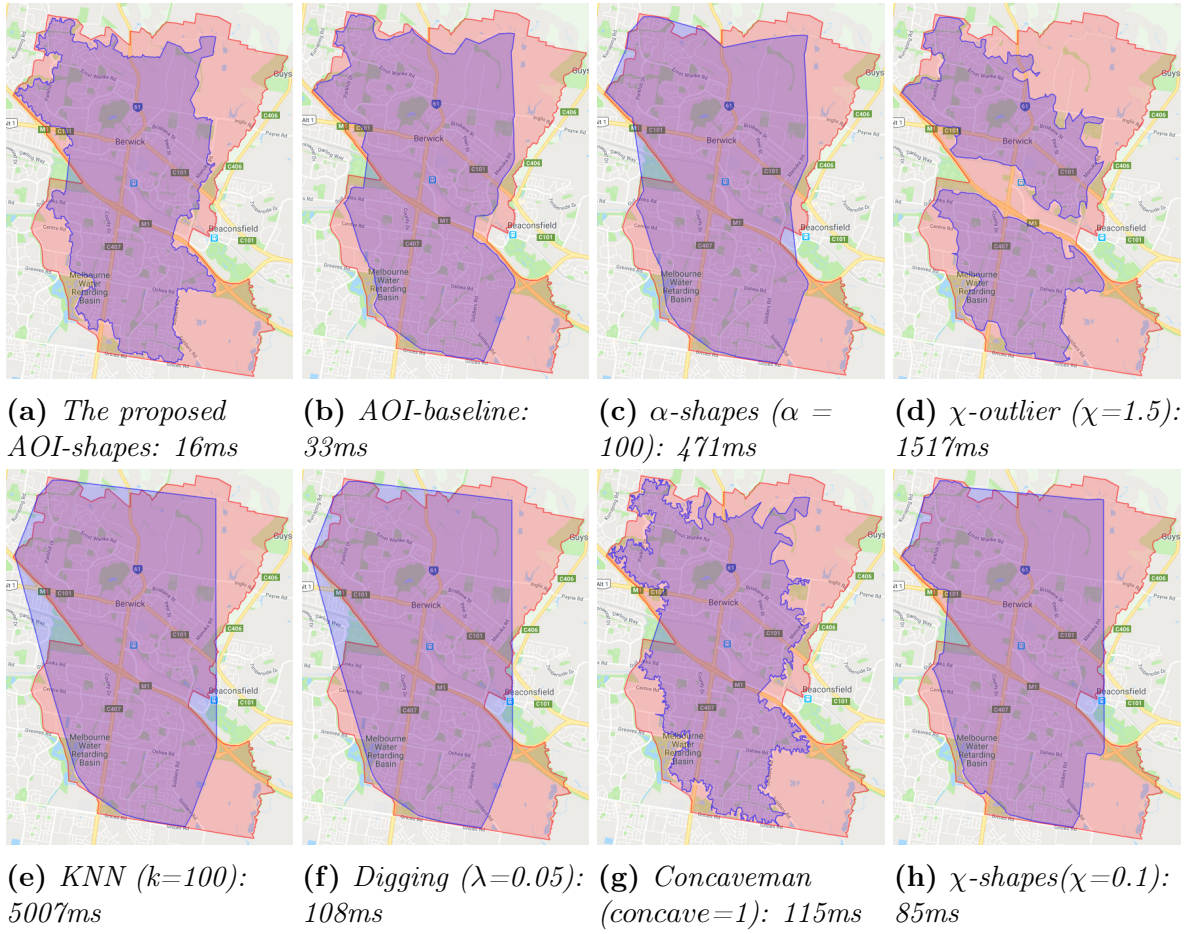


Figure 5.18: Properties in Berwick, Victoria (blue: generated footprint; red: ground truth shape).

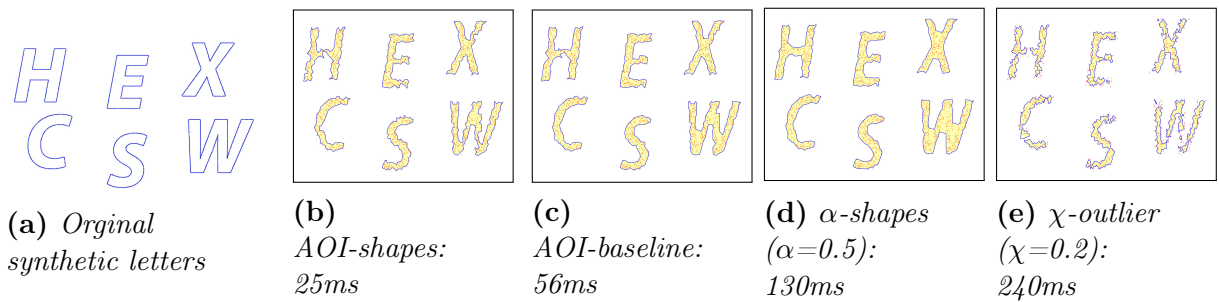


Figure 5.19: Footprint results based on the letter dataset.

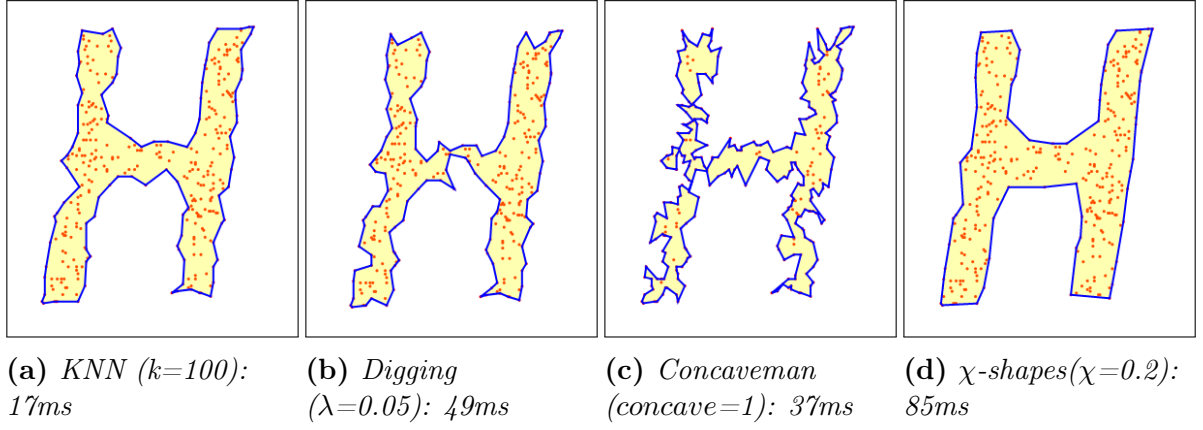


Figure 5.20: Footprint results based on an individual letter “H”.

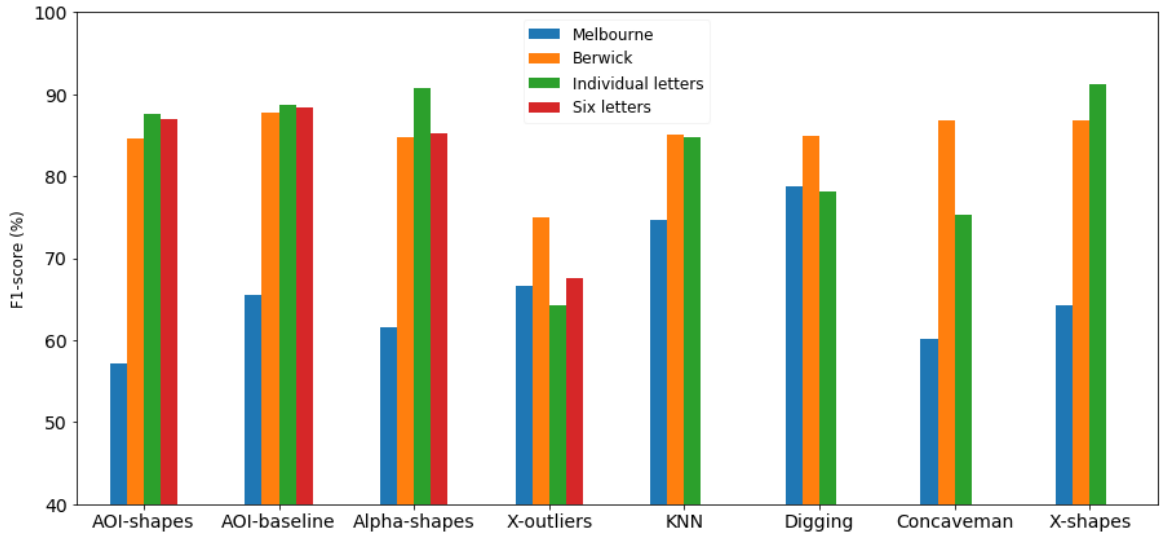


Figure 5.21: Region-based *F1*-score for quantitative evaluation.

check the results by exploring the online demo¹.

5.8.1.2 Quantitative evaluation

Since the boundary of AOI is usually vague and unknown, there is no ground truth of AOI for quantitative evaluation. To quantitatively compare our algorithms with existing methods, we conduct experiments by reconstructing “footprints” derived from known “ground truth” regions.

Experimental setup. Since all the alternative methods cannot detect inner holes and only two of them can recognize multiple regions, the “ground truth” regions are designed to be simple

hulls (no holes). The purpose of this experiment is to compare our methods with state-of-the-art methods regarding constructing simple hulls. Particularly, we designed the following datasets.

- **Dataset 1: administrative districts (Melbourne CBD) + real estate data points.** The first dataset is based on the region of Melbourne CBD and the real estate data (defined in Chapter 3) mapped in Melbourne CBD (1500 properties are included after removing duplicate locations). The “ground truth” shape is the geometry boundaries of Melbourne CBD, and the real estate data in the region form the P .
- **Dataset 2: administrative districts (Berwick) + real estate data points.** Similar to dataset 1, we select the suburb of Berwick (which has one of the highest numbers of properties sold in the regional Victoria, Australia) as the ground truth shape; and the properties mapped in the region are denoted as P (9350 properties).
- **Dataset 3: synthetic letters + synthetic points.** We adopt the letter-based dataset defined in [Zhong and Duckham, 2016], where six hole-free letters were arbitrarily chosen: “H”, “E”, “X”, “C”, “S”, and “W” are placed on a 2×3 grid, which is the “ground truth” shape (Figure 5.19(a)). It is worth noting that, since only the proposed methods and α -shapes with appropriate parameter settings can recognize holes, it is not fair to other methods if we generate letters with holes. We then generate the point set as described in [Zhong and Duckham, 2016], which are P in our algorithm.
- **Dataset 4: individual letter + synthetic points.** Since half of the methods cannot recognize multiple regions, we also generate six datasets with each individual letter as the ground truth shape.

Evaluation metric. Same as existing work (e.g., [Duckham et al., 2008, Asaedi et al., 2017, Zhong and Duckham, 2016]), we use the area-based F1-score, to evaluate the differences of the footprints based on the input points from the “ground truth” shape. The score is calculated as the harmonic mean of area-based precision and recall:

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \times 100\% \quad (5.1)$$

where precision is the proportion of the footprint area that intersects the true shapes, and the recall is the proportion of the true area that is captured by the footprint.

Experimental result. We visualize the results of all generated footprints (compared with the ground truth) in Figure 5.17-5.20 for each of the datasets. The F1-scores are presented in

Figure 5.21. Since our method considers \overline{P} , we can see that almost all the boundaries from AOI-shapes are within the actual ground truth shapes, corresponding to a better precision score. The reasons why our proposed AOI-shapes does not have significant advantages regarding F1-scores (Figure 5.21) include: (1) all the datasets do not have multiple regions and/or inner holes (which are the main strengths of our methods), (2) the ground truth of suburb geometry boundary has a significant proportion of gardens where there are no real estate properties; although the AOI-shapes correctly excludes those regions (shown in Figure 5.17(a) and Figure 5.18(a)), it results as a worse recall score compared to other methods.

It is worth noting that, parameter-free is one of the biggest strengths of the proposed AOI-shapes, since all the alternatives need a parameter which is hard to specify for users. For example, for χ -outliers, the range of parameter χ is $[0, 2]$. We have chosen an optimal χ as 0.15 in Figure 5.15(d) and 0.10 in Figure 5.16(d). However, as shown in Figure 5.17(i), a χ of 0.15 will result that most of the data points become outliers and thus give us multiple small regions; and after comparison of the F1-score, a χ of 1.5 gives us the best result, as shown in Figure 5.17(d) and 5.18(d).

5.8.2 Efficiency of the AOI-shapes

We evaluate the efficiency of the proposed AOI-shapes by comparing the construction time of AOI-shapes with the alternatives (Table 5.2). Since AOI-baseline is the only method whose complexity is based on N instead of n , in our implementation of AOI-baseline, we store a mapping from points to triangles in DT to make the baseline step 1 more efficient, so the AOI-baseline can also have a time complexity of $O(n)$, instead of $O(N)$.

Simulated user query. As shown in Figure 5.4(a), the AOI is often generated based on range queries (for quantitative attributes). Therefore, for efficiency evaluation, we simulated user queries based on the attribute range. For example, for a real estate dataset with 50k properties (Figure 5.22), we first sort the data based on the distance to the nearest train station. Simulated query (P) is selected as the first 2k, 4k, ..., 50k rows of the sorted data. Each of the user queries will correspond to a specific user query (e.g., 20k corresponds to a 15-minute walk to the nearest train station (Figure 5.15)).

For those alternative footprint methods which all need parameters, the choice of parameters might influence their construction time. For example, for χ -shapes, a larger χ will terminate the algorithm earlier than a smaller χ , and thus takes a shorter time to construct the footprint. As discussed in Example 2, the choice of χ will also influence the complexity of the resulted

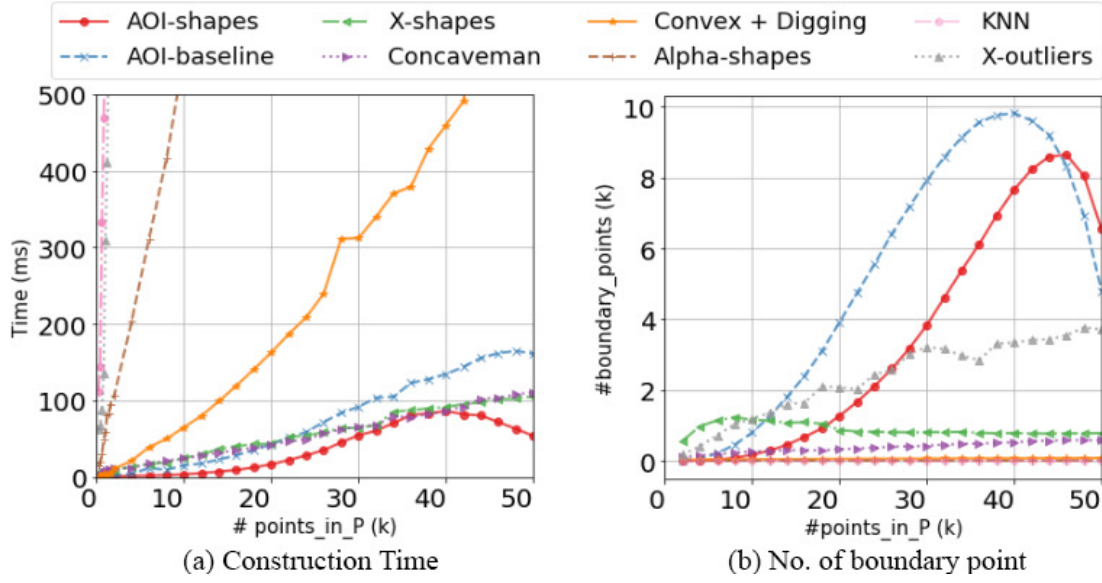


Figure 5.22: 50k real estate dataset in Melbourne (query: distance to train station).

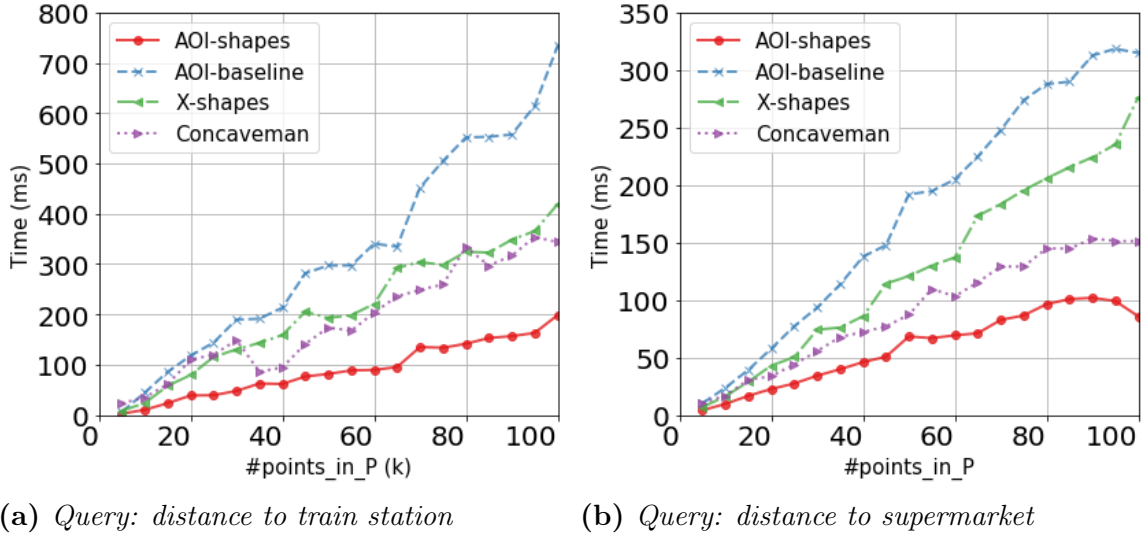


Figure 5.23: Construction time (500k real estate dataset).

shape, which can be measured by the number of boundary points. Our experiment shows that: the AOI-shapes can deliver a more complex shape (with more boundary points) in the least of construction time, compared to the alternatives with default parameters. As shown in Figure 5.22(a), four methods (AOI-shapes, χ -shapes, Concaveman, and AOI-baseline) are the most efficient ones, the construction of which are much shorter than the other four methods; however,

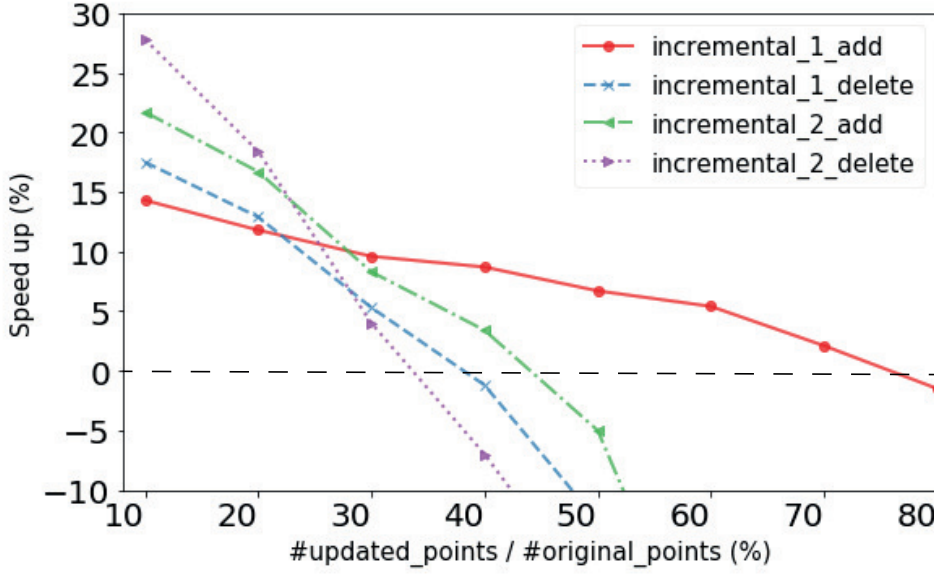


Figure 5.24: Experiment result of the incremental methods (based on 50k properties)

both χ -shapes and Concaveman only generate simple shapes (shown in Figure 5.22(b)). The only method that can generate complex shapes as AOI-shapes and AOI-baseline is χ -outliers; however, its construction time is at least a thousand times more compared to AOI-shapes. For AOI-shapes and AOI-baseline, both the number of boundary points and the construction time decrease after around 40k since different regions might merge. Also, AOI-shapes takes about half of the construction time of AOI-baseline thanks to the DCEL structure, although AOI-shapes has an extra digging process.

Among those four more efficient methods, we further evaluate their scalability on a larger dataset which has 500k real estate properties. The query is constructed based on the distance to the nearest train station (Figure 5.23(a)) and the distance to the nearest supermarket (Figure 5.23(b)). Both results reveal that the AOI-shapes is the most efficient one. Compared to Figure 5.23(a) and (b), we find that with *distance_to_supermarket* as input, the construction time of all footprints are shorter than the one using *distance_to_train_station* as input. The reason is that the resulted region of the query related to the supermarket is less complex with the same number of points in P . Also, if we compare Figure 5.22(a) and Figure 5.23, the size of D has no significant effect on the construction time of AOI-shapes, since only the edges connected with P will be visited for constructing the AOI-shapes (the size of D affects the construction time of the global DT, which is not included here).

5.8.3 Efficiency of the incremental AOI-shapes

We further evaluate and discuss whether the proposed two incremental methods can improve the efficiency compared with the original AOI-shapes. For evaluation purposes, we skip the process of comparing P_{new} and p_{pre} for the incremental methods (Lines 5-3.7 in Algorithm 3 & 4).

Similar to Section 5.8.1.2, we simulated the user query based on data ranges with the real estate data. Here we use the 50k real estate dataset as D . We select five attributes to define the range query: *price*, *distance_to_train_station*, *distance_to_supermakert*, *distance_to_hospital*, *school_rank*. We sort the data based on each attribute, and then choose the first $n\%$ (n is a random number between 10 and 50) of the data as the original P . Then we gradually add or delete data based on P and report whether the two incremental methods will speed up the construction process based on the ratio of updated point number and original point number (the speed-up percentage is average score based on the five chosen attributes).

As shown in Figure 5.24, when there are more than 35%-40% of data to delete, both the incremental methods are worse than the original method. The second incremental method (which updates the final shapes) works well when there are only a few data points (less than 20%) to be updated, while the first incremental method (which improves the efficiency at the labeling process) works better when there are more data points to update. The incremental methods will not speed up the process when there are more than about 35% (incremental_2) and 80% (incremental_1) of data points to be updated. The reason is that the updating process based on the final points (incremental_2) takes much longer time than the original construction process; for incremental_1, more added data points might also result in more original point labels to be revisited.

5.8.4 Discussion

In this subsection, we discuss the strengths and weaknesses of the proposed AOI-shapes, applications and extensions.

Strengths. Based on the experimental results, the proposed AOI-shapes has several advantages over the state-of-art footprint methods. First, all the alternatives need a parameter, which often has no real-life meaning and is hard to define, and the optimal parameter setting for the same method varies for different applications. Second, compared with χ -outliers and α -shapes which possibly present similar results with an appropriate choice of χ , the construction of AOI-

shapes is normally thousands of times faster. Third, compared with the proposed AOI-baseline, AOI-shapes is faster and more accurate on boundary detection (especially when \bar{P} is sparse, as shown in Figure 5.16). Finally, compared to all the other methods, AOI-shapes takes at least a similar time as the most efficient state-of-the-art χ -shapes, but can recognize multiple regions and inner holes.

Limitations. One obvious limitation of AOI-shapes is the need for \bar{P} . Without \bar{P} , the AOI-baseline will return a convex hull (Figure 5.11(b)), while the AOI-shapes might return a shape “better” than the convex hull (Figure 5.4(b)), but there is no guarantee of the quality of the result. While there is no parameter setting in AOI-shapes, other methods such as χ -shapes and χ -outliers might be more robust for applications with no available \bar{P} . One possible extension of the AOI-shapes is to introduce a parameter related to the angle in the “digging” process. Another limitation of the method is the pre-computation part of DT. Since DT computation is based on the global dataset D , more storage space is needed compared to the alternative methods.

Applications. AOI-shapes is more suitable for applications where both P and \bar{P} are available. Besides the real estate domain presented throughout the chapter, there are many other possible applied scenarios of the method. For example, in traffic analysis, those road points that have traffic jams can form a P , while other uncongested points can form a \bar{P} ; for facility deployment, P and \bar{P} can also be easily obtained based on online geo-tags.

Extensions. Based on our experiments, AOI-shapes works well over dataset under 500k records (most of the query can be returned within 0.5s). There are several directions to extend the AOI-shapes to support a dataset that is above 500k and avoid interactive latency. On the one hand, we could divide the global D into subsets based on either the administrative districts (e.g., states, suburbs) or using geospatial data structures such as Quadtree [Samet, 2006], then we can compute and store the DT for each subset of D . On the other hand, only a subset of the AOI might be of interest to the user even when the actual number of points in P is large. For example, the map returned to the user might only contain 10% of the points in P when the map is zoomed in; while when the map is zoomed out, the details of the boundaries might not be important to the user anymore.

5.9 Summary

In this chapter, we studied the problem of visualizing user-defined urban areas of interest (AOIs). First, we summarized the existing “footprint” methods. To address the drawbacks of

the existing studies, we proposed a novel parameter-free footprint method, named AOI-shapes, to generate the region boundary of user-defined urban AOIs to visualize on the map. Specifically, the proposed AOI-shapes is able to recognize multiple regions, outliers and inner holes of the AOI. To effectively update the AOI after users update their queries, we proposed two efficient and scalable algorithms to generate the AOI in an incremental manner. We designed and developed an online system which demonstrates the usefulness of the proposed AOI-shapes for interactive visualizations of urban AOIs. Extensive experiments on both synthetic and real-world datasets proved the effectiveness and efficiency of our proposed methods.

Chapter 6

Conclusion and Future Work

“To the future or to the past, to a time when thought is free, when men are different from one another and do not live alone - to a time when truth exists and what is done cannot be undone: From the age of uniformity, from the age of solitude, from the age of Big Brother, from the age of doublethink - greetings!”

— George Orwell, *Nineteen Eighty-Four*

In this chapter, we first conclude the whole thesis and then present some research directions for future work.

6.1 Conclusion

In this thesis, we studied the problem of visualizing geo-related multidimensional data. We solved three research questions proposed in our introduction, focusing on the visualization design (RQ1) and the scalability problems (RQ2 & RQ3). Specifically,

- In our first research question (Chapter 3), we proposed a visual analytics framework of geo-related multidimensional data using the real estate domain as an example. We designed and developed HomeSeeker¹ which augments existing commercial systems to help users understand the local real estate market, find preferred properties based on their personal preferences and compare properties from the aspects that they are concerned about. We presented a systematic visualization design study following with justifications. Case studies based on real-world datasets demonstrated the usefulness of our system, and users with different levels of knowledge on the market and different requirements (either general home buyers or investors) could benefit from the system.

¹<http://115.146.89.158>

- In the second research question (Chapter 4), we studied the problem of supporting visual analytics for large-scale geo-related multidimensional data. Specifically, we first proposed a cluster-based visualization abstraction of geographic data. Second, we designed a concave hull algorithm which includes geographical points in each cluster as compactly as possible. Third, we proposed a cluster-based data cube (ConcaveCubes) to support interactive response to users' visualized exploration on large-scale geographic multidimensional data. Fourth, we built a demo system² to demonstrate the design of cluster-based geographic visualization. We conducted extensive experiments using real-world datasets and compared ConcaveCubes with state-of-the-art cube-based data structures.
- Our third research question (Chapter 5) focused on the problem of visualizing user-defined urban areas of interest. We first described visualization idioms for visualizing AOIs and illustrated why a novel footprint method is needed. Then we proposed a novel footprint algorithm named AOI-shapes to calculate the geographic shape of disconnected AOIs based on geographic data points. AOI-shapes could be easily extended to incrementally compute the boundary of AOIs after user interaction and avoid the interactive scalability problem. Finally, we developed an online demo³ and conducted extensive experiments to demonstrate both the efficiency and effectiveness of the proposed AOI-shapes algorithm.

6.2 Future work

We propose the following directions for future research regarding the visual analytics problem of geo-related multidimensional data.

1. Data and tasks. We have used Australian's real estate domain as an example to study geo-related multidimensional data. Future research directions include continuing to collect data related to the real estate domain, cleaning the data, and applying the visual analytics framework on other datasets and domains.

- **Data collection and integration.** Our current real estate data include 1.42 million properties in Australia. Each property has 72 attributes falling into one of the five profiles. Most of the data profiles (Section 3.3.2) that we have defined are very common considerations to home buyers from all over the world, such as the information related

²<http://115.146.89.158/ConcaveCubes>

³<http://www.aoishapes.com>

to children’s education, and the distance to work and supermarkets. Regarding data collection and integration, on the one hand, future research can continue to collect other data related to the real estate domain in Australia, such as the crime rate of each region (which indicates the safety of the neighbourhood) and the accessibility to the freeway. On the other hand, our data integration framework can also be applied on the real estate domain in other countries, with some of the factors slightly modified. For example, in China, people might be more familiar with the price of units per m^2 . Certainly, there are different considerations across countries; we believe that our framework can inspire researchers from other countries to work on their real estate data.

- **Data cleaning supported by visual analytics.** This thesis has focused more on the data analytics part, instead of the data cleaning part. However, when playing with the HomeSeeker system, we found that visual analytics might help clean the original dataset. Example 6.1 illustrates the idea.

Example 6.1. *At the property-level exploration, HomeSeeker visualizes each property on maps. Users could filter the properties from the multidimensional view. As shown in Figure 6.1, when we filtered out the properties based on the property type, and only selected “houses”, we found that there was only a single “house” in the Melbourne CBD. We tried to make sense of that and had a hypothesis that Melbourne CBD might only have apartments but not houses. After confirming the hypothesis with real estate agents, we clicked on the single “house”, and found that the “house” is a parking slot for sale but was mistakenly labelled as a “house” (possibly by the real estate agent who uploaded the property information) at the original data source.*

Such kinds of anomalies (in Example 6.1) could be very hard to find without visualization, and even domain experts (e.g., real estate agents) might not remember all those domain-specific knowledge (so it is hard to pre-define all data cleaning rules). Visual analytics can help data wranglers define the rules of cleaning which can be applied on the same kind of anomalies.

- **Other geo-related multidimensional data.** While the process of visual analytics design has been believed to be extremely domain-specific, requiring customization and fine-tuning for any new applications [Parameswaran, 2018, Munzner, 2014], we encourage researchers to apply the visual analytics techniques (including those visualization

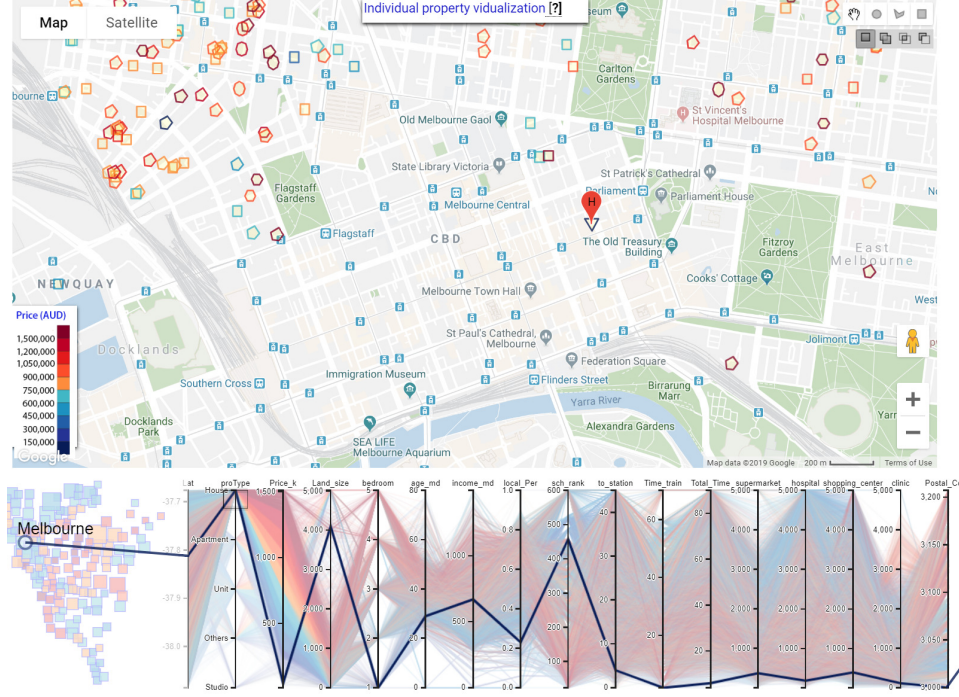


Figure 6.1: Illustration of the single “house” in Melbourne CBD.

designs presented in Section 3.5) on other geo-related multidimensional data (such as the restaurant data, crime data and census data) to solve practical problems.

2. Large-scale data visualization. We have used visualization-based data cubes to solve the scalability problem of large scale geographic visualization. Future works in this direction include the extension of the proposed ConcaveCubes, and the use of other methods to support large-scale visual analytics of geo-related multidimensional data.

- **Extension of ConcaveCubes.** The first extension of ConcaveCubes is to combine geospatial data structures to support faster response after map operations (e.g., zooming and panning). Map operations will change the window size (w). The change of w corresponds to the level of geographic semantics in ConcaveCubes. Therefore, after users apply zooming or panning on the map, we calculate a set subtraction between the current (w_c) and previous (w_p) map windows and then refresh the visualization based on the subtraction result. In our current implementation, we store a minimum bounding rectangle for each geo-boundary to approximately calculate the subtraction process. This process can be improved by indexing the geo-boundaries using geospatial data structures such as R-Tree and Quadtree [Samet, 2006]. Another extension of ConcaveCubes is to adapt the

method with dynamic data. To insert, update or delete a data item in ConcaveCubes after it is constructed, we might need to change the affected nodes in the hierarchical structure.

- **Combining with visualization reduction methods.** ConcaveCubes (as a data reduction method) can be combined with visualization reduction methods (e.g., predictive methods, progressive methods) to support large-scale data visualization. For example, it is possible to predict further user interactions (e.g., whether the user is going to do filtering, map panning or zooming) based on methods defined by Chan et al. [2008] and pre-load some of the visualization results to shorten the interactive latency.

3. Visualization of urban areas of interest. Following our third research questions, we have the following future directions related to footprint methods and the proposed AOI-shapes.

- **Online comparison of existing footprint methods.** In Chapter 5, we have surveyed existing footprint methods and presented both the construction process and the characteristics of them in Table 5.1. While different methods might present different footprint results, there is no standard which one is the best; and different applications might have different requirements. For example, in our second research question (Chapter 4) when we design ConcaveCubes, we need a method that returns a single hull for points in each cluster; while in our AOI visualization problem (Chapter 5), the result might contain multiple regions. Therefore, an online comparison system of footprint methods that allow users to upload their own data could help users choose the most appropriate footprint method in their own application.
- **Extension of AOI-shapes.** There are several directions to extend the AOI-shapes to support larger dataset and avoid interactive latency. On the one hand, the global dataset can be divided into subsets based on either the administrative districts (e.g., states, suburbs) or using geospatial data structures such as Quadtree [Samet, 2006]; then the DT for each subset of D can be computed and stored. On the other hand, only a subset of the AOI might be of interest to the user even when the actual number of points in P is large. For example, the map returned to the user might only contain 10% of the points in P when the map is zoomed in; while when the map is zoomed out, the details of the boundaries might not be important to the user anymore.
- **Application on other domains.** Besides the real estate domain presented in the thesis, there are many other possible applied scenarios of AOI-shapes. For example, in traffic

analysis, those road points that have traffic jams can form a P , while other uncongested points can form a \bar{P} ; for facility deployment, P and \bar{P} can also be easily obtained based on online geo-tags.

4. Evaluations. One of the most common challenges of any visualization research is the need for an in-depth, effective quantitative or qualitative evaluation. Related to this thesis, the following user studies might be useful for future research.

- **Comparing cluster maps with other geoVis methods.** In Section 4, we have presented boundary-based cluster maps to support cluster-based geographic visualization. We have discussed both the advantages and disadvantages of cluster maps compared with other geoVis methods such as heatmaps, binned plots; those statements could be the hypotheses of a well-design user study. We encourage future researchers to conduct formal user studies and compare those geographic visualization methods including but not limited to heatmaps, dot maps, bubble maps, Choropleth maps, Voronoi diagrams, binned plots, bubble-based cluster maps and boundary-based cluster maps.
- **User study of visual analytics systems.** While visual analytics systems always involve human interactions, it is a challenge to conduct user study and check how the system can help users with their original purposes. Taking our proposed HomeSeeker system as an example, we have defined several levels of visual exploration in the system. A proper laboratory user study might help to answer whether those explorations help the user with their home-seeking purposes and if it does, how much it might shorten users' time compared to the use of other systems.

5. Combination of other research fields. Another future direction to study the geo-related multidimensional data is to combine visual analytics with other research fields such as virtual reality and deep learning. On the one hand, augmented reality techniques might help users to understand both the data patterns in a visual analytics system and check environment-based information near the geo-location [ElSayed et al., 2013]. For example, in the real estate domain, visual analytics systems in a virtual reality environment can help the user understand the real estate market in a region while they check the surrounding environment of the region (e.g., whether the region has tree-lined streets). On the other hand, visual analytics might help to make the process of a black-box deep learning algorithm available for users. For example, we might design a multilayer perceptron (MLP) deep learning algorithm to predict house prices.

A proper visual analytics design will reveal the parameter learning process so that humans can change the parameters in the middle of the learning process.

6. Automatic visualization design for geo-related multidimensional data.

The problem of automatic visualization (i.e., visualization ranking/recommendations) has attracted the attention of data scientists in multiple areas [Luo et al., 2018, Wongsuphasawat et al., 2016, Bouali et al., 2016]. Different techniques have been proposed to address the problem, including Rank-by-Feature [Seo and Shneiderman, 2006], diversification [Mafrur et al., 2018], etc. However, most of the existing work only focus on simple charts such as bar charts and line charts. When users have a geo-related multidimensional dataset, they might be wondering what kind of map designs (i.e., heat maps, choropleth maps) they should use and on which dimensions they should apply the method (e.g., should they use the colour to represent house prices or property types). Therefore, there is a need to design an automatic visualization system for geo-related multidimensional data.

Bibliography

- C. Ahlberg and B. Shneiderman. Visual information seeking: tight coupling of dynamic query filters with starfield displays. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 313–317, 1994. Cited on pages 28 and 66.
- F. Akdag, C. F. Eick, and G. Chen. Creating polygon models for spatial clusters. In *International Symposium on Methodologies for Intelligent Systems*, pages 493–499, 2014. Cited on pages 94 and 98.
- H. Alani, C. B. Jones, and D. Tudhope. Voronoi-based region approximation for geographical information retrieval with gazetteers. *International Journal of Geographical Information Science*, 15(4):287–306, 2001. Cited on pages 74, 75, 100, and 102.
- A. M. Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216–219, 1979. Cited on page 99.
- D. F. Andrews. Plots of high-dimensional data. *Biometrics*, pages 125–136, 1972. Cited on pages 21 and 22.
- A. Arampatzis, M. van Kreveld, I. Reinbacher, C. B. Jones, S. Vaid, P. Clough, H. Joho, and M. Sanderson. Web-based delineation of imprecise regions. *Computers, Environment and Urban Systems*, 30(4):436–459, 2006. Cited on pages 74, 75, 100, and 102.
- G. A. Areas. GADM database of global administrative areas, 2012. Cited on page 87.
- S. Asaeedi, F. Didehvar, and A. Mohades. α -concave hull, a generalization of convex hull. *Theoretical Computer Science*, 702:48–59, 2017. Cited on pages 3, 65, 75, 101, 102, and 130.
- Australian Bureau of Statistics. Australian Statistical Geography Standard (ASGS): volume 1 - main structure and greater capital city statistical areas, 2016a. Cited on pages 38 and 87.

- Australian Bureau of Statistics. Census of population and housing: Nature and content, Australia, 2016b. Cited on page 87.
- R. Beecham, C. Rooney, S. Meier, J. Dykes, A. Slingsby, C. Turkay, J. Wood, and B. Wong. Faceted views of varying emphasis (FaVVEs): a framework for visualising multi-perspective small multiples. *Computer Graphics Forum*, 35(3):241–249, 2016. Cited on pages 2, 42, and 50.
- A. A. Bojko. Informative or misleading? heatmaps deconstructed. In *HCI*, pages 30–39, 2009. Cited on pages 7 and 64.
- F. Bouali, A. Guettala, and G. Venturini. VizAssist: an interactive user assistant for visual data mining. *Visual Computer*, 32(11):1447–1463, 2016. Cited on page 143.
- C. Brewer and M. Harrower. Colorbrewer 2.0: color advice for cartography. <http://colorbrewer2.org/>, accessed March 2017. Cited on page 46.
- C. A. Brewer, A. M. MacEachren, L. W. Pickle, and D. Herrmann. Mapping mortality: evaluating color schemes for choropleth maps. *Annals of the Association of American Geographers*, 87(3):411–438, 1997. Cited on pages 65 and 88.
- T. Brinkhoff, H.-P. Kriegel, R. Schneider, and A. Braun. Measuring the complexity of polygonal objects. In *International Workshop on Advances in Geographic Information Systems*, pages 109–117, 1995. Cited on page 78.
- A. Bykat. Convex hull of a finite set of points in two dimensions. *Information Processing Letters*, 7(6):296–298, 1978. Cited on page 99.
- R. J. Campello, D. Moulavi, and J. Sander. Density-based clustering based on hierarchical density estimates. In *PAKDD*, pages 160–172, 2013. Cited on page 81.
- Z. Cao, S. Wang, G. Forestier, A. Puissant, and C. F. Eick. Analyzing the composition of cities using spatial clustering. In *ACM SIGKDD International Workshop on Urban Computing*, pages 1–8, 2013. Cited on page 98.
- S. K. Card, G. G. Robertson, and J. D. Mackinlay. The information visualizer, an information workspace. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 181–186, 1991. Cited on pages 27, 89, and 105.

- S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999. Cited on page 19.
- D. B. Carr, R. J. Littlefield, W. Nicholson, and J. Littlefield. Scatterplot matrix techniques for large N. *Journal of the American Statistical Association*, 82(398):424–436, 1987. Cited on pages 21, 22, 23, and 67.
- E. Celikten, G. Le Falher, and M. Mathioudakis. Modeling urban behavior by mining geotagged social data. *IEEE Transactions on Big Data*, 3(2):220–233, 2017. Cited on page 98.
- R. Chadnov and A. Skvortsov. Convex hull algorithms review. In *Russian-Korean International Symposium on Science and Technology*, pages 112–115, 2004. Cited on page 99.
- S.-M. Chan, L. Xiao, J. Gerth, and P. Hanrahan. Maintaining interactivity while exploring massive time series. In *VAST*, pages 59–66, 2008. Cited on pages 28 and 141.
- T. M. Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete and Computational Geometry*, 16(4):361–368, 1996. Cited on page 99.
- A. Chaudhuri, B. Chaudhuri, and S. Parui. A novel approach to computation of the shape of a dot pattern and extraction of its perceptual border. *Computer Vision and Image Understanding*, 68(3):257–275, 1997. Cited on pages 100 and 102.
- J. Chen, E. C. Hui, and Z. Wang. Perceived risk, anticipated regret and post-purchase experience in the real estate market: the case of China. *Housing Studies*, 26(03):385–402, 2011. Cited on page 1.
- E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *SIGKDD*, pages 1082–1090, 2011. Cited on page 87.
- M. C. Chuah and S. G. Eick. Information rich glyphs for software management data. *IEEE Computer Graphics and Applications*, 18(4):24–29, 1998. Cited on page 22.
- J. E. Cohen and C. Small. Hypsographic demography: the distribution of human population by altitude. *Proceedings of the National Academy of Sciences*, 95(24):14009–14014, 1998. Cited on page 46.
- Department of the Prime Minister and Cabinet of Australia. Australian government public data policy statement, 2015. Cited on page 1.

- M. Drosou and E. Pitoura. Disc diversity: result diversification based on dissimilarity and coverage. *Proceedings of the VLDB Endowment*, 6(1):13–24, 2012. Cited on pages 28, 64, and 66.
- M. Duckham, L. Kulik, M. Worboys, and A. Galton. Efficient generation of simple polygons for characterizing the shape of a set of points in the plane. *Pattern Recognition*, 41(10):3224–3236, 2008. Cited on pages 9, 65, 74, 75, 78, 94, 95, 100, 102, 105, 118, 123, 125, and 130.
- T. Ebert, J. Belz, and O. Nelles. Interpolation and extrapolation: comparison of definitions and survey of algorithms for convex and concave hulls. In *IEEE Symposium on Computational Intelligence and Data Mining*, pages 310–314, 2015. Cited on pages 73, 74, and 95.
- W. F. Eddy. A new convex hull algorithm for planar sets. *ACM Transactions on Mathematical Software*, 3(4):398–403, 1977. Cited on page 99.
- B. Elias. Extracting landmarks with data mining methods. In *International Conference on Spatial Information Theory*, pages 375–389, 2003. Cited on page 97.
- N. A. ElSayed, C. Sandor, and H. Laga. Visual analytics in augmented reality. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 1–4, 2013. Cited on page 142.
- M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *SIGKDD*, pages 226–231, 1996. Cited on pages 69, 81, and 103.
- D. Fisher. Incremental, approximate database queries and uncertainty for exploratory visualization. In *IEEE Symposium on Large Data Analysis and Visualization*, pages 73–80, 2011. Cited on page 27.
- M. Foth. Urban informatics beyond data: media architecture, placemaking, and citizen action. In *Proceedings of the ACM First International Workshop on Understanding the City with Urban Informatics*, page 19, 2015. Cited on page 23.
- M. Friendly. Statistical graphics for multivariate data. In *SAS SUGI*, pages 1157–1162, 1991. Cited on page 52.

- Y. Fu and B. Liu. The impact of community safety on house ranking. In *the 2016 SIAM International Conference on Data Mining*, pages 459–467, 2016. Cited on page 34.
- Y. Fu and H. Xiong. A discovery system for finding high-value homes. In *the 15th IEEE International Conference on Data Mining Workshop*, pages 1612–1615, 2016. Cited on page 34.
- Y. Fu, Y. Ge, Y. Zheng, Z. Yao, Y. Liu, H. Xiong, and J. Yuan. Sparse real estate ranking with online user reviews and offline moving behaviors. In *IEEE International Conference on Data Mining*, pages 120–129, 2015a. Cited on page 34.
- Y. Fu, G. Liu, S. Papadimitriou, H. Xiong, Y. Ge, H. Zhu, and C. Zhu. Real estate ranking via mixed land-use latent models. In *the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 299–308, 2015b. Cited on page 34.
- A. Galakatos, A. Crotty, E. Zgraggen, C. Binnig, and T. Kraska. Revisiting reuse for approximate query processing. *Proceedings of the VLDB Endowment*, 10(10):1142–1153, 2017. Cited on page 27.
- A. Galton and M. Duckham. What is the region occupied by a set of points? In *GIScience*, pages 81–98, 2006. Cited on pages 73, 74, 77, 99, 100, and 102.
- S. Gao, K. Janowicz, and H. Couclelis. Extracting urban functional regions from points of interest and human activities on location-based social networks. *Transactions in GIS*, 21(3): 446–467, 2017. Cited on page 98.
- A. Gheibi, M. Davoodi, A. Javad, F. Panahi, M. Aghdam, M. Asgaripour, and A. Mohades. Polygonal shape reconstruction in the plane. *IET Computer Vision*, 5(2):97, 2011. Cited on pages 100 and 102.
- P. Godfrey, J. Gryz, and P. Lasek. Interactive visualization of large data sets. *IEEE Transactions on Knowledge and Data Engineering*, 28(8):2142–2157, 2016. Cited on page 7.
- R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132–133, 1972. Cited on pages 95, 98, 101, and 102.
- J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: a relational aggregation operator generalizing group-by, cross-tab,

- and sub-totals. *Data mining and knowledge discovery*, 1(1):29–53, 1997. Cited on pages 28 and 66.
- H. Guo, Z. Wang, B. Yu, H. Zhao, and X. Yuan. TripVista: triple perspective visual trajectory analytics and its application on microscopic traffic data at a road intersection. In *IEEE PacificVis*, 2011. Cited on pages 2 and 26.
- M. D. Hansen. Combining parallel coordinates and histograms, 2013. US Patent App. 13/931,785. Cited on page 48.
- J. Heer, B. Shneiderman, and C. Park. A taxonomy of tools that support the fluent and flexible use of visualizations. *ACM Queue*, 10(2):1–26, 2012. Cited on page 20.
- R. L. Hemminger. Line digraphs. In *Graph Theory and Applications*, pages 149–163. Springer, 1972. Cited on pages 21 and 22.
- P. Hoffman, G. Grinstein, K. Marx, I. Grosse, and E. Stanley. DNA visual and analytic data mining. In *Visualization’97., Proceedings*, pages 437–441, 1997. Cited on pages 21 and 22.
- P. E. Hoffman and G. G. Grinstein. A survey of visualizations for high-dimensional data mining. *Information visualization in data mining and knowledge discovery*, 104:4, 2002. Cited on page 21.
- Y. Hu, S. Gao, K. Janowicz, B. Yu, W. Li, and S. Prasad. Extracting and understanding urban areas of interest using geotagged photos. *Computers, Environment and Urban Systems*, 54: 240–254, 2015. Cited on pages 8, 9, 94, 97, and 98.
- T. Jacqui. What makes a great suburb? <http://www.domain.com.au/advice/makes-great-suburb/>, accessed March 2017. Cited on pages 32, 35, and 39.
- R. A. Jarvis. On the identification of the convex hull of a finite set of points in the plane. *Information Processing Letters*, 2(1):18–21, 1973. Cited on pages 75, 98, and 100.
- W. Javed and N. Elmqvist. Exploring the design space of composite visualization. In *IEEE PacificVis*, 2012. Cited on page 26.
- D. F. Jerding and J. T. Stasko. The information mural: a technique for displaying and navigating large information spaces. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):257–271, 1998. Cited on pages 27, 66, and 70.

- B. Jiang and Z. Li. Geovisualization: design, enhanced visual tools and applications. *The Cartographic Journal*, 42(1):3–4, 2005. Cited on page 23.
- U. Jugel, Z. Jerzak, G. Hackenbroich, and V. Markl. M4: a visualization-oriented time series data aggregation. In *Proceedings of the VLDB Endowment*, pages 797–808, 2014. Cited on pages 2 and 28.
- U. Jugel, Z. Jerzak, G. Hackenbroich, and V. Markl. VDDA: automatic visualization-driven data aggregation in relational databases. *The VLDB Journal*, 25(1):53–77, 2016. Cited on page 28.
- D. Jung, H. Park, R. Maeng, and S. Han. A geometric pattern-based method to build hierarchies of geo-referenced tags. *IEEE International Conference on Privacy, Security, Risk and Trust*, pages 546–551, 2010. Cited on page 98.
- D. A. Keim. Pixel-oriented visualization techniques for exploring very large data bases. *Journal of Computational and Graphical Statistics*, 5(1):58–77, 1996. Cited on pages 27, 28, and 66.
- D. A. Keim. Designing pixel-oriented visualization techniques: theory and applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):59–78, 2000. Cited on pages 21 and 22.
- D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, 2002. Cited on page 67.
- I. Kolingerová and B. Žalik. Reconstructing domain boundaries within a given set of points, using Delaunay triangulation. *Computers and Geosciences*, 32(9):1310–1319, 2006. Cited on pages 101 and 102.
- R. Kosara, F. Bendix, and H. Hauser. Parallel sets: interactive exploration and visual analysis of categorical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):558–568, 2006. Cited on page 51.
- H. Lam, T. Munzner, and R. Kincaid. Overview use in multiple visual information resolution interfaces. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1278–1285, 2007. Cited on page 2.
- J. A. Leonard, M. A. Kramer, and L. H. Ungar. Using radial basis functions to approximate a function and its error bounds. *IEEE Transactions on Neural Networks*, 3(4):624–627, 1992. Cited on pages 98 and 102.

- W. Li, T. Chen, E. A. Wentz, and C. Fan. Nmmi: a mass compactness measure for spatial pattern analysis of areal features. *Annals of the Association of American Geographers*, 104(6):1116–1133, 2014. Cited on page 98.
- L. Lins, J. Klosowski, and C. Scheidegger. Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2456–2465, 2013. Cited on pages 3, 7, 8, 28, 64, 66, 67, 69, 71, 72, and 87.
- Z. Liu, B. Jiang, and J. Heer. imMens: real-time visual querying of big data. *Computer Graphics Forum*, 32(3):421–430, 2013. Cited on pages 3, 7, 8, 28, 64, 67, 71, 72, and 87.
- Y. Lu, R. Garcia, B. Hansen, M. Gleicher, and R. Maciejewski. The state-of-the-art in predictive visual analytics. 36(3):539–562, 2017. Cited on page 28.
- Y. Luo, X. Qin, N. Tang, G. Li, and X. Wang. DeepEye: creating good data visualizations by keyword search. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1733–1736, 2018. Cited on page 143.
- A. M. MacEachren and D. R. F. Taylor. *Visualization in modern cartography*. Elsevier, 2013. Cited on page 23.
- J. R. Mackay. Dotting the dot map: an analysis of dot size, number, and visual tone density. *Surveying and Mapping*, 9(1):3–10, 1949. Cited on page 50.
- R. Mafrur, M. A. Sharaf, and H. A. Khan. Dive: diversifying view recommendation for visual data exploration. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1123–1132, 2018. Cited on page 143.
- E. Maguire, P. Rocca-Serra, S.-A. Sansone, J. Davies, and M. Chen. Taxonomy-based glyph design—with a case study on visualizing workflows of biological experiments. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2603–2612, 2012. Cited on page 20.
- Y. Maillot, B. Adam, and M. Melkemi. Shape reconstruction from unorganized set of points image analysis and recognition. In *Image Analysis and Recognition*, pages 274–283, 2010. Cited on pages 101 and 102.
- G. McKenzie, K. Janowicz, S. Gao, J.-A. Yang, and Y. Hu. Poi pulse: a multi-granular, semantic signature-based information observatory for the interactive visualization of big geosocial data. *Cartographica*, 50(2):71–85, 2015. Cited on page 98.

- M. Melkemi. A-shapes of a finite point set. In *Symposium on Computational Geometry*, pages 367–369, 1997. Cited on pages 100 and 102.
- M. Melkemi. Computing the shape of a planar points set. *Pattern Recognition*, 33(9):1423–1436, 2000. Cited on pages 100 and 102.
- S. Methirumangalath, A. D. Parakkat, and R. Muthuganapathy. A unified approach towards reconstruction of a planar point set. *Computers and Graphics*, 51:90–97, 2015. Cited on page 102.
- F. Miranda, L. Lins, J. Klosowski, and C. 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. Cited on page 67.
- A. Moreira and M. Y. Santos. Concave hull: a k-nearest neighbours approach for the computation of the region occupied by a set of points. In *International Conference on Computer Graphics Theory and Applications*, pages 61–68, 2007. Cited on pages 65, 70, 74, 75, 95, 100, 101, 102, and 125.
- T. Munzner. *Visualization analysis and design*. CRC Press, 2014. Cited on pages 3, 5, 10, 11, 19, 20, 21, 26, 31, 32, 40, 44, and 139.
- T. Murata and H. Ishibuchi. MOGA: multi-objective genetic algorithms. In *IEEE International Conference on Evolutionary Computation*, pages 289–294, 1995. Cited on page 71.
- F. Murtagh. A survey of recent advances in hierarchical clustering algorithms. *The Computer Journal*, 26(4):354–359, 1983. Cited on page 81.
- H. R. Nagel. Scientific visualization versus information visualization. In *Workshop on State-Of-The-Art in Scientific and Parallel Computing*, 2006. Cited on page 18.
- S. Nutanong, E. H. Jacox, and H. Samet. An incremental Hausdorff distance calculation algorithm. *PVLDB*, 4(8):506–517, August 2011. Cited on pages 8 and 64.
- E. Packer, P. Bak, M. Nikkilä, V. Polishchuk, and H. J. Ship. Visual analytics for spatial clustering: using a heuristic approach for guided exploration. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2179–88, 2013. Cited on page 98.

- 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. Cited on pages 7, 8, 28, 64, 67, 69, 71, 72, 87, 89, and 91.
- A. Parameswaran. Visual data exploration: a fertile ground for data management research. <http://wp.sigmod.org/?p=2342>, 2018. Cited on pages 3, 20, 31, 32, and 139.
- J.-S. Park and S.-J. Oh. A new concave hull algorithm and concaveness measure for n-dimensional datasets. *Journal of Information Science and Engineering*, 28:587–600, 2012a. Cited on pages 9, 95, 99, 102, 118, and 125.
- J.-S. Park and S.-J. Oh. A new concave hull algorithm and concaveness measure for n-dimensional datasets. *Journal of Information Science and Engineering*, 28:587–600, 2012b. Cited on pages 74 and 75.
- R. E. Patterson, L. M. Blaha, G. G. Grinstein, K. K. Liggett, D. E. Kaveney, K. C. Sheldon, P. R. Havig, and J. A. Moore. A human cognition framework for information visualization. *Computers & Graphics*, 42:42–58, 2014. Cited on page 17.
- J. Peethambaran and R. Muthuganapathy. A non-parametric approach to shape reconstruction from planar point sets through Delaunay filtering. *CAD Computer Aided Design*, 62:164–175, 2015. Cited on pages 101, 102, and 118.
- W. Peng, M. O. Ward, and E. A. Rundensteiner. Clutter reduction in multi-dimensional data visualization using dimension reordering. In *InfoVis*, pages 89–96, 2004. Cited on pages 27 and 66.
- C. Pettit. Using an Online Spatial Analytics Workbench for Understanding Changing Housing Markets across Australian Cities. In *Proceedings of the ACM First International Workshop on Understanding the City with Urban Informatics*, page 1. ACM, 2015. Cited on page 23.
- M. Pohl and D. Feldmann. Generating straight outlines of 2d point sets and holes using dominant directions or orthogonal projections. In *Conference on Computer Vision*, pages 59–71, 2016. Cited on pages 101 and 102.
- E. Polisciuc, A. Alves, and P. Machado. Understanding urban land use through the visualization of points of interest. In *Proceedings of the Fourth Workshop on Vision and Language*, 2015. Cited on pages 94, 99, and 102.

- F. P. Preparata and S. J. Hong. Convex hulls of finite sets of points in two and three dimensions. *Communications of the ACM*, 20(2):87–93, 1977. Cited on page 99.
- F. P. Preparata and M. I. Shamos. *Computational Geometry: an Introduction*. Springer Science & Business Media, 2012. Cited on page 112.
- R. S. Pressman. *Software engineering: a practitioner's approach*. Palgrave Macmillan, 2005. Cited on page 32.
- J. D. Radke. On the shape of a set of points. *IEEE Transactions on Information Theory*, 29(4):551–559, 1983. Cited on pages 101, 102, and 125.
- M. Raubal and S. Winter. Enriching wayfinding instructions with local landmarks. In *International Conference on Geographic Information Science*, pages 243–259, 2002. Cited on page 97.
- G. J. Rawlins and D. Wood. Optimal computation of finitely oriented convex hulls. *Information and Computation*, 72(2):150–166, 1987. Cited on page 102.
- E. Rosén, E. Jansson, and M. Brundin. Implementation of a fast and efficient concave hull algorithm. Technical report, Uppsala University, 2014. Cited on pages 99, 100, 102, and 125.
- M. Rubio-Sanchez, L. Raya, F. Diaz, and A. Sanchez. A comparative study between RadViz and Star Coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):619–628, 2016. Cited on pages 21 and 22.
- A. Saalima and D. A. Shajahan. Shape reconstruction by analysing all the inner holes. In *International Conference on Emerging Technological Trends*, pages 1–6, 2016. Cited on page 100.
- D. Sacha, A. Stoffel, F. Stoffel, B. C. Kwon, G. Ellis, and D. A. Keim. Knowledge generation model for visual analytics. *IEEE transactions on visualization and computer graphics*, 20(12):1604–1613, 2014. Cited on page 19.
- H. Samet. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006. Cited on pages 65, 96, 100, 106, 135, 140, and 141.
- H. Samet and M. Tamminen. An improved approach to connected component labeling of images. In *International Conference on Computer Vision and Pattern Recognition*, pages 312–318, Miami Beach, FL, June 1986. Cited on page 81.

- H. Samet and R. E. Webber. On encoding boundaries with quadtrees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(3):365–369, May 1984. Cited on page 81.
- R. Seidel. The upper bound theorem for polytopes: an easy proof of its asymptotic version. *Computational Geometry*, 5(2):115–116, 1995. Cited on pages 78 and 112.
- J. Seo and B. Shneiderman. Knowledge discovery in high-dimensional data: Case studies and a user survey for the rank-by-feature framework. *IEEE transactions on visualization and computer graphics*, 12(3):311–322, 2006. Cited on page 143.
- M. Shahbazi, J. R. Barr, V. Hristidis, and N. N. Srinivasan. Estimation of the investability of real estate properties through text analysis. In *the 10th International Conference on Semantic Computing*, pages 301–306, 2016. Cited on page 34.
- B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *IEEE Symposium on Visual Languages*, pages 336–343, 1996. Cited on pages 18 and 68.
- H. Siirtola and K.-J. Räihä. Interacting with parallel coordinates. *Interacting with Computers*, 18(6):1278–1309, 2006. Cited on pages 21 and 22.
- A. D. Singleton and P. A. Longley. Geodemographics, visualisation, and social networks in applied geography. *Applied Geography*, 29(3):289–298, 2009. Cited on page 26.
- R. Spence. Representation. In *Information visualization: an introduction*, pages 41–110. 2014. Cited on page 35.
- C. D. Stolper, A. Perer, and D. Gotz. Progressive visual analytics: user-driven visual exploration of in-progress analytics. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1653–1662, 2014. Cited on page 27.
- G. Sun, R. Liang, F. Wu, and H. Qu. A web-based visual analytics system for real estate data. *Science China Information Sciences*, 56(5):1–13, 2013. Cited on page 35.
- F. Tan, C. Cheng, and Z. Wei. Modeling real estate for school district identification. In *IEEE International Conference on Data Mining*, pages 1227–1232, 2017. Cited on page 34.
- D. R. F. Taylor. Perspectives on visualization and modern cartography. In *Visualization in modern cartography*, pages 333–341. 1994. Cited on page 23.

- the National Association of Realtors. Home buyer and seller generational trends report 2017 & 2016, 2017. Cited on page 1.
- P. Thornhill. Real estate buying guide - Melbourne: a first home buyer's guide. Cited on pages 32, 35, and 39.
- W. R. Tobler. A computer movie simulating urban growth in the detroit region. *Economic geography*, 46(sup1):234–240, 1970. Cited on pages 8 and 64.
- Trulia. Real estate regrets: Recovery edition, 2017. URL <https://www.trulia.com/blog/trends/regrets-2017/>. Cited on page 1.
- M. Trutschl, G. Grinstein, and U. Cvek. Intelligently resolving point occlusion. In *InfoVis*, pages 131–136, 2003. Cited on pages 27 and 66.
- E. R. Tufte. *The visual display of quantitative information*, volume 2. Graphics press Cheshire, CT, 2001. Cited on page 17.
- C. Turkay, A. Slingsby, H. Hauser, J. Wood, and J. Dykes. Attribute signatures: dynamic visual summaries for analyzing multivariate geographical data. *IEEE Trans. Vis. Comput. Graph.*, 20(12):2033–2042, 2014. Cited on pages 2 and 26.
- C. Turkay, E. Kaya, S. Balcisoy, and H. Hauser. Designing progressive and interactive analytics processes for high-dimensional data analysis. *IEEE Transactions on visualization and computer graphics*, 23(1):131–140, 2017a. Cited on page 53.
- C. Turkay, E. Kaya, S. Balcisoy, and H. Hauser. Designing progressive and interactive analytics processes for high-dimensional data analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):131–140, 2017b. Cited on pages 7 and 27.
- L. Tweedie, B. Spence, D. Williams, and R. Bhogal. The Attribute Explorer. In *Conference on Human Factors in Computing Systems (CHI)*, pages 435–436, 1994. Cited on page 35.
- J. Vertesi. Mind the gap the London underground map and users' representations of urban space. *Social Studies of Science*, 38(1):7–33, 2008. Cited on page 23.
- C. Viau, M. J. McGuffin, Y. Chiricota, and I. Jurisica. The flowvizmenu and parallel scatterplot matrix: hybrid multidimensional visualizations for network exploration. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1100–1108, 2010. Cited on page 23.

- S. Wang, S. Ali, T. Yue, and M. Liaaen. Integrating weight assignment strategies with NSGA-II for supporting user preference multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 22(3):378–393, 2017a. Cited on page 71.
- Z. Wang, N. Ferreira, Y. Wei, A. S. Bhaskar, and C. Scheidegger. Gaussian cubes: real-time modeling for visual exploration of large multidimensional datasets. *IEEE Transactions on Visualization and Computer Graphics*, 23:681–690, 2017b. Cited on pages 7 and 67.
- M. O. Ward, G. Grinstein, and D. Keim. *Interactive data visualization: foundations, techniques, and applications*. CRC Press, 2010. Cited on pages 21, 22, and 23.
- M. O. Ward, G. Grinstein, and D. Keim. *Interactive data visualization: foundations, techniques, and applications*. AK Peters/CRC Press, 2015. Cited on pages 5, 17, and 18.
- C. Ware. *Information visualization: perception for design*. Elsevier, 2012. Cited on pages 21 and 22.
- D. B. West. *Introduction to Graph Theory*. Prentice Hall Upper Saddle River, 2001. Cited on page 110.
- L. Wilkinson and M. Friendly. The history of the cluster heat map. *The American Statistician*, 2012. Cited on pages 21, 22, 48, and 50.
- C. Williamson and B. Shneiderman. The dynamic HomeFinder: Evaluating dynamic queries in a real-estate information exploration system. In *ACM SIGIR*, pages 338–346, 1992. Cited on page 34.
- K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: exploratory analysis via faceted browsing of visualization recommendations. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):649–658, 2016. Cited on page 143.
- C. Xia, W. Hsu, M. L. Lee, and B. C. Ooi. BORDER: Efficient computation of boundary points. *IEEE Transactions on Knowledge and Data Engineering*, 18(3):289–303, 2006. Cited on pages 100 and 102.
- H. Xing, Y. Meng, D. Hou, J. Song, and H. Xu. Employing crowdsourced geographic information to classify land cover with spatial clustering and topic model. *Remote Sensing*, 9(602):1–20, 2017. Cited on page 98.

- J. Xu, Y. Feng, Z. Zheng, and X. Qing. A concave hull algorithm for scattered data and its applications. In *International Congress on Image and Signal Processing*, volume 5, pages 2430–2433, 2010. Cited on pages 101 and 102.
- X. Yuan, P. Guo, H. Xiao, H. Zhou, and H. Qu. Scattering points in parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1001–1008, 2009. Cited on page 23.
- X. Zhong and M. Duckham. Characterizing the shapes of noisy, non-uniform, and disconnected point clusters in the plane. *Computers, Environment and Urban Systems*, 57:48–58, 2016. Cited on pages 102, 103, 106, 125, and 130.
- X. Zhong and M. Duckham. An efficient incremental algorithm for generating the characteristic shape of a dynamic set of points in the plane. *Geographical Information Science*, 31(3):569–590, 2017. Cited on pages 3, 9, 95, and 102.
- H. Zhu, H. Xiong, F. Tang, Q. Liu, Y. Ge, E. Chen, and Y. Fu. Days on market: Measuring liquidity in real estate markets. In *the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 393–402, 2016. Cited on page 34.
- J. Zhu, Y. Sun, and Y. Pang. A density based algorithm to detect cavities and holes from planar points. *Computers and Geosciences*, 109(1):178–193, 2017. Cited on page 102.