DESIGN AND IMPLEMENTATION OF A CO-LOCATION ANALYSIS TOOL

A Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Xiaoxi Man

December 2014

DESIGN AND IMPLEMENTATION OF A CO-LOCATION ANALYSIS TOOL

Xiaoxi Man

APPROVED:

Dr. Christoph Eick
Dept. of Computer Science

Dr. Ricardo Vilalta
Dept. of Computer Science

Dr. Klaus Kaiser
Dept. of Mathematics

Dean, College of Natural Sciences and Mathematics

Acknowledgments

I would like to thank my advisor, Dr. Eick, for his invaluable advice and guidance during my study. I really appreciate his contribution of time and discussion for guiding me how to do research.

Also, I would like to take this opportunity to thank all my committee members: Dr. Vilalta and Dr. Kaiser for their time, insightful comments, and help.

Last, but not least, my dissertation is dedicated to my husband, Wei Ding, and my daughter, Jennifer Yuqi Ding. Also, I cannot express my appreciation for endless supports from my parents, Hongliu Man and Youping Li. Their love and support provided me the driving force, courage, and perseverance. Thank you all for supporting me along the way.

DESIGN AND IMPLEMENTATION OF A CO-LOCATION ANALYSIS TOOL

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Xiaoxi Man

December 2014

Abstract

In recent years the widespread usage of scanning device, such as GPS-enabled devices, PDAs, and video cameras, has resulted in an abundance of spatial data. Therefore, there is an increasing interest in mining hidden patterns in spatial data. Discovery of co-location patterns has been a research area in association analysis for several years.

In this thesis, we designed and implemented a user-friendly, interactive *Co-location Analysis Tool* which can be used to extract co-location patterns from spatial datasets. By using this tool, we are able to extract co-location patterns at different levels of granularity; these results can help with business decision-making, ecology research, and urban planning. The tool provides two approaches to analyze collocation patterns: Ripley's K-function approach, and a novel approach called K-Nearest-Neighbor distance approach. Both approaches compute spatial statistics for different neighborhood sizes and compare these characteristics with spatial characteristics obtained by placing objects randomly to determine the presence of collocation and anti-collocation. The second approach uses summaries of K-nearest neighbor distances of objects in the dataset to diagnose the presence of collocation patterns. In addition, the tool provides visualization techniques to present the data analysis experimental results. Finally, we validated the tool and compared the two collocation analysis approaches for a building dataset.

Contents

1 Introduction **1**

1.1 Overview 1

1.2 Research contributions 3

1.3 Thesis organization 4

2 Related work **5**

2.1 Spatial data mining 7

2.1.1 Co-location patterns mining 8

2.1.2 Tools for spatial data mining 9

2.1.2.1 ArcGIS 10

2.1.2.2 R package 11

2.1.2.3 Google Earth 12

3 Algorithms **13**

3.1 Polygon wrapping algorithm 13

3.1.1 Andrew's monotone chain algorithm 14

3.1.2 Basic concepts of Andrew's monotone chain algorithm 14

3.1.3 Andrew's monotone chain algorithm 15

3.1.4 Algorithm for judging if a point is inside a polygon 17

3.2 Co-location data mining algorithm 18

3.2.1 K-Nearest-Neighbor distance approach 18
3.2.2 The Ripley's K-function approach 19

4 Co-location Analysis Tool Design 23

4.1 Architecture of the co-location analysis tool 23
4.2 GUI-design-related classes 26
4.3 Co-location analysis work flow 29
4.4 KML file parsing 30
4.5 Spatial data analysis and visualization 32
4.6 Evaluation and verification 32

5 Experiment Results 43

5.1 Summary of experiment dataset 43
5.2 Experiment result 45
5.2.1 Case one 47
5.2.1.1 Using Ripley's K-function approach for case one . . . 48
5.2.1.2 Using KNN distance approach for case one 49
5.2.1.3 Comparing the two approaches in case one 52
5.2.2 Case two 53
5.2.2.1 Using Ripley's K-function approach for case two . . . 54
5.2.2.2 Using KNN distance approach for case two 55
5.2.2.3 Comparing the two approaches in case two 56
5.2.3 Case three 57
5.2.3.1 Using Ripley's K-function approach for case three . . 58
5.2.3.2 Using KNN distance approach for case three 59
5.2.3.3 Comparing the two approaches in case three 61
5.2.4 Case four 62

5.2.4.1 Using Ripley's K-function approach for case four . . 62
5.2.4.2 Using KNN distance approach on case four 64
5.2.4.3 Comparing the two approaches in case four 65
5.2.5 Case five 67
5.2.5.1 Using Ripley's K-function approach for case five . . . 68
5.2.5.2 Using KNN distance approach for case five 69
5.2.5.3 Comparing the two approaches in case five 70
5.2.6 Summary 71

6 Conclusion and Future Work **73**
6.1 Conclusion 73
6.2 Future Work 74

Bibliography **76**

List of Figures

3.1 Example of Andrew's monotone chain algorithm 16
3.2 KNN distance algorithm example. 18
3.3 KNN distance algorithm work flow. 20
3.4 K-function diagram. 21

4.1 Functional modules of the co-location analysis tool. 24
4.2 Overview of the co-location analysis tool. 25
4.3 Co-location analysis tool - main classes 27
4.4 Welcome window of the co-location analysis tool 28
4.5 Qwt classes used to store points coordinate information. 32
4.6 Qwt classes used for points and curves drawing. 33
4.7 Visualization of verification points. 34
4.8 Visualization of spatial data from KML files. 35
4.9 Visualization of spatial data from KML files. 36
4.10 Visualization of polygon based on the input spatial data. 37
4.11 Visualization of randomly generated spatial points within polygon. . . 38
4.12 Visualization of randomly generated spatial points together with the original spatial points within polygon. 39
4.13 Verification of co-location analysis by using the K-function algorithm. 41
4.14 Verification of co-location analysis by using the KNN algorithm. . . . 42

5.1 A neighborhood of city of Strasbourg visualized by Google Earth . . 44
5.2 Distribution of garages and single houses in original input dataset . . 47
5.3 Random distribution of garages and single houses 48
5.4 Ripley's K-function case 1: garages vs single houses 49
5.5 KNN distance approach on case 1: garages vs single houses 50
5.6 Zoomming in result of KNN distance approach on case 1 51
5.7 Distribution of garages and collective houses in original input dataset 53
5.8 Random distribution of garages and collective houses 54
5.9 Ripley's K-function case 2: garages vs collective houses 55
5.10 KNN distance approach case 2: garages vs collective houses 56
5.11 Distribution of commercial buildings and light buildings in original input dataset 58
5.12 Random distribution of commercial buildings and light buildings . . . 59
5.13 Ripley's K-function case 3: commercial buildings vs light buildings . . 60
5.14 KNN distance approach case 3: commercial buildings vs light buildings 61
5.15 Distribution of commercial buildings and school in original input dataset 63
5.16 Random distribution of commercial buildings and schools 64
5.17 Ripley's K-function case 4: commercial buildings vs schools 65
5.18 KNN distance approach case 4: commercial buildings vs schools . . . 66
5.19 Distribution of garages and collective houses in original input dataset 67
5.20 Random distribution of garages and collective houses 68
5.21 Ripley's K-function case 5: garges vs collective houses 69
5.22 KNN distance approach case 5: collective houses vs garages 70

List of Tables

2.1 A example of sequential dataset of web log sessions. 6

3.1 Algorithms used for convex hull construction. 14

4.1 Summarization of class functions. 40

5.1 Building numbers of different building types 45
5.2 Statistical data of KNN distance approach for case one 50
5.3 Average execution time of both approaches 52
5.4 Statistical data of KNN distance approach for case two 57
5.5 Average execution time of both approach of case two 57
5.6 Statistical data of KNN distance approach for case three 62
5.7 Average execution time of both approach of case three 62
5.8 Statistical data of the KNN distance approach for case four 64
5.9 Average execution time of both approach of case four 66
5.10 Statistical data of KNN distance approach for case five 71
5.11 Average execution time of both approach of case five 71

Chapter 1

Introduction

1.1 Overview

Data mining [18], also called "Knowledge Discovery in Databases" (KDD), is a subfield of computer science that aims to extract interesting and useful information from large data [4]. In recent years, the widespread availability of spatial data [20, 40, 49] has led to an awareness of the importance of mining spatial data. In order to provide new understanding of certain domains, spatial data mining has become important in the research area of data mining. One important spatial data mining task is to discover interesting but implicit spatial patterns [19, 37, 43] that might exist in spatial data. Discovering spatial patterns from spatial data is much more difficult than discovering patterns in traditional data. Challenges include the huge amounts of spatial data [48], and the complex data types of spatial data such as lines, polygons, and spatial autocorrelation [38].

Spatial data are generated by devices such as satellites, GPS, city cameras, etc. A lot of useful and interesting information are hidden in these tons of data. For example, ecologists might be interested in studying how American alligators and American crocodiles are distributed in a wetland, and whether or not they co-occur frequently in a specific region. Therefore, a task that aims to explore co-located patterns in spatial data has become an important research area. Some institutions and companies have developed spatial analysis softwares or packages to enable people to analyze spatial data. For example, the Esri company developed ArcGIS to help people to visualize and analyze spatial data; it also provides services including overlay, surface, proximity, suitability, network analysis,interpolation analysis, and other geo-statistical modeling techniques [8]. Other popular spatial analysis softwares or packages including Google Earth, PostGIS, ArcGIS, spatial package of R(spatstat package), etc., all of which provide services of mining knowledge of spatial data.

In order to ease the job for people to analyze co-location patterns in spatial data, we create a *Co-location Analysis Tool* in this thesis. The developed co-location analysis tool incorporates two different methods to analyze co-location patterns for spatial data. The first approach is called Ripley's K-function which is a popular approach used to analyze spatial data [35]. The second approach is called K-Nearest-Neighbor distance (KNN distance for short) function which is used for the first time to analyze the co-locations. In order to make sure it is easy to use, we also designed a user-friendly graphical user interface for the tool by using QT, which merged the spacial data analysis algorithms and experiment result visualization function under one framework.

1.2 Research contributions

There are some spacial data mining analysis applications such as mathlab, R, etc., but they only provide very generic APIs or functions which can be used for the spatial data analysis. Users are required to be familiar with Mathlab and R programming languages in order to develop code to do spatial data co-location analysis. In order to do the spatial data co-location analysis, users have to learn to parse and get the spatial data and write program to do analysis. This is a time-consuming and error-prone process. After getting the experiment results, people either have to write codes to show the results or use some other software.

The main contribution of this work is that we have designed and created a spatial data co-location analysis tool which can greatly ease the work of analysis. It can automatically handle spatial data file reading (*.kml from Google Earth) and to do co-location analysis based on different methods users choose. This tool also integrates a visualization module which can present the data analysis experimental results transparently. Another advantage of this tool is that we built our tool based on the Qt implementation framework, which can easily deploy this tool under both Linux and Windows environment. Additionally, we also provided a module which allows the tool to dump all the intermediate data analysis results, which can be used by other software like Excel, etc.. Moreover, we introduce a new spatial statistic approach called KNN distance approach to analyze spatial patterns.

1.3 Thesis organization

The rest of the thesis is organized as follows: Chapter 2 first provides an overview of association analysis and then introduces the related work of spatial data mining. Specifically, it describes what is co-location pattern mining and introduces motivation for designing a tool for extracting co-location patterns from spatial data. It also describes several popular softwares and tools for spatial data analysis. Chapter 3 describes several algorithms used for the development of the tool, which including Ripley's K-function and KNN distance approach, and Andrews monotone chain algorithm for determining convex hull for a set of points. Chapter 4 introduces the framework and architecture of the tool. Additionally, details of its implementation and design are described as well. Chapter 5 shows the experimental results of several cases and gives proper interpretation and analysis for each case. Chapter 6 provides a conclusion and describes future work.

Chapter 2

Related work

A frequent pattern usually refers to a pattern (sub-graph, sub-sequences, sub-structures, etc.) which co-occurs frequently in a dataset. Problems such as what products are frequently purchased together or what web pages are consecutively clicked are quite common in our daily life.

Association analysis, which seeks to find frequent patterns that describe the relationships among attributes (variables) of a data set, has been a research area of data mining for a while. Companies have created a unique set of association analysis tools that are widely employed in business and science [42]. Classical association analysis focuses on analyzing transaction data which can help retailers to develop marketing strategies [33] by discovering the frequent items that are always purchased together by customers. For example, Christmas trees and Christmas tree decorations are frequently bought together.

Sequence pattern mining [12] is another important association analysis sub-area with widespread applications [29]. It serves an important role for professionals working in bio-informatics, genomics, web services, and financial data analysis. [17]. For example, it can help discover customer shopping behavior in retail stores by extracting useful information from sequence databases. A sequential database is a collection of sequences of ordered elements or events [21, 50]. Sequential pattern mining is somewhat similar to frequent item-set mining, but with mining sequential pattern we need to take the event order into consideration. Table 2.1 shows an example of a sequence dataset. Here we use web sessions which actually are a consecutive series of web page clicks as an example. An item such as BCD in this table means that a user first visits page B, then page C, and then page D.

Table 2.1: A example of sequential dataset of web log sessions.

Session ID	**Session sequence**
1	ABCE
2	ABEF
3	BCD
4	BCD
5	CD
6	ACF

mining is to discover subgraphs that frequently co-occur in a given set of graphs. A subgraph is regarded as frequent if the frequency is above a given support threshold.

There are several algorithms used to mine graphs [34]. Inokuchi et al. [25] proposed an apriori-based algorithm which can discover both connected and disconnected frequent subgraphs. Another algorithm called FSG was then developed by

Kuramochi and Karypis [27]; it is based on the idea of an edge-growing strategy. Xifeng Yan and Jiawei han proposed an algorithm called gSpan which is based on depth-first search (DFS) in frequent subgraph mining [51]. Other algorithms for graph mining, including simple path patterns [13], generalized path patterns [32], simple tree-like topology patterns [30], tree-like patterns [47], general graph patterns [27], etc., are also used in data mining reserch area. In recent years more and more researchers have focused on how to deal with large-scale graphs that contain billions of entities [28]. In order to improve computational efficiency for graph mining, cloud computing is frequently used to solve the problem.

2.1 Spatial data mining

Spatial data mining [37, 39] is a sub-area of data mining which aims to discover interesting patterns from large spatial data. The research area of spatial data mining serves as an important role, as it helps people discover hidden information from spatial data and reduce the unclear of possible scientific hypotheses. With the widespread usage of spatial data [41] and new data resources such as data from satellite, mobile phone data, and Google Earth data, etc., the topic of spatial data mining has become very important [22].

In the last decade, G.Kiran Kumar and P.Premchand defined "Spatial data mining, or knowledge discovery in spatial database, which refers to the extraction of implicit knowledge, spatial relations, or other patterns not explicitly stored in spatial databases." [26]. The objects of spatial patterns not only refer to points, but

also frequently refer to lines, polygons, and other complex geometrical structures.

Mining spatial data is more difficult than the classical databases because spatial data have some unique features. In general, spatial data have complex data types (data could refer to points, lines, and polygons, etc.), special spatial relationships, and spatial autocorrelation. Because of the unique features that spatial data have, it limits the usage of classical techniques for spatial data mining.

2.1.1 Co-location patterns mining

Spatial co-location patterns mainly refer to subsets of spatial features whose instances are frequently located together in a local regional spatial proximity [23] [14]. For example, if `Walmart` and `Target` supermarkets are located close to each other in a spatial data set, they might be considered co-located; another example is that the location of alcohol-related traffic accidents and the location of bars are frequently located in close proximity. Mining spatial co-location patterns is quite different from mining classical association rules. Classical association rule mining aims to discover interesting relationships in large data [46], which takes traditional transaction-based approach by using minimum threshold such as support, confidence to discover association rules, etc.. However, there is no transaction when mining spatial co-location patterns. Consequently, it is hard to use classical approaches to generate association rules to mine spatial data.

A lot of prior work have been focused on co-location data mining. Morimoto et al. propose an apriori-like algorithm to mining co-location pattern, which is used

to discover frequent neighboring co-locations in spatial data [31]. Their method is based on a plane partitioning approach by using a Voronoi diagram. After that, Shekhar and Huang et al. introduced an apriori-like generate and test approach to discover subsets of spatial features that in close proximately of each other [41]. This method uses spatial join operation, which is very expensive to discover co-location patterns; especially when the data set is very large, it will take a long time to analyze the pattern. In order to speed up pattern analysis, Yoo et al. developed a joinless algorithm which materializes spatial data [52]. In their method, join operations are substituted by scanning operations and look-up operations, so the performance of this method largely depends on the efficiency of the scanning operations and look-up operations. Some other clustering analysis techniques are also explored [24] to discover co-locations; however, the resulting patterns varied when different clustering methods are used. In most of pattern analysis methods discussed above, an appropriate scale of analysis is required. For example, a distance threshold or participation ratio is needed to analyze co-locations. Compared with their work, our tool can summarize spatial patterns (co-located or anti-colocated) over a range of distance without specifying the scale of the analysis.

2.1.2 Tools for spatial data mining

There are several applications that are designed for mining spatial data, each of them has different usages. Based on their goals, we can classify them into the following sections.

2.1.2.1 ArcGIS

ArcGIS is a commercial product of the Esri company, and a geographic information system (GIS) for creating, visualizing, managing, and analyzing spatial data [1]. This tool is very powerful, as it helps users to explore interesting spatial information by analyzing spatial data.

ArcGIS has three platforms [5]: The first platform is GIS professionals [1], which provides services that allow users to get the solutions from the service it supports. This platform includes:

- ArcGIS Online
- ArcGIS for Desktop
- ArcGIS for Mobile
- ArcGIS for Server

The second platform is location analytics [1], which provides services that allowing users to visualize and analyze spatial data. This platform includes:

- Esri Maps for Office
- Esri Maps for IBM Cognos
- Esri Maps for SharePoint
- Esri Maps for Dynamics CRM

- Esri Maps for MicroStrategy
- Esri Maps for SAP BusinessObjects

The third platform is developers [1], which provides services that allowing users to add location to apps quickly. This platform includes:

- ArcGIS for Developers
- Esri Developer Network

Users of ArcGIS can also add software extensions to get solutions for specific problems. These software extensions include ArcGIS Spatial Analyst (advanced raster modeling, VBA for raster analysis, etc.), ArcGIS 3D Analyst (three-dimensional modeling tools, real-time interactive three-dimensional scenes), and Geo-statistical Analyst (exploratory spatial data analysis tools, probability mapping, etc.) [1].

2.1.2.2 R package

Besides the commercial software that we just discussed, R also provides a package that deals with spatial data; it is called *spatstat* and is a package for spatial point patterns analysis. By using this package, users can analyze two-dimensional and three-dimensional spatial datasets. It also provides functions that visualize spatial data, exploratory data analysis, model-fitting, simulation, spatial sampling, model diagnostics, and formal inference [11]. Although the *spatstat* package can provide functions to analyze spatial data, using only the functions of this package cannot

satisfy our research goal, because we need to process the KML file first before we can use this package. Besides, we cannot generate the boundary of a given polygon, and categories cannot be specified either.

2.1.2.3 Google Earth

Google Earth is a virtual globe, map, and geographical information program that was originally called EarthViewer 3D created by Keyhole, Inc, a Central Intelligence Agency (CIA) funded company acquired by Google in 2004 [6]. This tool is able to map the earth by the superimposition of images obtained from satellite imagery, aerial photography, and geographic information systems (GIS) in a vivid 3D global view, which has been widely used in our daily life. Google Earth is composed of a couple of layers, such as: Sky, Ocean, Buildings in 3D, Weather, Places of Interest, etc.. Google Earth supports managing three-dimensional Geo-spatial data through Keyhole Markup Language (*.kml), an extension of *.xml file. Google Earth reads objects' longitude, latitude, etc. information and provides visualization for those objects.

Chapter 3

Algorithms

In this chapter, we are going to introduce several algorithms that were used in the development of our tool. Based on the usage of an algorithm, they are classified into two groups: a) algorithms used for processing spatial data, and b) algorithms for discovering spatial patterns. Specifically, Andrew's monotone chain algorithm is used for wrapping the whole objects of interest data. The Ripley's K-function and K-Nearest-Neighbor distance (KNN distance for short) approaches are used for analyzing the co-location patterns.

3.1 Polygon wrapping algorithm

In this work, in first step, co-location analysis approaches are used to analyze the input spatial dataset. Andrew's monotone chain algorithm is then used to search for boundary points which are used to wrap points inside a polygon. After determining

the boundary polygon, new points are generated randomly inside the boundary. At last, discovering spatial patterns algorithms is used again to capture the spatial statistic of the randomly generated dataset.

3.1.1 Andrew's monotone chain algorithm

In order to generate a polygon to wrap a set of objects in a spatial dataset, we adopt Andrew's monotone chain algorithm to construct a convex hull. Andrew's monotone chain algorithm (Andrew, 1979) is a variation of Graham scan algorithm. It uses a linear lexicographic sort of the point set by x and y-coordinates [3] which works better if the order of a set is already known. Some other algorithms can also be used to construct convex hull, which are listed in Table 3.1.

Table 3.1: Algorithms used for convex hull construction.

Algorithm Name	**Research Year**
Gift wrapping	1973
Graham scan	1972
QuickHull	1978
Divide and conquer	1977
Incremental convex hull	1984
The ultimate planar convex hull	1986
Chan's algorithm	1996

3.1.2 Basic concepts of Andrew's monotone chain algorithm

To understand Andrew's monotone chain algorithm, firstly we need to figure out the convex hull definition. In mathematics, a convex hull is defined as the smallest

convex set of points that contains a given series of points [2]. For example, given a set of points P, the input data is P= $\{p_1, p_2, p_3, ..., p_n\} \subset R^2, n \in N$, then output should be a sequence set $(q_1, q_2, ..., q_n), 1 \leq h \leq n$ (counter-clockwise order). Figure 3.1 below gives an example of the convex hull of a given set of points.

In addition, the concept of chain method also needs to be understood. Suppose we have a set of points D= $\{P_1, P_2, P_3, ..., P_n\}$. A chain is a graph which contains a vertex set $\{P_1, P_2, P_3, ..., P_n\}$ and an edge set $\{(P_1, P_2), (P_2, P_3), ..., (P_n, P_{n+1})\}$.

Moreover, the concept of monotone chain method is an extension of the chain method. If a line orthogonal to a straight line L intersects the chain exactly one time, it is considered to be a monotone chain. To compute the monotone chain, both the upper hull and lower hull of a points set of a monotone chain need to be computed.

3.1.3 Andrew's monotone chain algorithm

To compute a convex hull, the Andrew's monotone chain algorithm has two steps. First, it computes the upper hull and lower hull of a monotone chain of points, then the convex hull is constructed by joining the upper and lower hulls. Details will be illustrated in each step.

- Step 1, compute the upper hull and lower hull. In this step, it first sorts the point set by increasing x and then y coordinate values, then finds the minimum and maximum x-coordinates and the minimum and maximum y-coordinates. Note P_{minmin} as the point that has the minimum x-coordinates

and minimum y-coordinates, P_{minmax} as the point that has the minimum x-coordinates, and maximum y-coordinates, P_{maxmin} as the point that has the maximum x-coordinates and minimum y-coordinates, P_{maxmax} as the point that has the maximum x-coordinates and maximum y-coordinates. Next, create a lower bound L_{lower} by joining two points P_{minmin} and P_{maxmin} together and create an upper bound L_{upper} by joining the two points P_{minmin} and P_{maxmin} together. Then create the lower hull Ω_{lower} by processing the points that are under the L_{lower} and create the upper hull $\Omega_{upper}$by processing the points that are above the L_{upper}.

- Step 2, join the lower hull Ω_{lower} and the upper hull Ω_{upper} together to be a convex hull.

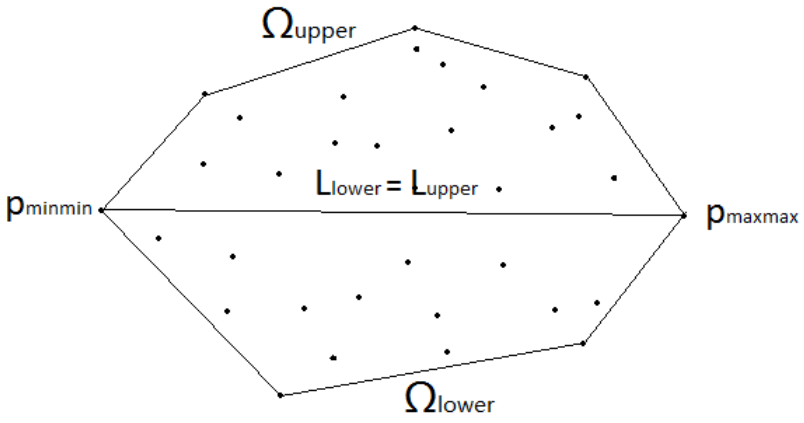

Figure 3.1: Example of Andrew's monotone chain algorithm

For example, as shown in the figure 3.1, we can see that there is a set of points. First we sort the points by increasing the value of the x-coordinate and y-coordinate. Then P_{minmin}, P_{minmax}, P_{maxmin}, and P_{maxmax} are fixed. In this specific example,

$P_{minmin} = P_{minmax}$, and $P_{maxmin} = P_{maxmax}$. After that the lower bound L_{lower} and upper bound L_{upper} are constructed; here the lower bound is the same as upper bound L_{lower}= L_{upper}. Then we generate the lower hull Ω_{lower} and upper hull Ω_{upper}, and join them together to construct a convex hull.

3.1.4 Algorithm for judging if a point is inside a polygon

The algorithm [10] used for evaluating whether a given point is inside a polygon is a simple but useful method. This algorithm has three steps, the detailed procedure of which described below:

1. Check whether the y-coordinate of a test point is within the edges scope. If it is inside then go to step 2; if not, this point is outside the polygon.

2. Check whether a test point is to the left of the line. If that's true, draw a line rightwards from the test point until it crosses that edge.

3. Count the number of times that the rightward line crosses the polygon. If it crosses an odd number of times, then the test point is inside the polygon; if an even number, the test point is outside the polygon. When introducing a new test point, repeat the three steps above.

3.2 Co-location data mining algorithm

3.2.1 K-Nearest-Neighbor distance approach

The K-Nearest-Neighbor distance (or the KNN distance) approach is used in this thesis to create spatial statistics based on the observed K-nearest neighbor distance in a spatial data set. It is used to generate K-nearest neighbor distance curves based on similarity measure (distance function).

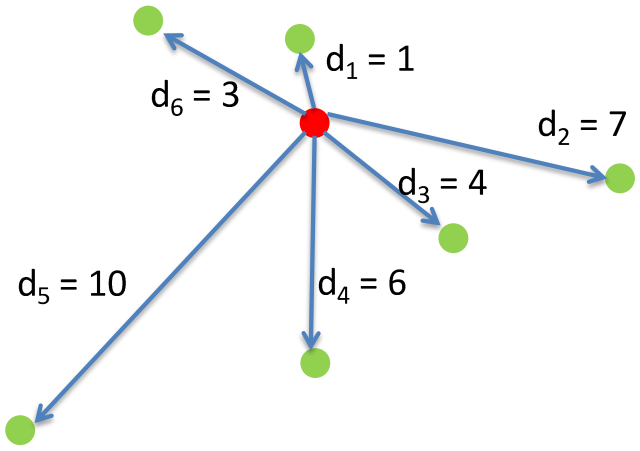

Figure 3.2: KNN distance algorithm example.

To illustrate this approach, an example is given in Figure 3.2. In this figure, we can see there are a couple of green points around a red point. Assume the distances from the red point to the rest of green points are d_1, d_2, d_3, d_4, d_5, and d_6 respectively. If k is set to 1, the 1st nearest neighbor distance is d_1; if k equals to 2, then d_2 is the 2nd nearest neighbor distance. The same way for calculating the kth distance calculation is applied to all the rest green points. Note that there is only one red point in this example for simplicity. If we have multiple red points, we then apply

the above process to each red point and calculate the average distance for each k value. Figure 3.3 shows the work flow of the KNN distance approach.

The pseudocode of KNN distance approach is listed as below:

```
FOR k=k1,   ,kr DO
   FOR all green objects gp DO
      Compute distance rdk to k-nearest red object to g ENDDO
   Compute average rdk of values observed in previous loop
   Put entry (k, rdk) into the Curve
   ENDDO
```

3.2.2 The Ripley's K-function approach

In 1976, Ripley [36] used more points to provide an estimate of spatial dependence over a wider range of scales based on all distances between events in the area of interest.

The K-function defined by Ripley [36] is as follows:

$$K(r) = \frac{1}{\lambda}E$$

where E represents the number of extra events within distance d of a randomly chosen event and λ means the intensity of events. It is usually used in the area of ecology [44] and geographical epidemiology [15, 45].

The formula is as below:

$$K(r) = \frac{R}{n^2} \sum \sum \frac{I_r(d_{ij})}{w_{ij}}$$

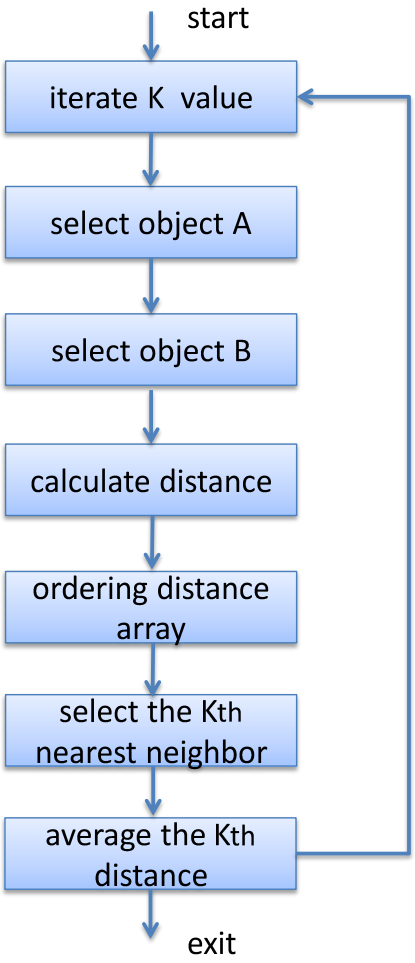

Figure 3.3: KNN distance algorithm work flow.

where n represents the number of point, R means the study area, $I_r(d_{ij})$ is the dummy variable I if $d_{ij} \leq h$, otherwise it counts 0, and w_{ij} represents the proportion of circumference of circle. This formula indicates that the probability of an event at any point in the area of R is independent of any other events that have occurred. Figure 3.4 illustrates the main steps that are used for the co-location analysis.

In general, the Ripley's K-function [16] is a spatial analysis method that is used to describe how point patterns of spatial point data occur over a given study area of

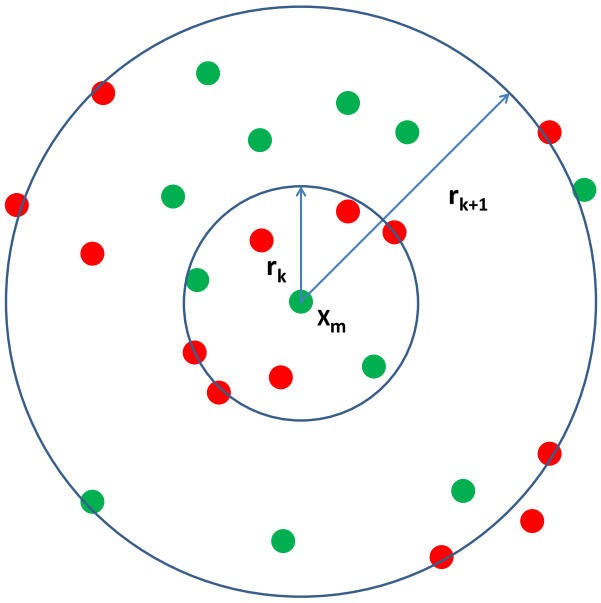

Figure 3.4: K-function diagram.

interest. It can help researchers and scientists to determine whether the phenomenon of interest (for example, different categories of trees, different type of buildings, locations of traffic accident, bars, etc.) appears to be dispersed, clustered, or randomly distributed throughout the region based on distance approach.

Besides, the Ripley's K-function is generally calculated at multiple distances so that point pattern distributions can be changed with scale. For example, when the distances are short, the points could be clustered, and when the distances increase, points might be randomly distributed at the same region.

The Ripley's K-function is used to describe the point patterns of interest. Assume single houses are represented by green dots and collective houses are represented by red dots, the function counts the number of neighboring collective houses

found within a given distance of each individual single house. The average number of observed neighboring collective houses is then compared to the average number of collective houses one would expect to find based on a completely spatially random point pattern. If the average number of collective houses found within a given distance of each individual single house is greater than that for a random distribution, the distribution is considered to be clustered. Otherwise, the distribution is dispersed.

The pseudocode of Ripley's K-function is listed as below:

```
DO radi r1, . . . ,rn:
   DO all green objects g:
      Compute no. of red objects within radius rj of g ENDDO
   Compute average roj of values observed in previous loop
   Put entry (rj, (roj/total_number_of_red_objects)) into Curve
   ENDDO
ENDDO
```

Chapter 4

Co-location Analysis Tool Design

In this chapter, section 4.1 discusses the architecture of the tool. Section 4.2 summarizes the main classes used in the development of co-location analysis tool GUI design and data analysis. Section 4.3 introduces the basic work flow of the tool. Section 4.4 describes the KML file format and the utility that we designed for parsing *.kml files for the input of the co-location analysis. Section 4.5 discusses the techniques that are used for displaying spatial dataset and co-location analysis result visualization. Evaluation and test results are presented in section 4.6.

4.1 Architecture of the co-location analysis tool

Based on functionality, this tool can be divided into 3 different modules. Those 3 modules are QT GUI visualization, co-location pattern analysis, and experimental results visualization & saving respectively, which are presented in figure 4.1. The

GUI visualization module is mainly used to generate the window related controls. The co-location pattern analysis module is used to process co-location pattern analysis by using two different spatial data mining methods. The experimental results visualization & saving module takes care of co-location pattern analysis result visualization and intermediate result file saving. The generated result file can be read by other tools directly.

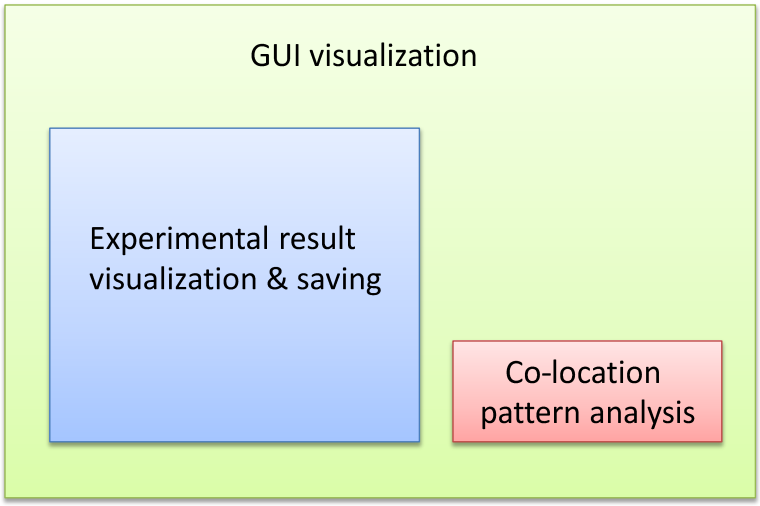

Figure 4.1: Functional modules of the co-location analysis tool.

The general overview of the co-location analysis tool is listed in figure 4.2. The input file is a *.kml file which is a spatial dataset that contains objects information. Our co-location analysis tool will first load the *.kml file and then parse object information inside. An intermediate *.dat file will be generated, which contains object coordinate information. After loading, the input spatial dataset can be visualized in the left-hand board of this tool by clicking the check box “Original points plot”. After that, a method used for analysing spatial patterns needs to be specified and this method will be used for the input spatial dataset for spatial co-location analysis.

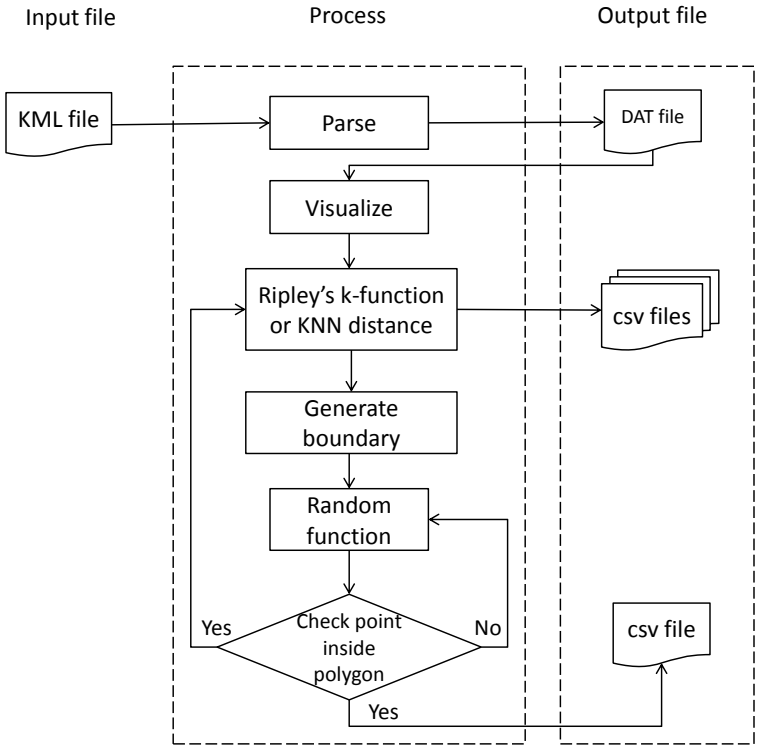

Figure 4.2: Overview of the co-location analysis tool.

In order to randomly place objects in the same area, we need to find the boundary which can include all the objects of the dataset. By using Andrew's monotone chain algorithm, a polygon can be generated to surround all the objects. Next, we randomly generate the same number of objects within the polygon we generated in the previous step. In the meantime, an intermediate *.csv file is generated which stores the new dataset coordinate information. Finally, the previously selected co-location analysis method will be used to analyze the spatial co-location pattern for original objects and the randomly generated objects. The final analysis result will be visualized by this tool. Moreover, the final result will be stored into a *.csv file for reference.

4.2 GUI-design-related classes

The co-location analysis tool has been implemented based on the Qt application development framework. Figure 4.3 summarizes the relationships of main classes used for the tool GUI development and spatial data co-location analysis. Key functions of each main functional class are listed in Table 4.1. The main function is the program entry point, where it firstly initializes class `MainWindow` for the main window generation. In the meantime, class `MainWindow` initializes classes `Plot` and `Panel` which are two classes used for generating left-hand experimental result plotting board and right-hand experimental parameters input text boxes, and down control units respectively. Function descriptions for each class are summarized below:

- MainWindow: Creation of the main window, which creates menu, Menu-bar, status bar, panel, and potting board. It also handles basic operations such as *.kml file loading, saving, printing and data analysis, etc..
- Co-Location: Handles *.kml parsing, reading, random spatial data points generation within polygon and spatial data analysis and computation
- Panel: Implement spatial data analysis visualization, polygon visualization, experimental results curve drawing, diagram zooming in and out, mouse drag, etc..
- QwtPlot: Basic implementation which handles curve style setting, polygon symbol settings, data sampling for visualization.
- QwtPlotCurve: Underneath implementation for drawing curves and points.

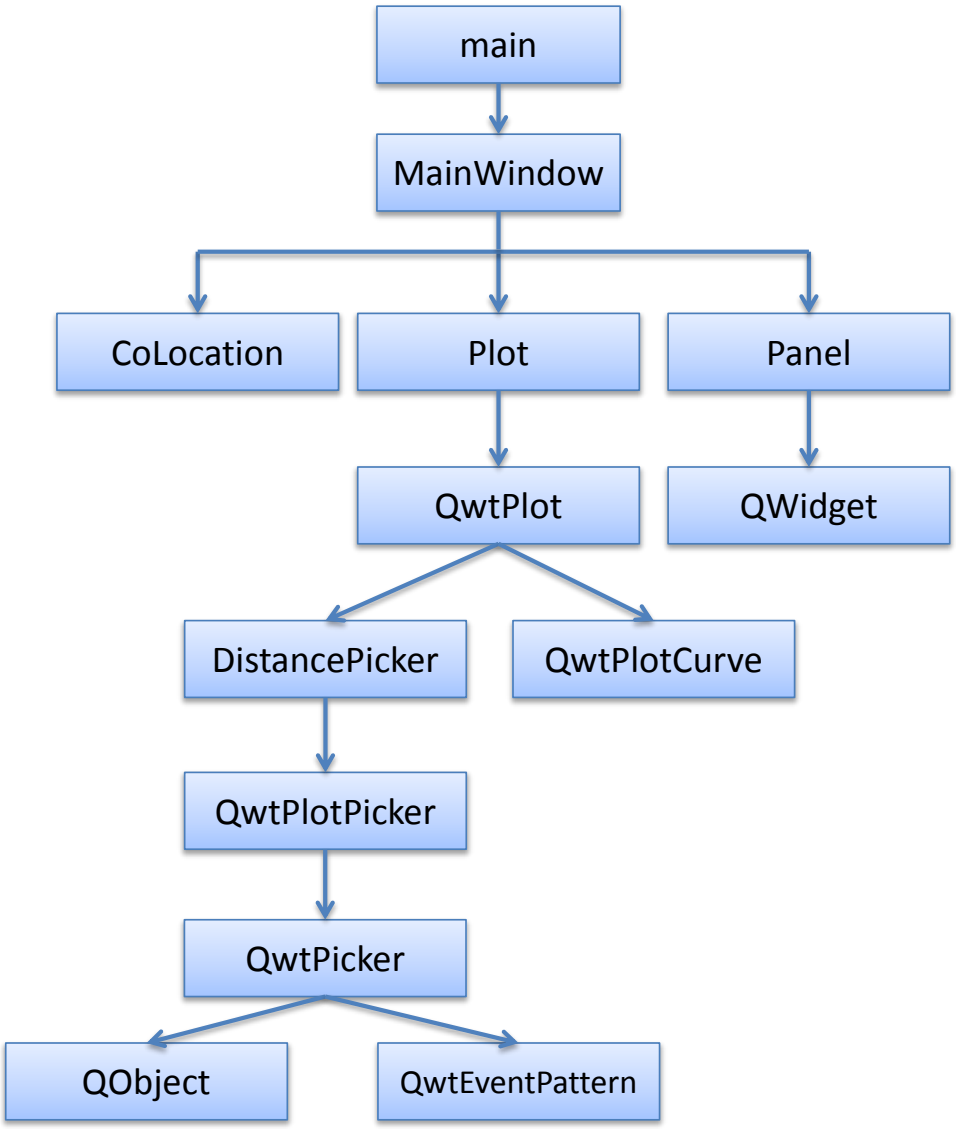

Figure 4.3: Co-location analysis tool - main classes

Figure 4.4 shows the main window that we designed for this tool. It is mainly composed of five parts: Menu, Menu-bar, Plotting board, Parameter input panel, Status bar. By either using Menu or Menu-bar, users can load *.kml file and analyze spatial data. Plotting board is used to visualize the spatial dataset and boundary polygon, as well as the experimental results. Parameter input panel enable users to select analysis method and configure experimental parameters. Status bar is used to update program status for users.

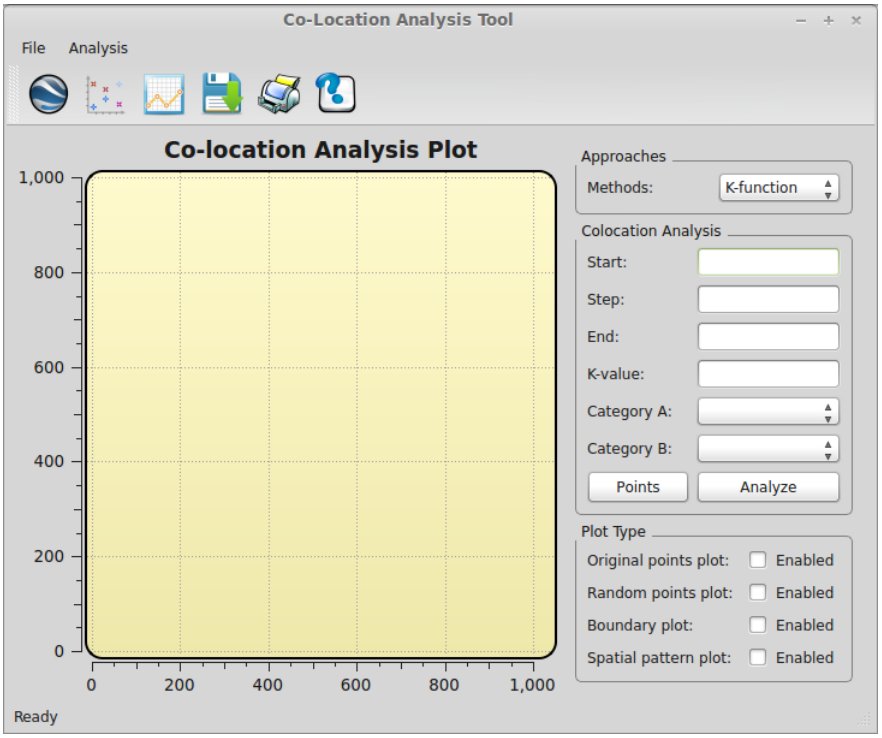

Figure 4.4: Welcome window of the co-location analysis tool

4.3 Co-location analysis work flow

First let's begin with presenting some basic concepts in this section. In a spatial dataset, we denote O to be a set of objects, where O={o1, o2, o3,....on}, and each object is tuple as <building_type, longitude, latitude>. In this thesis, we propose a GUI tool for spatial co-location pattern analysis. Two distance-based approaches are used: Ripley's K-function approach, K-Nearest-Neighbor distance approach, in addition, Euclidean distances are used as a distance function.

Based on the main functions of the tool, we summarize spatial co-location patterns mining work flow in 7 steps as follows:

Step 1: Load a KML format file and parse it into an intermediary *.dat file.

Step 2: Visualize the input spatial dataset.

Step 3: Use one of the spatial pattern analysis approaches to analyze the input spatial dataset.

Step 4: Create the boundary polygon for the input spatial dataset by using Andrew's monotone chain algorithm.

Step 5: Randomly place the objects in the space, and check whether each object is inside the boundary which is created in step 4. If not, re-place it again until all the objects are inside the boundary, and a new dataset is then generated.

Step 6: Use the spatial pattern analysis approach specified in step 3 for the new dataset that generated in step 5.

Step 7: Draw the co-location curves based on the results of step 3 and step 6.

The input and output file used and produced by the co-location analysis tool are

summarized below: Input:

1. A *.kml file.
2. A method for spatial pattern analysis.
3. Two categories based on input dataset.
4. Start radius.
5. Step.
6. Maximum radius.
7. Maximun k value.

Output:

1. An intermediate dat file that contains information we need.
2. An intermediate csv file that contains randomly placed objects.
3. Csv files contain spatial pattern analysis results.

4.4 KML file parsing

KML is an open standard officially named the OpenGIS KML Encoding Standard (OGC KML). It is maintained by the Open Geospatial Consortium, Inc. (OGC) [7]. The *kml file is readable by the GoogleEarth [6] for easy visualization. Figure 5.1

shows the effects of input file that we have for later spatial data analsysis. The *.kml file looks like an XML file extension; Although structurally they are very similar, Google Earth 5.0 has provided a number of new features for KML extensions. These extensions use the gx prefix and the following namespace URI:

```
xmlns:gx="http://www.google.com/kml/ext/2.2"
```

This namespace URI must be added to the <kml> element in any KML file using gx-prefixed elements:

```
<kml xmlns="http://www.opengis.net/kml/2.2"
 xmlns:gx="http://www.google.com/kml/ext/2.2">
```

Although the *.kml file is different from the *.xml file, we can pre-process the KML file directly like the XML file. There are a bunch of libraries that have been provided to parse the XML files. In our implementation, we first convert a KML to an XML file and then use those available libraries to parse the converted XML files. Minimal DOM implementation is a minimal implementation of the Document Object Model interface, with an API similar to that in other languages [9], we use the provided APIs together with Python programming language to help us extract information as input for the co-location analysis tool. Note that some redundant gx prefixes might exist in the provided *.kml file, so we need to remove redundant files during the parsing process.

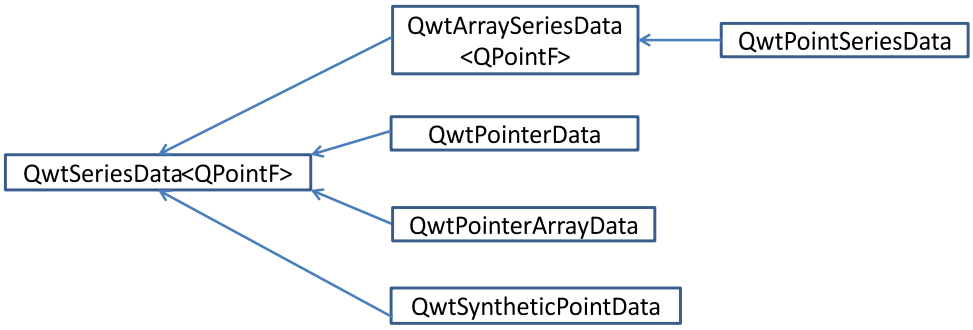

Figure 4.5: Qwt classes used to store points coordinate information.

4.5 Spatial data analysis and visualization

After parsing, building type and coordinate information for each object are kept for further visualization and analysis. To ease the process of visualization, we adopted a third-party library for spatial data points and experimental result visualization. Figure 4.5 shows the relationship of the classes which are used to store sampling data for the *Qwt* control visualization. Since we only need to keep x and y coordinate information, we naturally used the **QPointF** data type to store the coordinate information. Before we invoke the Qwt provided API *QwtSetSample()* to draw diagrams, also we are required to set up the data points visualization symbols, point color, and drawing style (e.g. lines, curves, points, etc.). Figure 4.6 lists the relationship bwtween drawing classes.

4.6 Evaluation and verification

After introducing key modules and functions used in the development of our tool, next we need to verify our tool and demonstrate how to use this tool to analyze

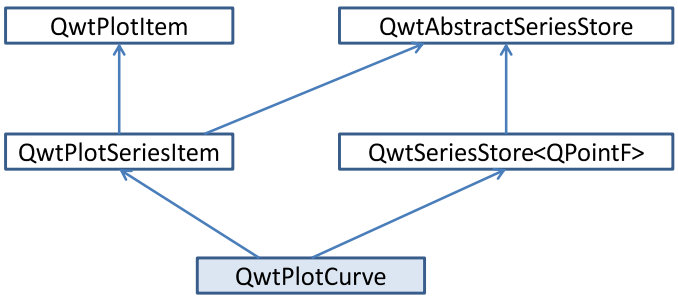

Figure 4.6: Qwt classes used for points and curves drawing.

co-location patterns. In order to verify the tool, we wrote a python script to intentionally create 100 objects placed into two circles. Each circle includes a red object surrounded by 49 green objects points with an radius less than 1. The coordinate of two red points are (1, 1) and (4, 1) respectively, which is shown in figure 4.7.

Figure 4.4 is the welcome window after running this tool. By default, it will select Ripley's K-function approach to analyze spatial patterns. Users can also change to a different method by selecting the drop-down list on the right configuration panel. Figure 4.8 shows the process that co-location analysis tool loads the data file which we created for verification. After loading the file, by clicking scatter-plot button on the menu bar, the tool will visualize the input spatial dataset. From figure 4.7, we can see that purple points represents the original input dataset. Based on the algorithm described in section 3.1, we can get the boundary points and then draw the polygon that incorporates all the scatter points. The red line in figure 4.10 shows the boundary polygon for the input dataset. After getting the boundary, next comes the step of generating randomly distributed points within the polygon. Green points in figure 4.11 shows the randomly generated points which are evenly distribute within this area incorporated by the red polygon lines. By enabling the "random point

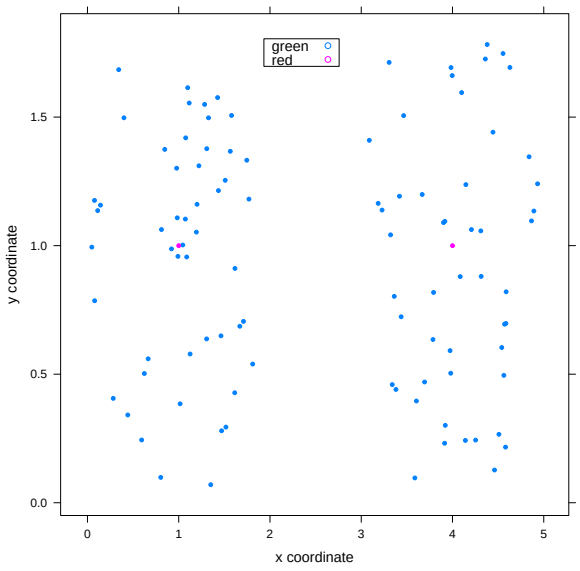

Figure 4.7: Visualization of verification points.

plot" check box on the right panel box, we can call random function and visualize the boundary polygon. Figure 4.12 shows the visualization of the original input dataset and randomly generated dataset together within the boundary polygon. The category drop-down lists in the right panel will automatically initialize the category types according to the input dataset. Therefore, users can specify the data analysis method and category types, start value, step_size, etc. to do further co-location analysis.

To evaluate the tool, we use a dataset that only has two categories: red objects and green objects; there are 100 objects in total. The test dataset we use looks like two circles; each circle has 50 objects (including one red object and 49 green objects) and the radius of each circle is less than 1. The two red objects are located in (1, 1) and (4, 1). Note that there are no points between 2 and 3 in the x axis direction.

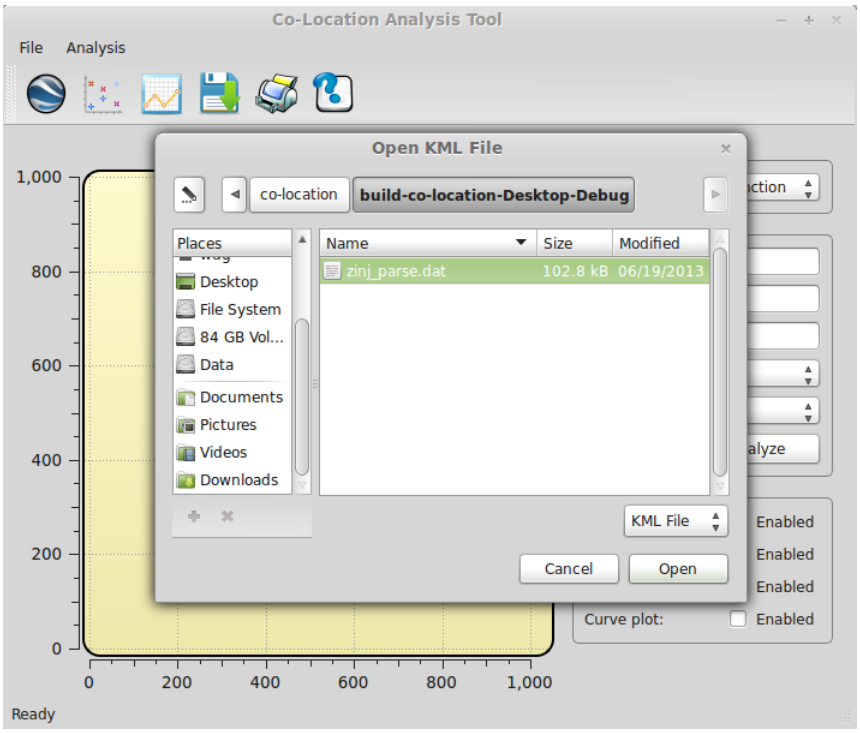

Figure 4.8: Visualization of spatial data from KML files.

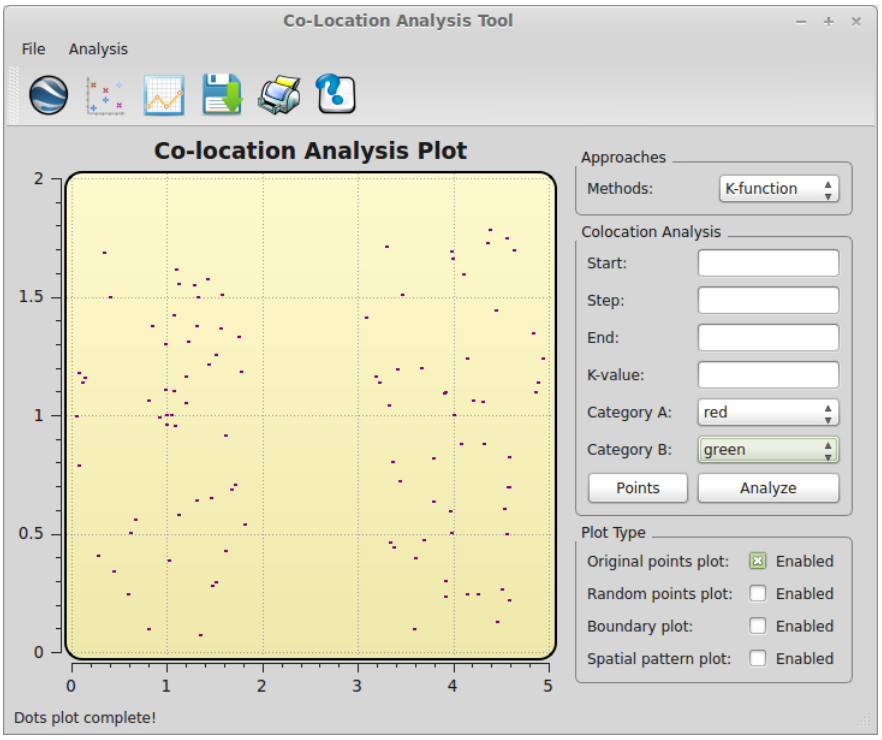

Figure 4.9: Visualization of spatial data from KML files.

Therefore, if we use Ripley's K-function approach to analyze co-location patterns, the purple curve, which represents the original input dataset patterns, is expected to increase by the same increment as the radius until the radius reaches 1. When the radius is equal to 1, the average percentage of green objects is expected to be 50%. Then before the radius reaches 2, the average percentage of green objects is expected remain 50%, because there is no green objects between the 2 and 3 in the x axis. After that, the purple curve is expected to increase as the radius increases. However, if the KNN distance approach is specified as the method to analyze input dataset, the purple curve is expected to increase little by little as the k value increments. When k equals 50, a big jump is expected in the curve, since the 50th object is located in the other circle. After that the curve is expected to increase as the k values increase

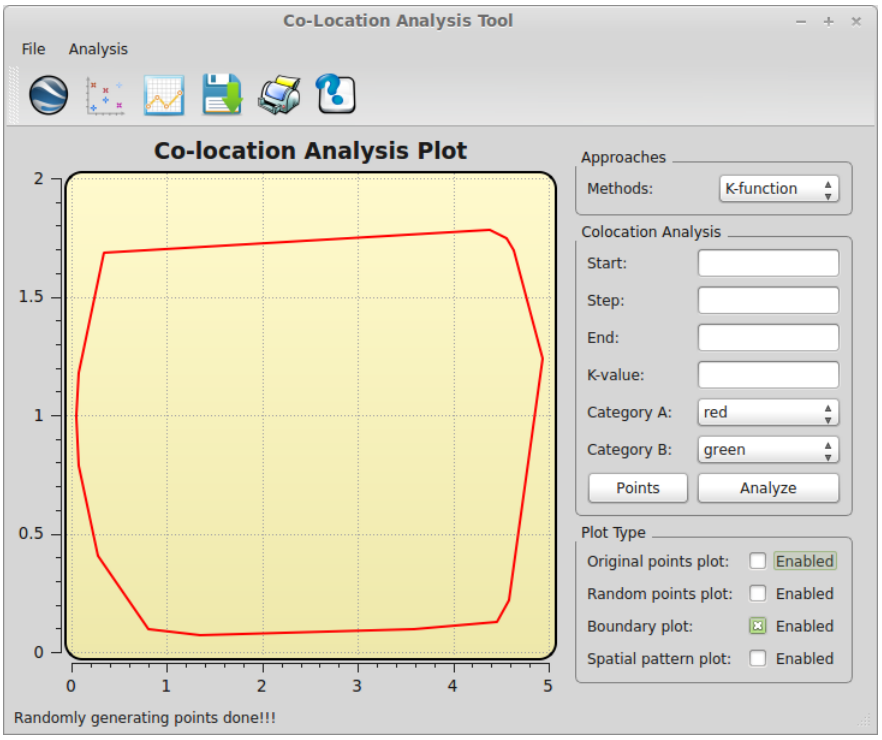

Figure 4.10: Visualization of polygon based on the input spatial data.

again.

Firstly, we evaluate the Ripley's K-function approach. We specify the start value to be 0.1, step size to be 0.1, end value to be 3.0, category A to be red, and category B to be green; we can then get the results of co-location analysis. Figure 4.13 shows the experimental of the test dataset. As we can see from the figure the purple curve, which represents the original input dataset, is increased as the radius increases; when the radius reached 1, it stopped increasing. Meanwhile the y axis value equals to 50. The purple curve is parallel to the x axis when the radius is from 1 to 2. It increases again after the radius reaches 2. Based on the observation above, the experimental result of the test dataset exactly agrees with the prediction, as we

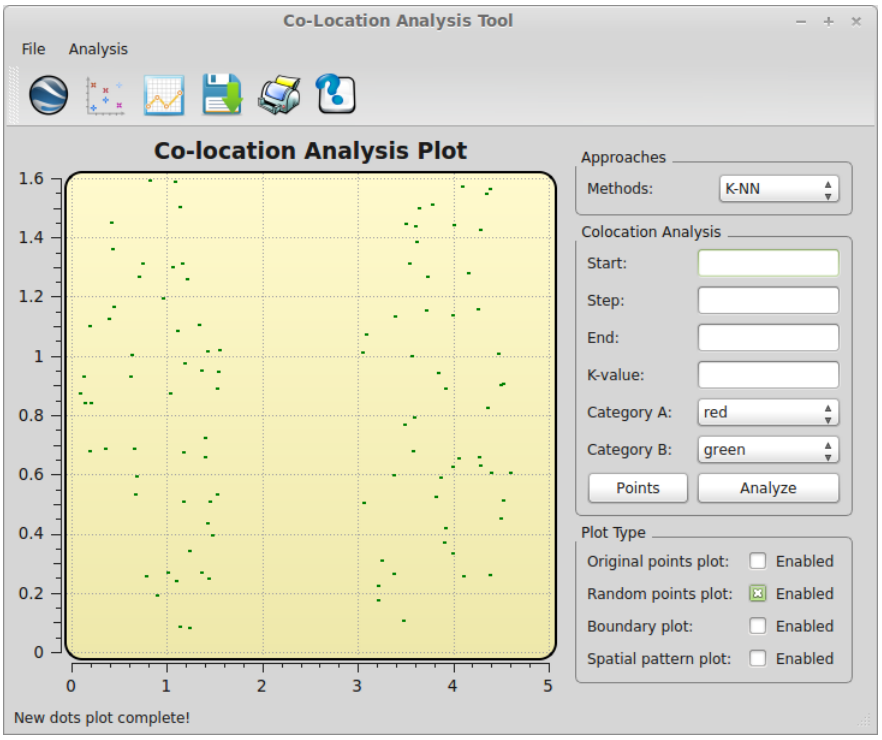

Figure 4.11: Visualization of randomly generated spatial points within polygon.

discussed previously.

Next, we are going to evaluate KNN distance approach. we specify the largest k to be 70. From figure 4.14 we can observe that the purple curve increases little by little as the k value increases. However, the curve has a big jump when k equals 50. After that, it increases slowly as the k value increases. This result is in agreement with the prediction as well.

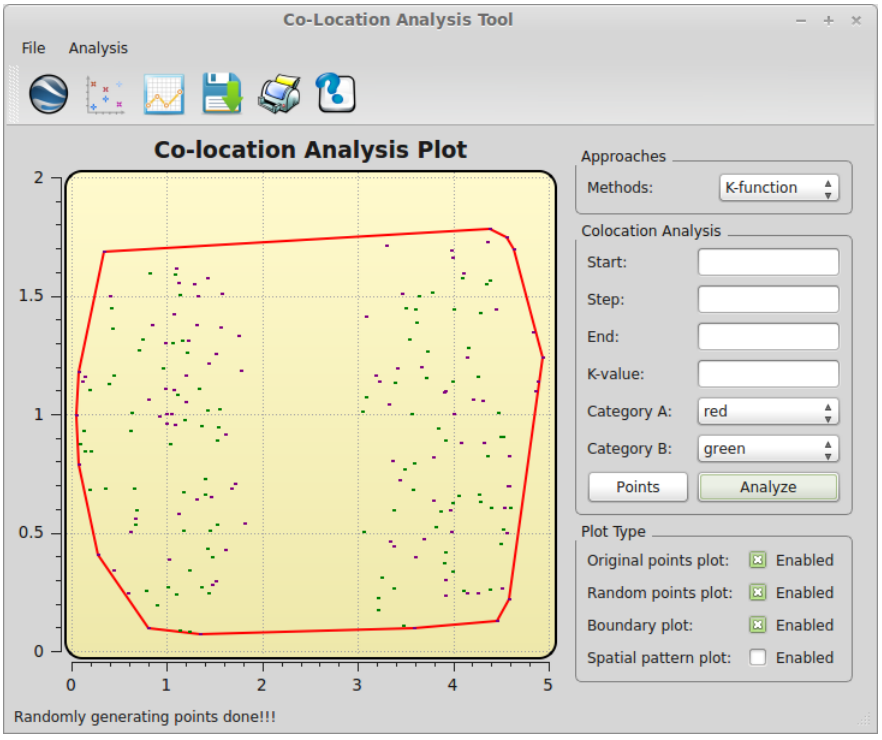

Figure 4.12: Visualization of randomly generated spatial points together with the original spatial points within polygon.

Table 4.1: Summarization of class functions.

Class	**Key Functions**
MainWindow	CreateStatusBar() Create_toolbar Print() Help() Save() LoadKML()
CoLocation	Read_points() Get_convex_hull() Is_point_in_polygon() Get_distance() KNN_Calculate_old_ratio() KNN_Calculate_new_ratio() Calculate_old_ratio() Calculate_new_ratio() DumpPointsInfo() Generate_points()
Plot	Plot() setPolygonSymbol() setPolygonSamples() setSymbol() setNewSamples() setOldRatioSamples() setNewRatioSamples() setSamples()

Panel	Q_SIGNALS:
	void dot_selected();
	void boundary_selected();
	void new_dot_selected();
	void curve_selected();
	void method_selected();
	void edited();
	void cate_A_edited();
	void cate_B_edited();
	void generate_points();
	void do_analysis();

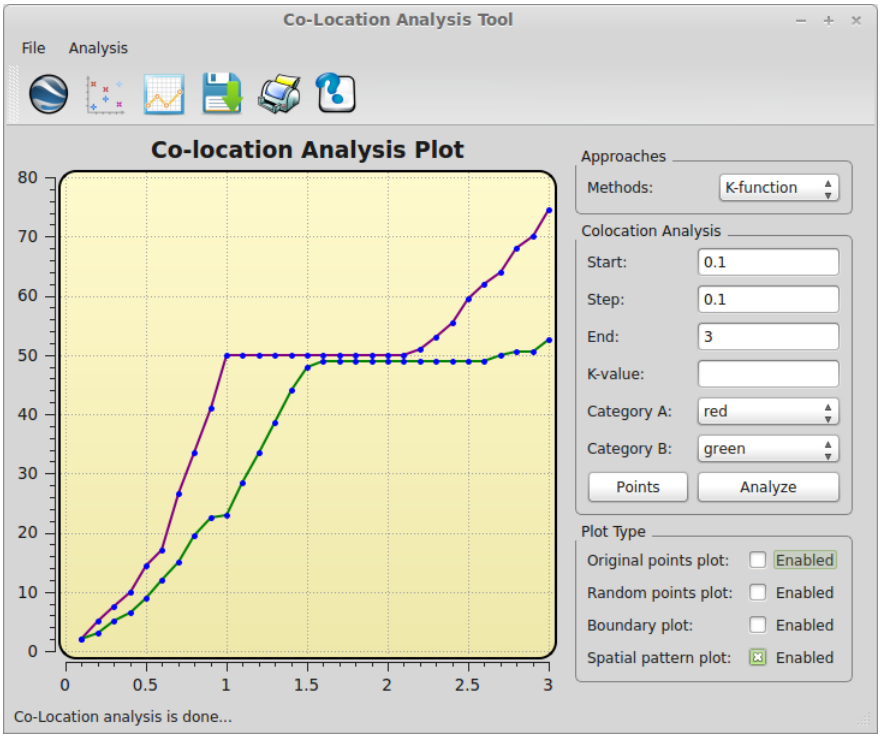

Figure 4.13: Verification of co-location analysis by using the K-function algorithm.

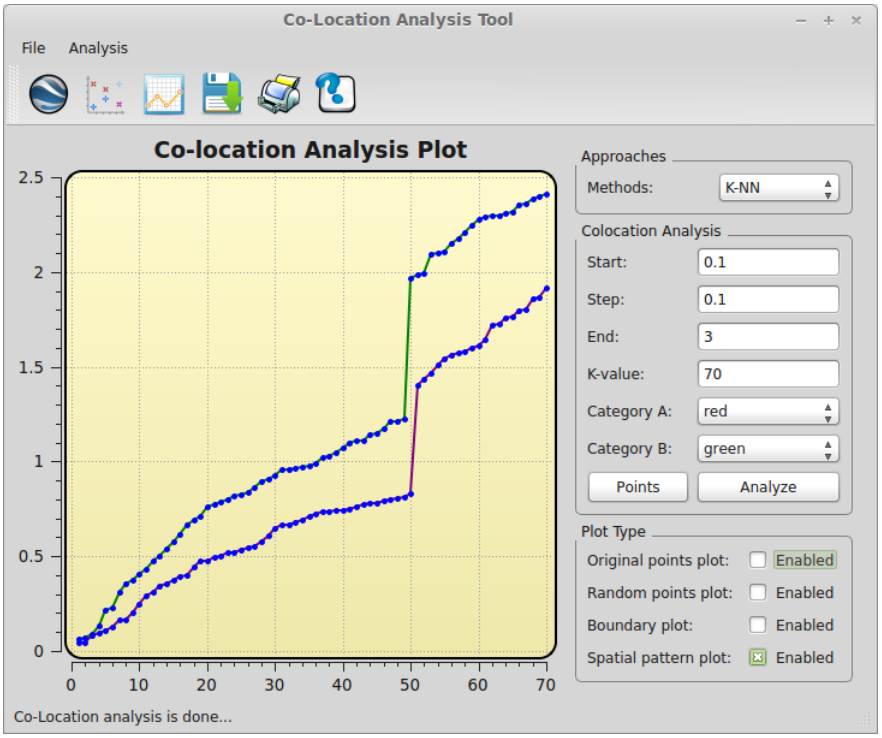

Figure 4.14: Verification of co-location analysis by using the KNN algorithm.

Chapter 5

Experiment Results

The goal of this chapter is to demonstrate the co-location analysis tool, which displays the experiment results for several different cases, and compares the accuracy and efficiency of two co-location analysis approaches.

5.1 Summary of experiment dataset

In our experiments, a dataset which contains different types of buildings in a neighborhood of the city of Strasbourg in France is used. The dataset is a KML file which contains building location (longitude, latitude) information, building types, building area, etc..

In our analysis, we particularly focus on the analysis of co-location patterns for the following six major types of buildings from the city of Strasbourg:

- Single houses, represented by the green color.
- Garages, represented by the red color.
- Commercial buildings, represented by the yellow color.
- Light buildings, represented by the blue color.
- Collective houses, represented by the pink color.
- Schools, represented by the cyan color.

The neighborhood of Strasbourg in the dataset is visualized in Figure 5.1:

Figure 5.1: A neighborhood of city of Strasbourg visualized by Google Earth

5.2 Experiment result

In our experiments, we conduct spatial analyses based on Ripley's K-function approach and K-Nearest-Neighbor distance approach with respect to different types of buildings in a data set. The number of each type of building is shown in Table 5.1.

Table 5.1: Building numbers of different building types

Building type	**Number**
single house	1598
garage	81
commercial building	133
light building	56
collective house	157
school	14
sport building	2
church	2

When using the co-location analysis tool, users first need to select a co-location analysis method that they want to use. If users choose to use the Ripley's K-function approach, they need to further specify other parameters, including a starting distance (the distance unit is based on each case), a distance increment, and an ending distance. Users also need to specify the two categories they want to evaluate. These two categories represent two types of objects that users are going to analyze. There are six major building types in our experimental dataset; we could select any two of the building types to analyze their co-location patterns based on Ripley's K-function.

K-Nearest-Neighbor distance (KNN for short) approach is an alternative approach which is used for the first time to analyze co-locations patterns. To use the KNN distance approach, users need to specify the largest k values, and two categories

they want to analyze: *category A* and *category B*. These two categories represent two types of objects that users are going to analyze their co-location patterns.

If the method is specified as Ripley's K-function, then the x-coordinate represents the radius and the y-coordinate represents the average percentage of objects in category B.

If the method is specified to be the KNN distance approach, then the x-coordinate represents the k value and the y-coordinate represents the average distance of Kth nearest object in category B. By using these two approaches, the co-location analysis tool can summarize spatial patterns (co-located or anti-co-located) over a range of distances.

In our experiments, five different cases are analyzed, which are listed below:

- Case one: garage, single house
- Case two: garage, collective house
- Case three: commercial building, light building
- Case four: commercial building, school
- Case five: collective house, garage

For each case, two different co-location analysis approaches are used. Please note that the all radii unit that I used in the experiments is mired (micro-reciprocal-degree). For the plot part in our tool, note that the unit of x-axis is mired and the unit of y-axis is percentage.

5.2.1 Case one

In case one, we analyze co-locations between garages and single houses. The distribution of original input dataset of these two types is displayed by figure 5.2. We can see that on the left-bottom section of the figure,there are some red points surrounded by few green points. Figure 5.3 shows scatter plot of garages and single houses after they are randomly placed in the same region, we can observe that both of these two types of buildings are uniformly distributed in this figure.

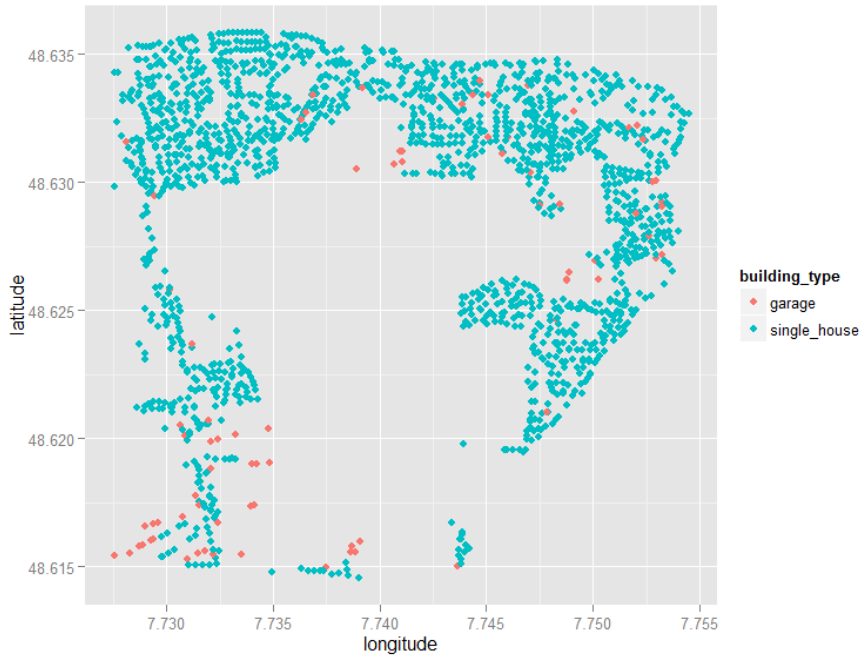

Figure 5.2: Distribution of garages and single houses in original input dataset

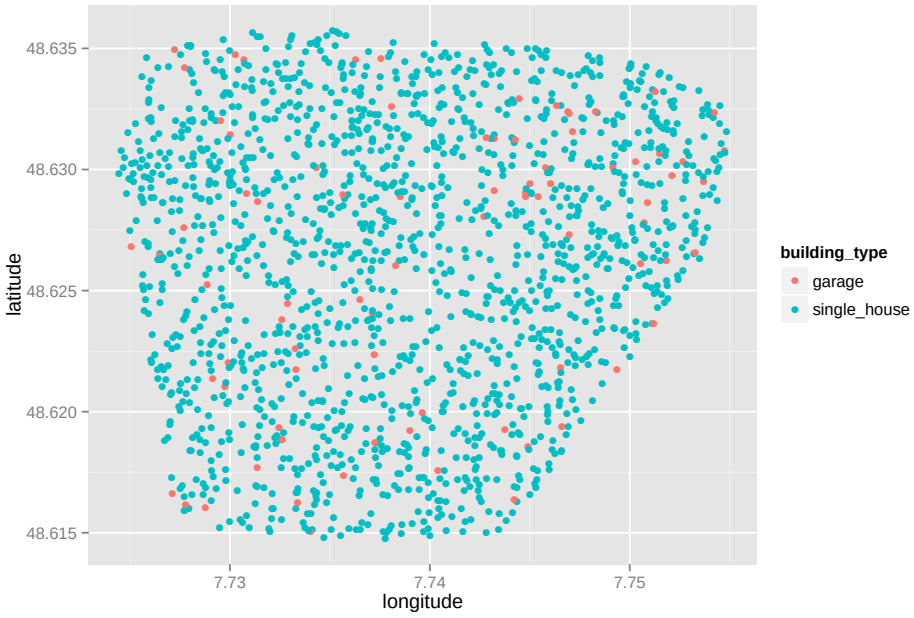

Figure 5.3: Random distribution of garages and single houses

5.2.1.1 Using Ripley's K-function approach for case one

Garage and single house are specified to be category A and category B respectively in this experiment. First, Ripley's K-function is selected as the method to analyze spatial patterns. Before using this method, the starting radius is specified to 0, the distance increment is set to 0.001, and the ending radius is set to 0.032. The figure 5.4 below shows the result of using the Ripley's K-function.

In figure 5.4, the purple curve represents the observed spatial pattern, and the green curve represents the expected random spatial pattern. From figure 5.4 below, we can see that the purple curve is slightly above the green curve over the range 0 to 0.006, which means the observed spatial pattern is slightly larger than the expected spatial pattern in this scale of distance. Where the radius ranges from 0.006 to 0.03,

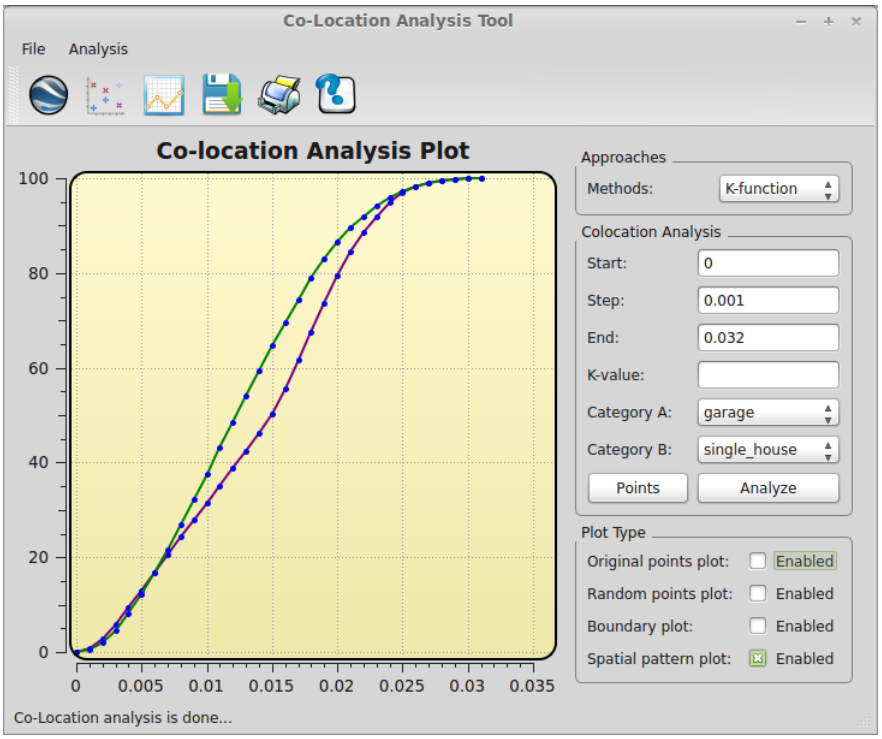

Figure 5.4: Ripley's K-function case 1: garages vs single houses

the purple curve is below the green one. These two curves overlap and reach 100% after the radius reaches to 0.03. Therefore, we can reach a conclusion that when the radius ranges from 0 to 0.006, the distribution of garages and single houses are considered very slightly co-located. These two types of buildings are more anti-co-located than a random distributed spatial pattern with radius of 0.006 to 0.03.

5.2.1.2 Using KNN distance approach for case one

As with the Ripley's K-function, the two categories are specified to be garage and single house; thus, by using this approach we will discover the 1th, 2th, 3th,....., Kth nearest single house for each garage and compute the average distance between

them. In case one, in order to get the overview result of all instances of single house, k value is specified to be 1598, which is the total number of single houses in the input data set. If users choose a k value larger than the total number of the input dataset, it will not influence the result, because the tool will automatically stop when k reaches to the total number of the instances. Figure 5.5 below shows the result of using KNN distance approach on case 1.

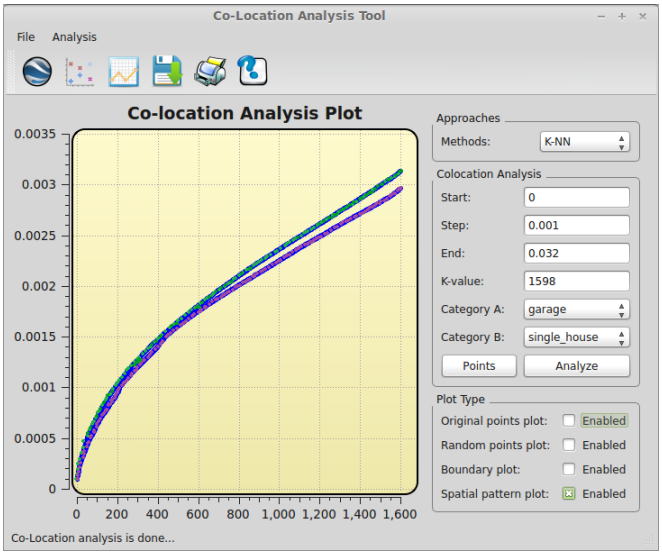

Figure 5.5: KNN distance approach on case 1: garages vs single houses

Table 5.2: Statistical data of KNN distance approach for case one

K value	10	240	580	713	918	1322
Observed average distance	0.000205	0.001079	0.001728	0.001904	0.002155	0.002633
Expected average distance	0.000227	0.001142	0.001781	0.001982	0.002262	0.002769

In figure 5.5, the purple curve represents the observed spatial pattern and the

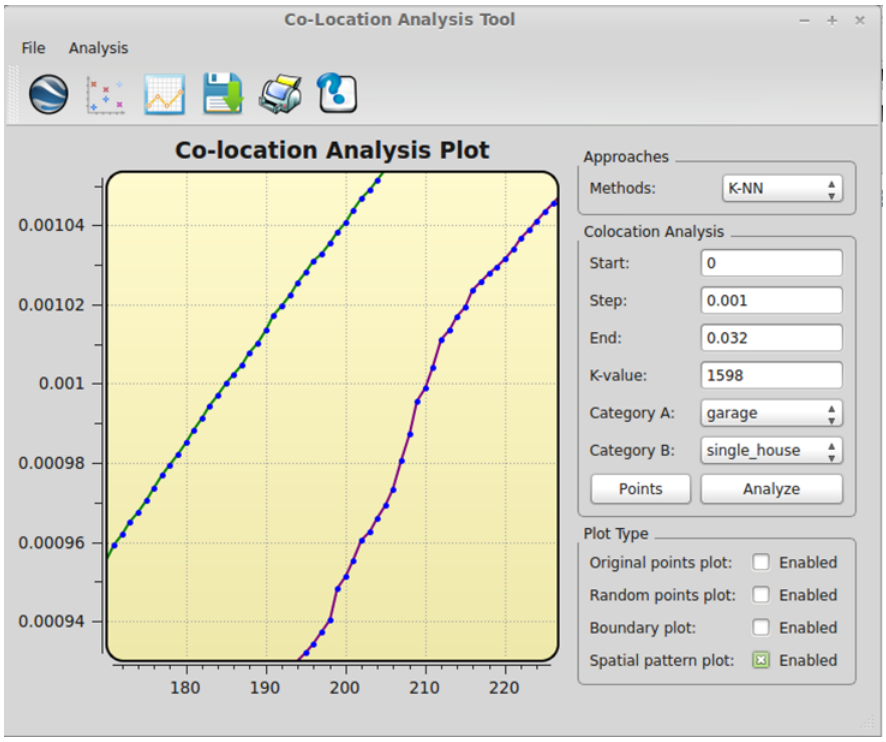

Figure 5.6: Zoomming in result of KNN distance approach on case 1

green curve represents the expected random spatial curve. When the purple curve is below the green one, it means the average distance of the observed distribution is smaller than the average distance of the expected random distribution; therefore, the two categories are considered co-located. On the other hand, when the purple curve is above the green one, then the average distance of the observed distribution is larger than the average distance of the expected random distribution.

Figure 5.5 shows the result of the KNN distance approach, but these two curves are not clearly displayed. In order to clearly observe the result, we zoom in on the figure 5.6, where we can tell the green curve is everywhere above the purple one. This result indicates that garages are slightly co-located with single houses with this

approach. Due to a large range of k values, only a few of the k values and their corresponding observed, and expected average distances are listed in Table 5.2.

5.2.1.3 Comparing the two approaches in case one

As we discussed above, when using the Ripley's K-function approach on case one, we get the conclusion that garages and single houses are slightly co-located over the range 0 to 0.006, but from 0.006 to 0.03, these two types of buildings are anti-co-located with each other. If we use KNN distance approach, the results indicate that these two types of buildings are slightly co-located with each other over the range 0 to 0.03. Based on the experiment results, these two approaches partially agree with each other. The possible reason why they disagree with each other for larger distance is that KNN distance approach only considers the shortest range of distance among objects, whereas Ripley's K-function approach considers a wider range of radius based on all distances between objects in the area of interest. In order to compare the efficiency of these two approaches, we also list the wall clock time of each approach in Table 5.3.

Table 5.3: Average execution time of both approaches

Approach	Ripley's K-function	KNN distance
Execution time	1.394 seconds	2674.23 seconds

From the table, we can observe that Ripley's K-function approach is much more efficient than KNN distance approach in case one.

5.2.2 Case two

The building types of garage and collective house are analyzed in case two. The distribution of the original input dataset of these two types is displayed in figure 5.7. We can observe that on the left-bottom section of the figure, the blue points are somehow frequently located with red points . Figure 5.8 shows the scatter plot of garages and collective houses after they are randomly placed in the same region. Both of these two types of buildings are uniformly distributed in this figure.

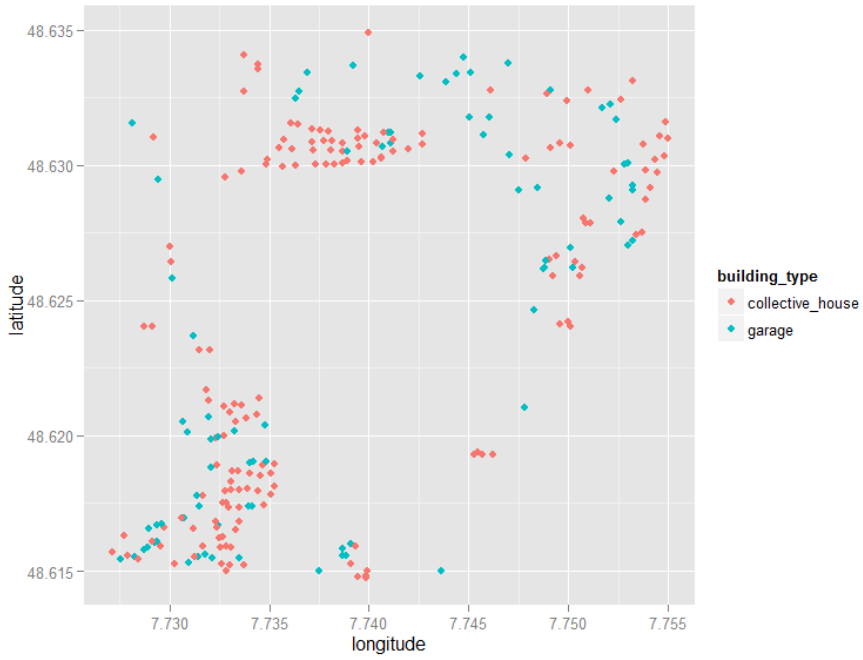

Figure 5.7: Distribution of garages and collective houses in original input dataset

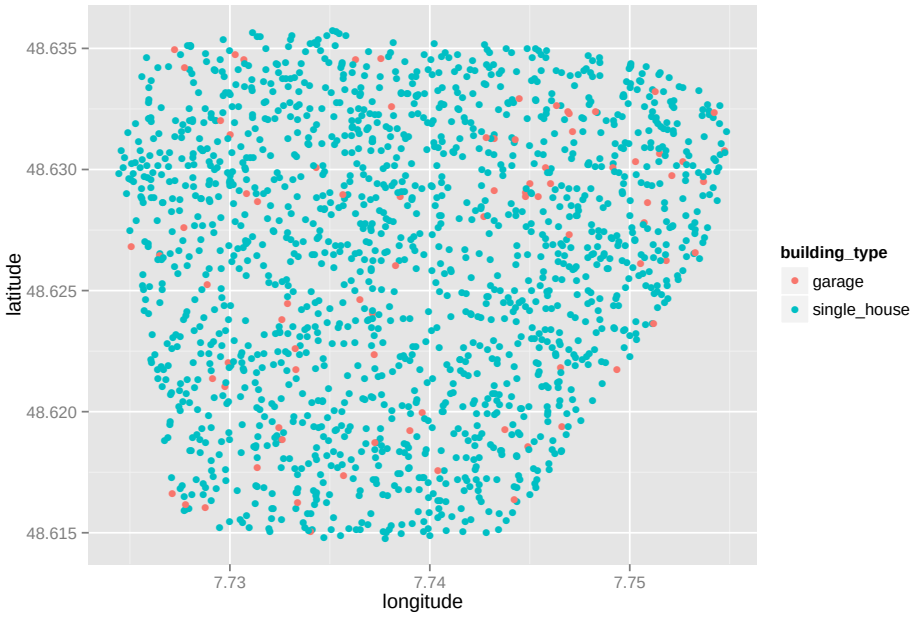

Figure 5.8: Random distribution of garages and collective houses

5.2.2.1 Using Ripley's K-function approach for case two

Garage and collective house are specified to be category A and category B respectively in this case. We first use Ripley's K-function to analyze spatial patterns for case two. Before using this method, the begining radius is specified to be 0, a distance increment is set to be 0.001 and ending radius is 0.032. Figure 5.9 below shows the result of using Ripley's K-function approach.

From figure 5.9 below, we can easily find that the purple curve is above the green curve when radius ranges from 0 to 0.009, which indicates the observed spatial pattern is more co-located than the expected spatial pattern in this scale of distance. But when radius ranges from 0.009 to 0.03, the purple curve is below the green one, and they reach 100% after radius comes to 0.03. Thus, we can get the conclusion

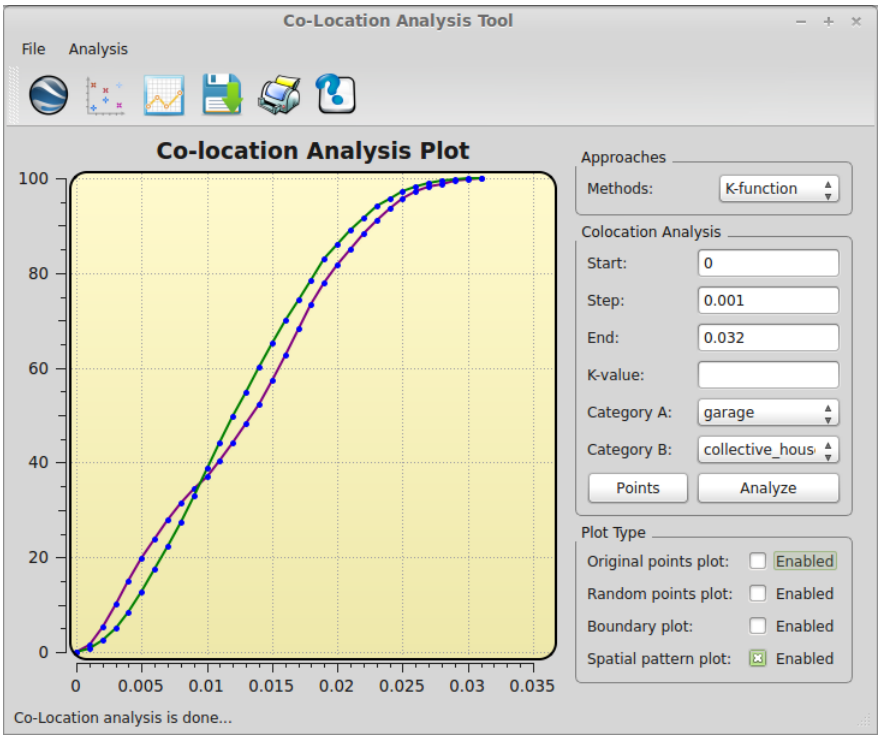

Figure 5.9: Ripley's K-function case 2: garages vs collective houses

that when distance ranges from 0 to 0.009, the distribution of garages and collective houses are co-located, but they are anti-co-located where the radius ranges from 0.009 to 0.03.

5.2.2.2 Using KNN distance approach for case two

In case two, the k value is specified to be 157 because there are 157 collective houses in the input dataset. Figure 5.10 below shows the result of using the KNN distance approach for case 2.

From figure 5.10, we can observe that the purple curve is below the green one. This result indicates that garages are co-located with collective houses where distance

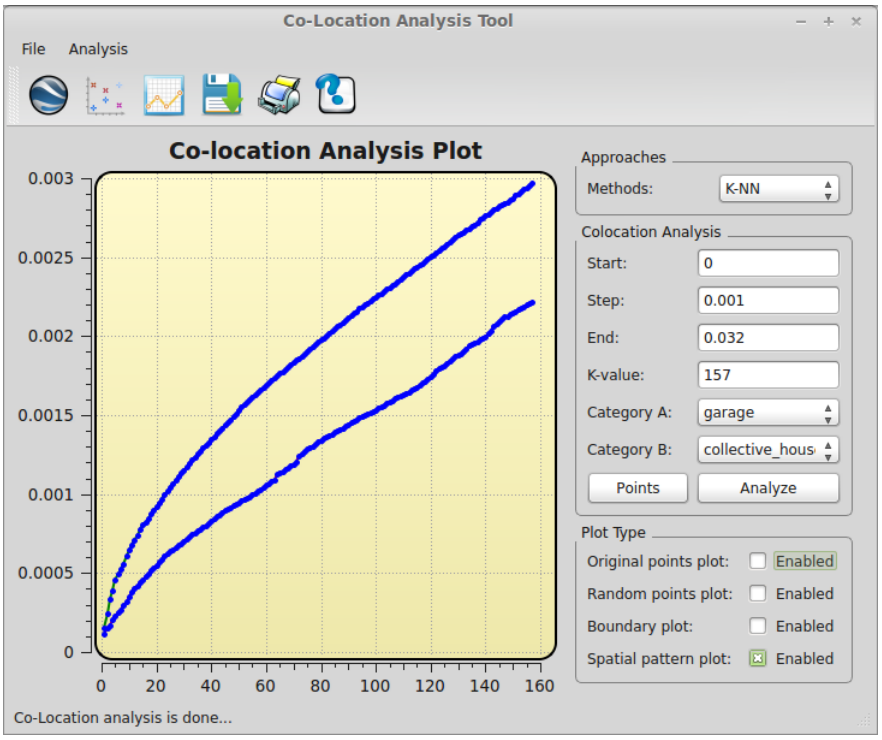

Figure 5.10: KNN distance approach case 2: garages vs collective houses

ranges from 0 to 0.03. Selected k values and their corresponding observed, and expected average distances are listed in Table 5.4.

5.2.2.3 Comparing the two approaches in case two

As we discussed above, when using Ripley's K-function approach for case two, we can get the conclusion that garages and collective houses are co-located in the distance range 0 to 0.009, and anti-co-located with each other with distance from 0.006 to 0.03. If we use the KNN distance approach, the results indicate that these two types of buildings are co-located with each other within the 0 to 0.03 distance range. Based on the experiment results, these two approaches disagree with each other most of the

Table 5.4: Statistical data of KNN distance approach for case two

K value	12	36	76	98	125	152
Observed average distance	0.000403	0.000777	0.001259	0.001518	0.001816	0.002167
Expected average distance	0.000708	0.001269	0.001920	0.002220	0.002568	0.002903

time. Therefore, these two approaches only agree with each other in part. Where larger distances are considered, they disagree with each other in this case. We also list the execution time of each approach in Table 5.5.

Table 5.5: Average execution time of both approach of case two

Approach	Ripley's K-function	KNN distance
Execution time	1.145 seconds	30.861 seconds

In this case Ripley's K-function approach is much more efficient than KNN distance approach.

5.2.3 Case three

The building types of commercial building and light building are analyzed in case three. The distribution of the original input dataset of these two types is displayed in figure 5.11. It is easy to find that if we draw circles for each red points, there are only very few blue pints inside each circle. The figure 5.12 shows scatter plot of commercial buildings and light buildings after they are randomly placed in the same region. It shows that these two types of buildings are uniformly distributed in this

figure.

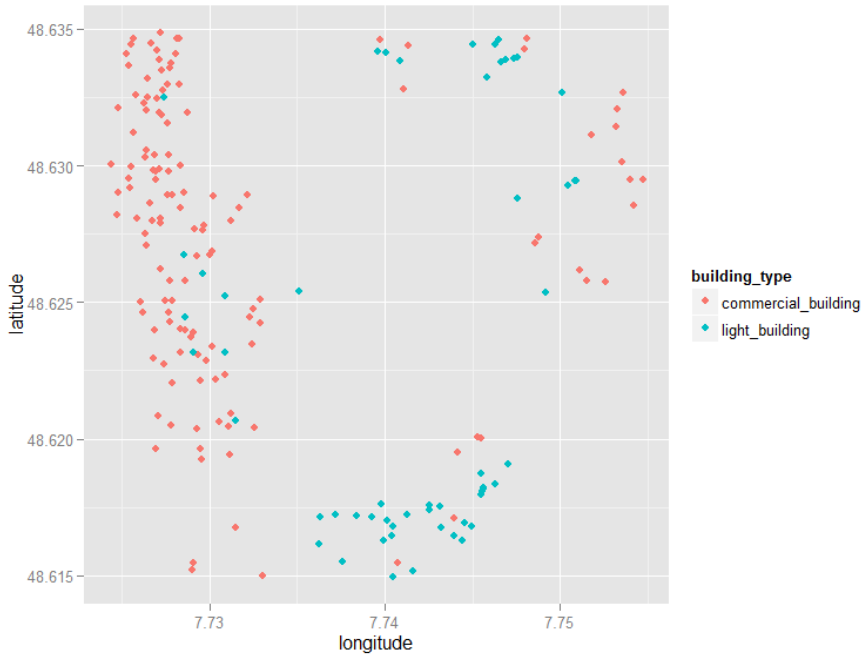

Figure 5.11: Distribution of commercial buildings and light buildings in original input dataset

5.2.3.1 Using Ripley's K-function approach for case three

Commercial buildings and light buildings are specified to be category A and category B respectively in this case. We first use Ripley's K-function to analyze spatial patterns for case three. Before using this method, the starting radius is specified to be 0, a distance increment is set to be 0.001, the ending radius is 0.032. The result is illustrated in the figure 5.13.

From the figure 5.13 below, we can see that the purple curve is below the green

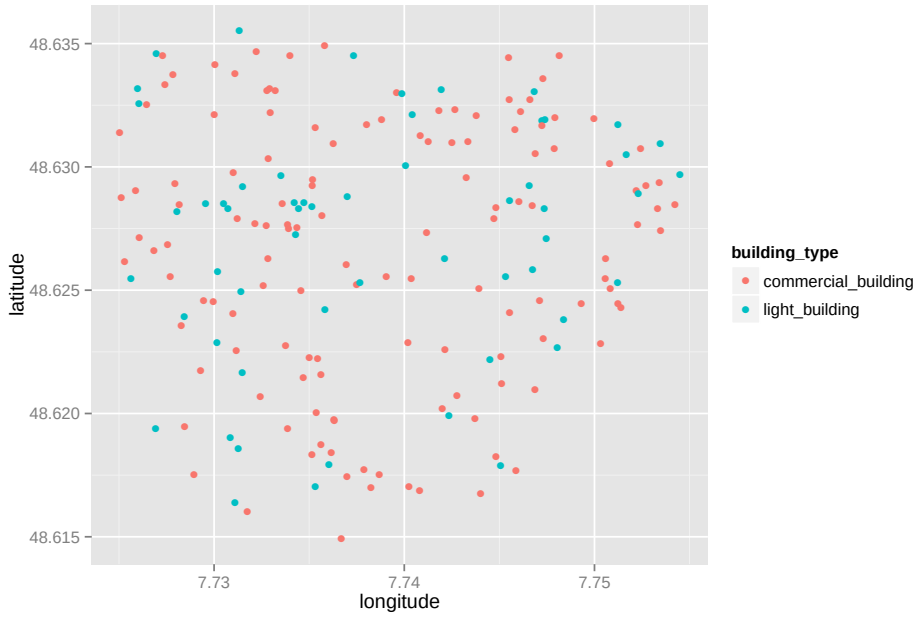

Figure 5.12: Random distribution of commercial buildings and light buildings

curve where the radius ranges from 0 to 0.025, which means these two types of buildings are anti-co-located in this scale of distance. Where the radius ranges from 0.025 to 0.03, the purple curve is slightly above the green one. These two curves overlap and reach 100% where the radius reaches 0.03. So, we can reach the conclusion that where distance range from 0 to 0.025, the distribution of commercial buildings and light buildings are anti-co-located, but they are considered very slightly co-located where distance ranges from 0.025 to 0.03.

5.2.3.2 Using KNN distance approach for case three

In case three, k value is specified to be 56, becasuse the total number of light buildings in original input dataset is 56. Figure 5.14 below shows the result of using the KNN

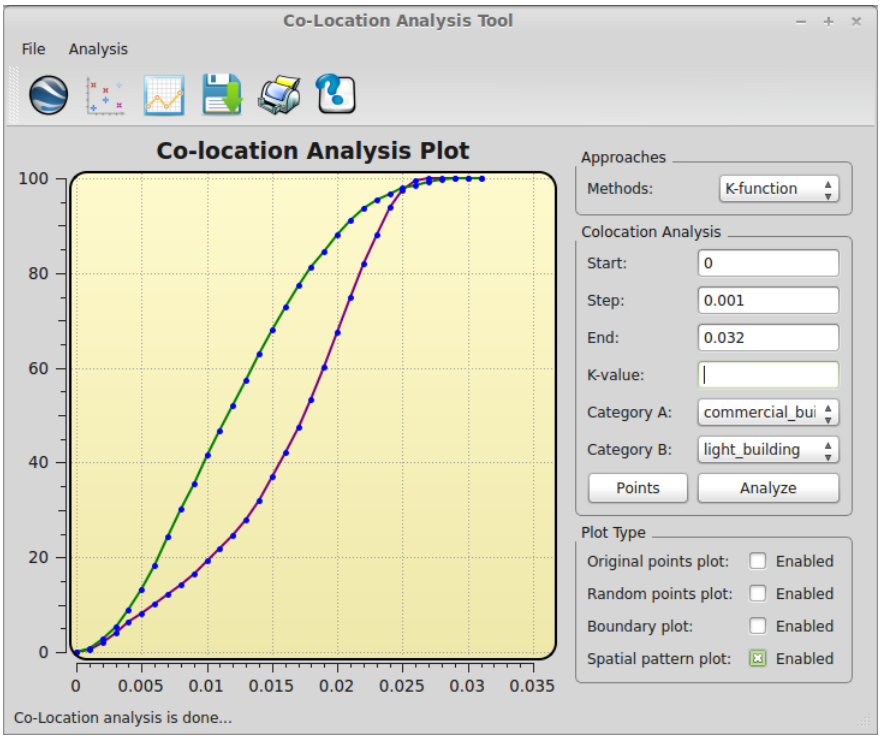

Figure 5.13: Ripley's K-function case 3: commercial buildings vs light buildings

distance approach on case three.

Figure 5.14 shows the result of the KNN distance approach, where we can observe that the purple curve is above the green one. This result indicates that commercial buildings are anti-co-located with light buildings in the distance range of 0 to 0.025, and they are very slightly co-located in the distance range of 0.025 to 0.03. Table 5.6 lists selected of k values and their corresponding observed, and expected average distances for case three.

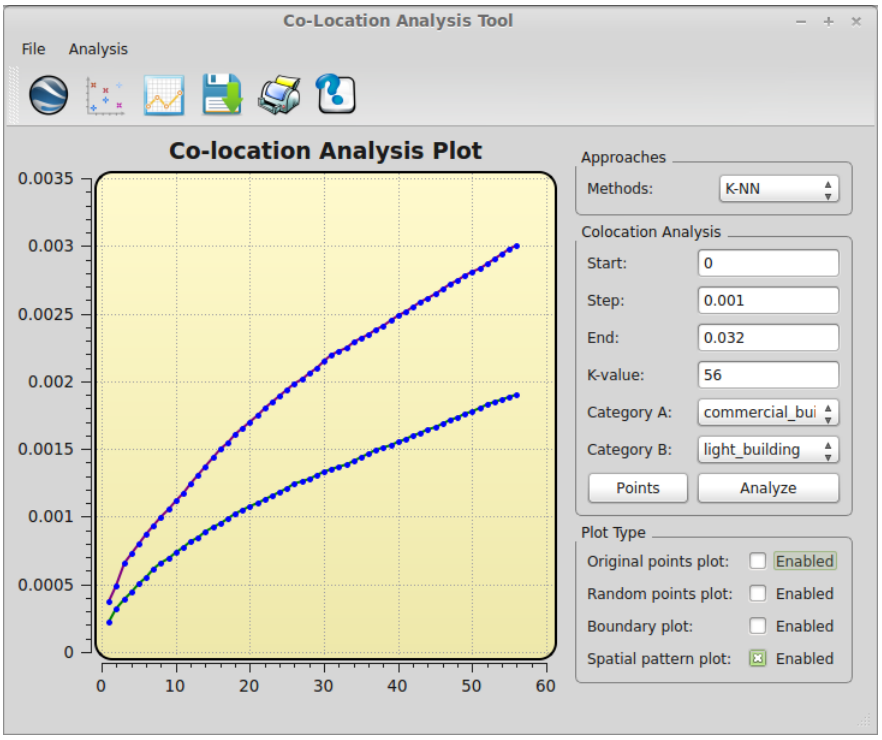

Figure 5.14: KNN distance approach case 3: commercial buildings vs light buildings

5.2.3.3 Comparing the two approaches in case three

As we discussed above, when using Ripley's K-function approach on case two, we get the conclusion that commercial buildings and light buildings are considered anti-co-located over the distance of 0 to 0.025, and slightly co-located with each other with distance from 0.025 to 0.03. If the KNN distance approach is used in this case, the results show that these two types of buildings are anti-co-located with each other throughout all distance from 0 to 0.03. Based on the experiment results, these two approaches mostly agree with each other. In order to compare the efficiency of these two approaches, we also list the average wall clock time of each approach in Table 5.5.

From the table, we can observe that Ripley's K-function approach is much more

Table 5.6: Statistical data of KNN distance approach for case three

K value	3	18	25	34	47	52
Observed average distance	0.000657	0.001604	0.001937	0.002291	0.002715	0.002872
Expected average distance	0.000390	0.001019	0.001212	0.001411	0.001711	0.001828

Table 5.7: Average execution time of both approach of case three

Approach	Ripley's K-function	KNN distance
Execution time	1.706 seconds	16.376 seconds

efficient than KNN distance approach in case three.

5.2.4 Case four

In case four, the building types of commercial building and school are analyzed. The distribution of the original input dataset of these two types is displayed in figure 5.15. Obviously there are very few blue points located with red points. Figure 5.16 shows a scatter plot of commercial buildings and schools after they are randomly placed in the same region; both types of buildings are uniformly distributed in this figure.

5.2.4.1 Using Ripley's K-function approach for case four

Commercial buildings and schools are specified to be category A and category B respectively in this case. We first use Ripley's K-function to analyze spatial patterns for case three. Before using this method, the starting radius is specified to be 0, a

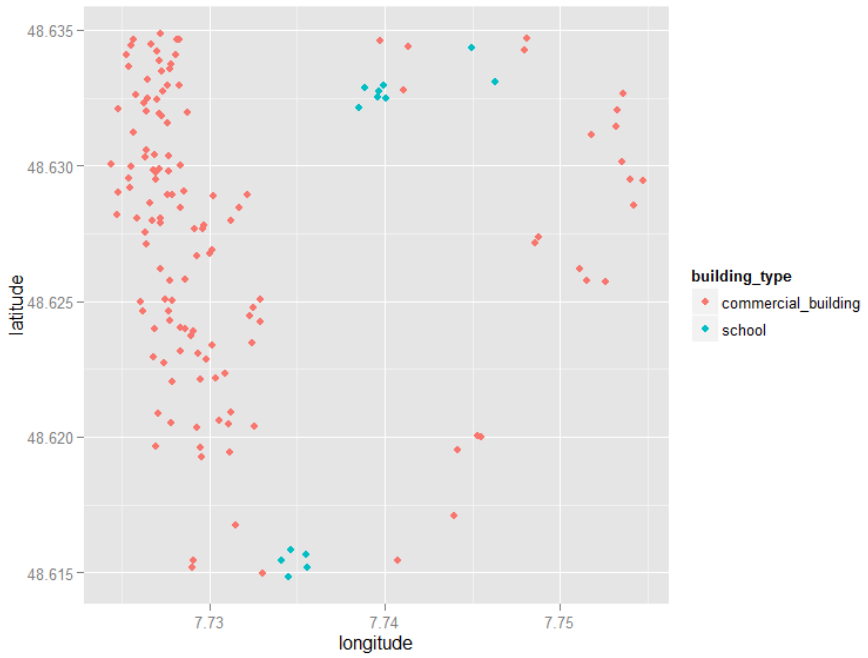

Figure 5.15: Distribution of commercial buildings and school in original input dataset

distance increment is set to be 0.001 and ending radius is 0.032. Figure 5.17 below shows the result of using Ripley's K-function approach in case four.

From figure 5.17 below, we can see that the purple curve is below the green curve where radius ranges from 0 to 0.03, which means the observed spatial patterns are anti-co-located in this range of distances. These two curves overlap and reach 100% where the radius approaches 0.03. Therefore, commercial buildings and schools are considered anti-co-located in the range 0.025 to 0.03 by referring to figure 5.17.

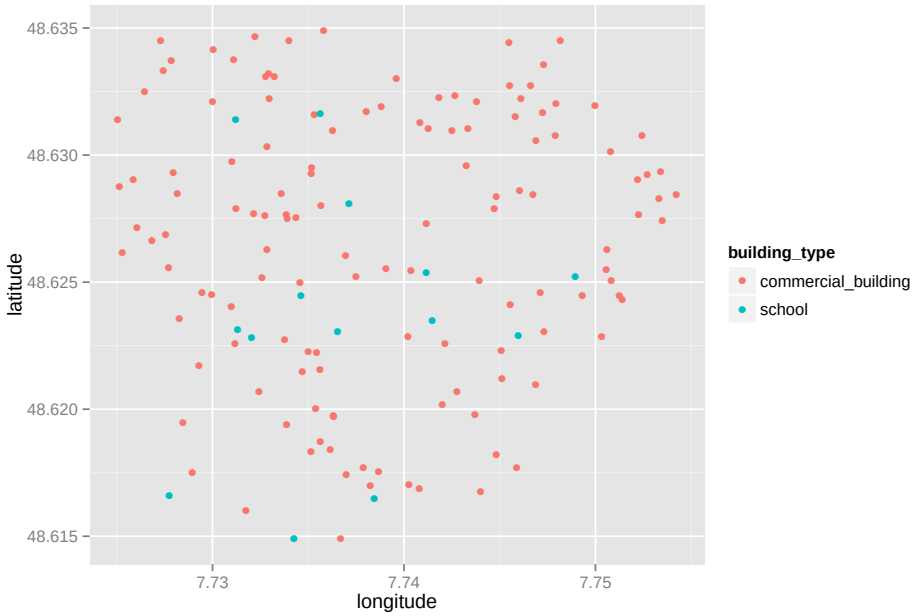

Figure 5.16: Random distribution of commercial buildings and schools

5.2.4.2 Using KNN distance approach on case four

In case four, the k value is specified to be 14, as the total number of schools in the original input dataset is 14. Figure 5.18 below shows the result of using the KNN distance approach on case four.

Table 5.8: Statistical data of the KNN distance approach for case four

K value	2	5	6	9	11	14
Observed average distance	0.004938	0.005391	0.005455	0.006083	0.006859	0.007092
Expected average distance	0.000948	0.001375	0.001516	0.001854	0.002087	0.002418

Figure 5.18 shows the result of the KNN distance approach. From the figure we

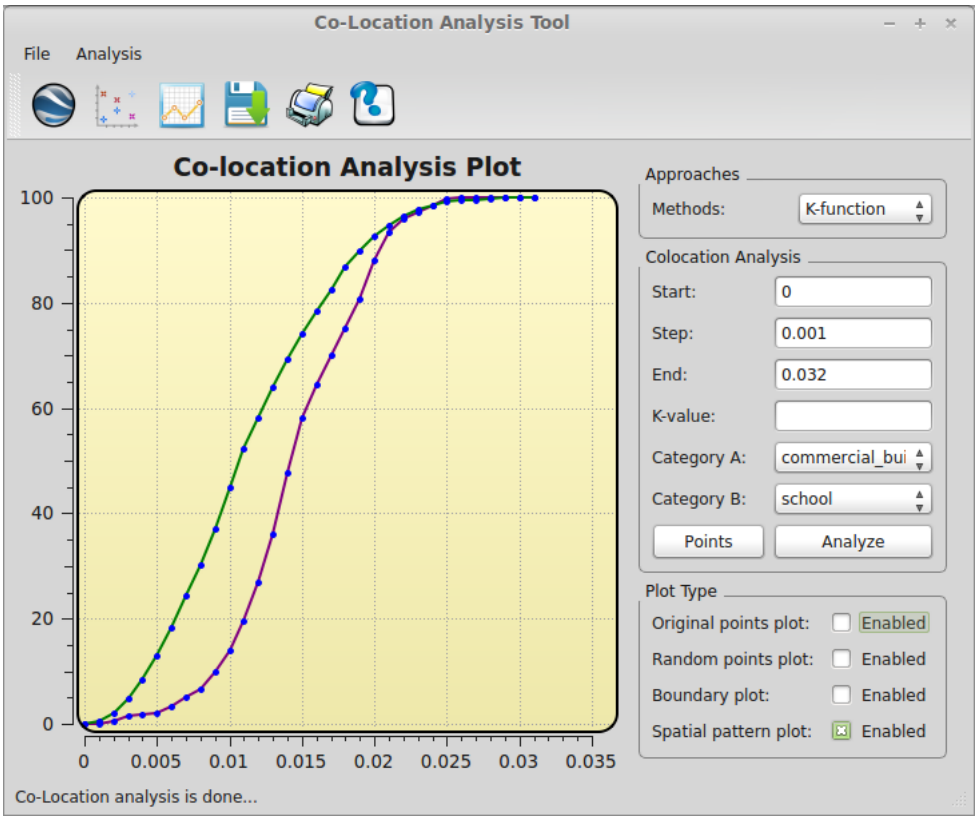

Figure 5.17: Ripley's K-function case 4: commercial buildings vs schools

can observe that the purple curve is above the green one, indicating that commercial buildings are anti-co-located with schools over the range 0 to 0.03. Experimental results for selected k values and their corresponding observed, and expected average distance are also provided in Table 5.8.

5.2.4.3 Comparing the two approaches in case four

As we discussed above, when using Ripley's K-function approach on case two, commercial buildings and schools are anti-co-located in the distance range of 0 to 0.03. If we use the KNN distance approach, the results indicate that these two types of

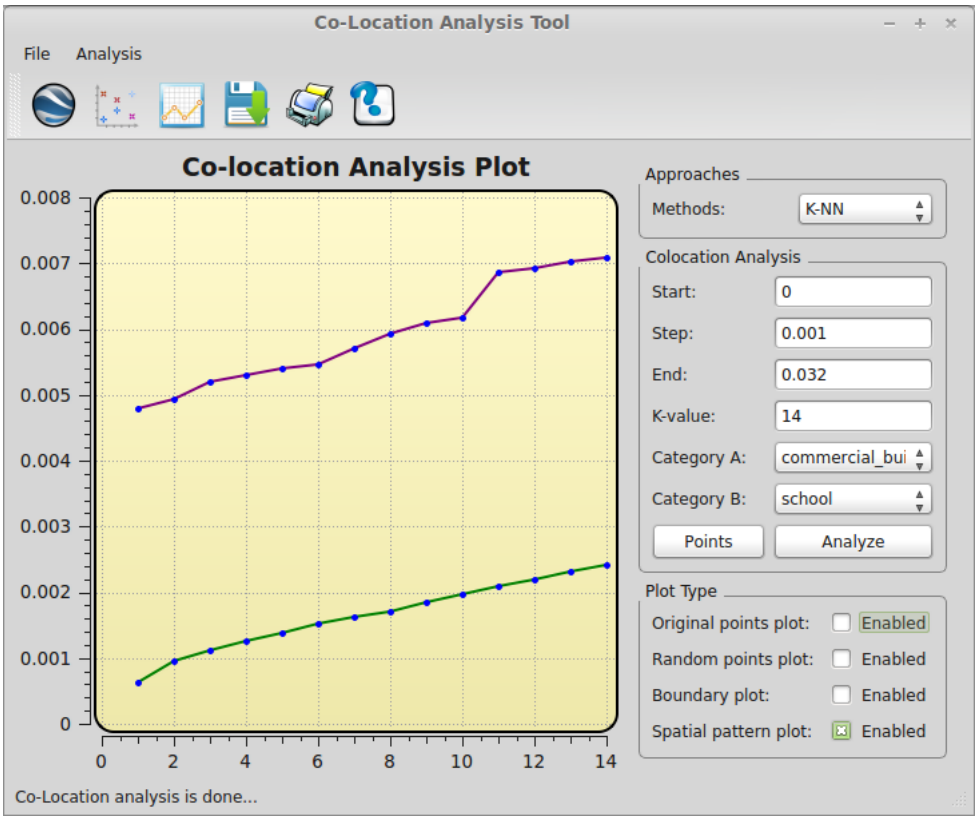

Figure 5.18: KNN distance approach case 4: commercial buildings vs schools

buildings are anti-co-located with each other from 0 to 0.03 distance range as well. Based on the experiment results, these two approaches agree with each other for case four. In order to compare the efficiency of these two approaches, we also list the average wall clock time of both approaches in Table 5.9.

Table 5.9: Average execution time of both approach of case four

Approach	Ripley's K-function	KNN distance
Execution time	1.692 seconds	1.126 seconds

From the table, we can see that Ripley's K-function approach is a little bit less efficient than KNN distance approach in case four.

5.2.5 Case five

The building types of collective house and garage are analyzed in case five. The distribution of the original input dataset of these two types is displayed in figure 5.19; collective houses are represented by red points and garages are represented by blue points. Figure 5.20 shows a scatter plot of garages and collective houses after they are randomly placed in the same region; we can observe that both of these two types of buildings are uniformly distributed in this figure.

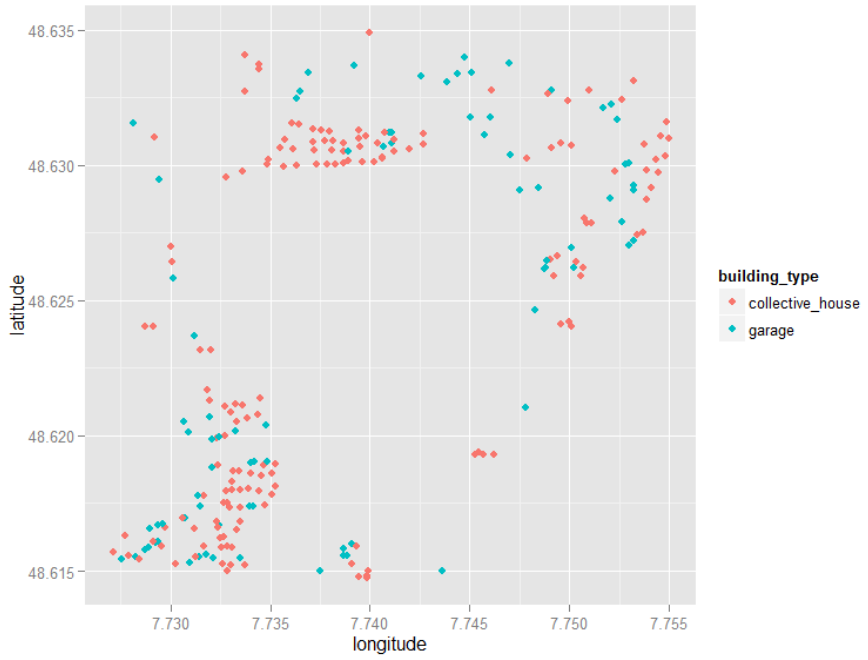

Figure 5.19: Distribution of garages and collective houses in original input dataset

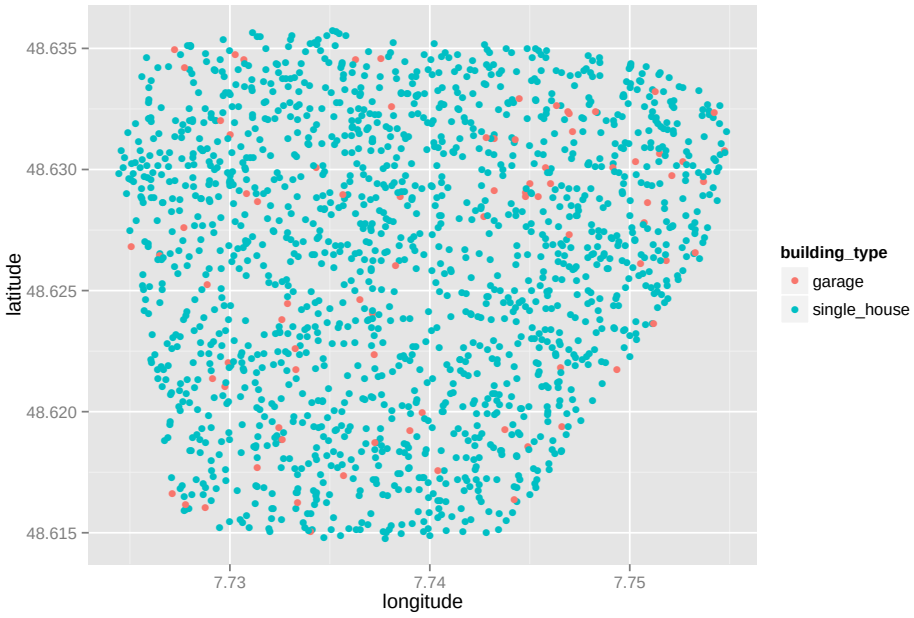

Figure 5.20: Random distribution of garages and collective houses

5.2.5.1 Using Ripley's K-function approach for case five

Garage and collective house are specified to be category A and category B respectively in this case. We first use Ripley's K-function to analyze spatial patterns for case two. Before using this method, the starting radius is specified to be 0, a distance increment is set to be 0.001, and the ending radius is specified to be 0.032. Figure 5.21 below shows the result of using Ripley's K-function approach.

From figure 5.21 below, we can see that the purple curve is above the green curve when radius ranges from 0 to 0.009, so the observed spatial patterns are co-located over this range. Then when radius ranges from 0.009 to 0.03, the purple curve is below the green one; these two curves overlap and reach 100% after the radius comes to 0.03. Thus, when distance ranges from 0 to 0.009, the distribution of garages and

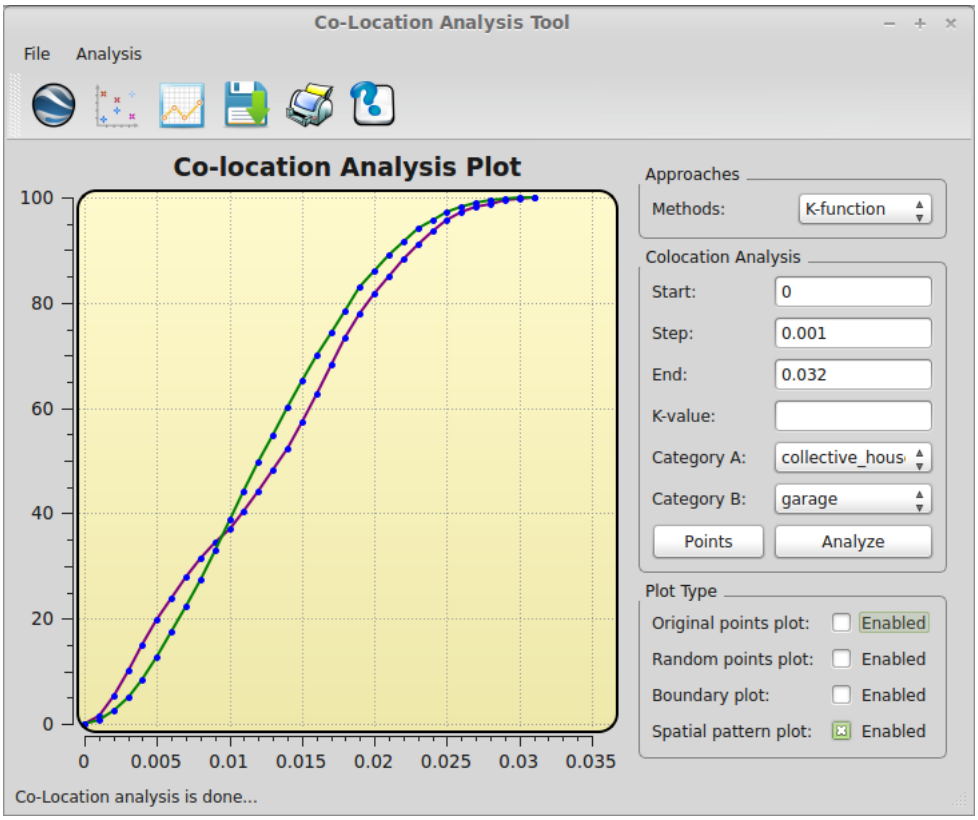

Figure 5.21: Ripley's K-function case 5: garges vs collective houses

collective houses are considered co-located, however, these two types of buildings are anti-co-located with each other over the range of 0.009 to 0.03.

5.2.5.2 Using KNN distance approach for case five

In case three, the k value is specified to be 81 became the total number of garages in original input dataset is 81. Figure 5.22 below shows the result of using the KNN distance approach on case five.

In figure 5.22, the purple curve represents the observed spatial pattern and the

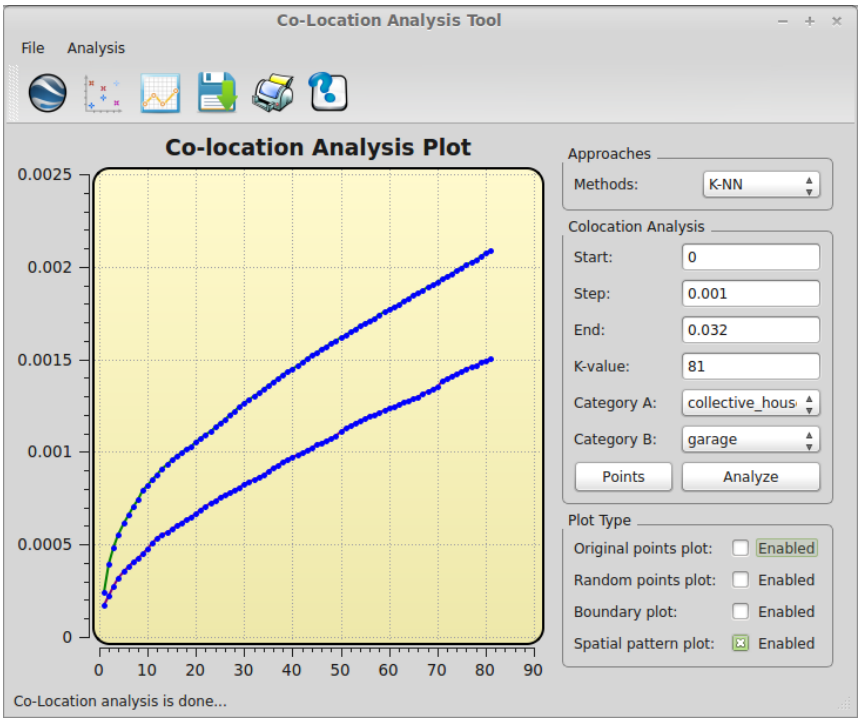

Figure 5.22: KNN distance approach case 5: collective houses vs garages

green curve represents the expected random spatial curve. Figure 5.22 shows the result of KNN distance approach. From the figure we can observe that the purple curve is below the green one, which indicates that collective houses are co-located with garages in the distance range of 0 to 0.0015. Table 5.10 shows selected k values and their corresponding observed, and expected average distances.

5.2.5.3 Comparing the two approaches in case five

As we discussed above, when using Ripley's K-function approach on case two, we get the conclusion that garages and collective houses are co-located in the distance range of 0 to 0.009, and are anti-co-located with each other for distances from 0.006 to 0.03.

Table 5.10: Statistical data of KNN distance approach for case five

K value	12	32	44	51	63	79
Observed average distance	0.000531	0.000850	0.001022	0.001129	0.001269	0.001482
Expected average distance	0.000876	0.001300	0.001520	0.001629	0.001812	0.002056

If we use the KNN distance approach, the results indicate that these two types of buildings are co-located with each other from 0 to 0.0015 distance range. Based on the experiment results, these two approaches partially agree with each other in case five. In order to compare the efficiency of these two approaches, we also list the execution time of each approach in Table 5.11.

Table 5.11: Average execution time of both approach of case five

Approach	Ripley's K-function	KNN distance
Execution time	2.012 seconds	32.064 seconds

From the table, we can see that Ripley's K-function approach is a little bit less efficient than KNN distance approach in case five.

5.2.6 Summary

Based on the experiment results described in previous sections, we found that in case one, two, and three, the Ripley's K-function approach, and the KNN distance approach agree to each other when the radius is small and disagree when the radius is large. However, the two approaches agree with each other in cases three and

four. Therefore, we can reach the conclusion that by using the building data set, the experiment results show that these two approaches agree to each other most of the time. This might be because the KNN distance approach only considers the shortest scales of variation, whereas Ripley's K-function approach provides an estimate of spatial dependence over a wider radii of scales based on all distances between objects in the area of interest. In addition, from the experiment using the KNN distance approach, we can observe that there is always a gap between the green curve and purple curve. This might because that there is a hole in the middle of the study region, but we still need to figure out the reasons which contributes to these kinds of results. Moreover, as we measure the wall clock time of the two approaches, it is easy to get the conclusion that the performance of Ripley's K-function approach is more computation efficient than the KNN distance approach.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Although there are some co-location analysis tools available for co-location analysis which are able to help users to do analysis to some extent, none of them can directly provide functions to do spatial data co-location analysis without any need for programming. Users need to be familiar with APIs or the languages provided by those tools to write code to do the analysis. Obviously, there is a need for an integrated environment to ease the co-location analysis process. The main contribution of this research is that we have created a user friendly tool for co-location analysis by incorporating two popular algorithms called Ripley's K-function and KNN distance approaches. Moreover, the KNN distance approach is used for co-location mining for the first time. It uses summaries of k-nearest neighbor distances of objects in the dataset to diagnose the presence of collocation patterns.

By using our tool, users can select the co-location analysis algorithm and set up experimental parameters based on their needs; then the tool is able to do the co-location analysis. Once the analysis is done, an experimental result will be automatically displayed. To ease future analysis work, the points coordinates and analysis results will be automatically dumped into a form of *.csv, which is readable any third party visualization tool for verification. In our implementation, we use the Qt framework to develop our tool, which makes our tool very portable for both Windows and Unix/Linux systems.

In addition, the tool is able to read and parse the *.kml file directly, and visualize those spatial datasets automatically without using any third-party tools or utilities to parse the file. Besides, this tool provide functions to visualize point objects and randomly generated points within the boundary polygon.

Finally, we use an artificial dataset and a real world building dataset to verify the correctness of our tool, and then make a comparison by using two clustering methods. The real-world building dataset has more than 2,000 building objects, our tool is able to process those spatial data analysis correctly.

6.2 Future Work

In future work, we are going to incorporate more co-location analysis algorithms into our tool, which can provide more choices for users to match their co-location analysis needs. For our current version of this tool, users need to manually specify two co-related objects to do analysis. We will provide that function in our next tool

release. We have also noticed that there is an intensive computing involved in the co-location data analysis for the KNN distance approach when the input spatial data has a large volume, which contains thousands of objects. In order to shorten analysis time, we are planning to use some parallel computation models, such as pThreads, Open MP, MPI, CUDA, etc. to parallelize our codes to make it time-efficient to process the co-location analysis on the KNN distance approach for a large volume of spatial data sets.

Bibliography

[1] Arcgis wiki. http://en.wikipedia.org/wiki/ArcGIS.

[2] Convex hull. http://en.wikipedia.org/wiki/Convex_hull.

[3] The convex hull of a planar point set. http://geomalgorithms.com/a10-_hull-1.html.

[4] Data mining wiki. http://en.wikipedia.org/wiki/Data_mining#cite_note-Fayyad-1.

[5] Esri's official website. http://www.esri.com/products.

[6] Google earth. http://www.google.com/earth/.

[7] Kml reference. https://developers.google.com/kml/documentation/kmlreference.

[8] List of spatial analysis software wiki. http://en.wikipedia.org/wiki/List_of_spatial_analysis_software.

[9] Minimal dom implementation. http://docs.python.org/2/library/xml.dom.minidom.html.

[10] Point in polygon. http://en.wikipedia.org/wiki/Point_in_polygon.

[11] spatstat package in r language. http://cran.r-project.org/web/packages/spatstat/index.html.

[12] R. Agrawal and R. Srikant. Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE, 1995.

[13] M.-S. Chen, J. S. Park, and P. S. Yu. Efficient data mining for path traversal patterns. *Knowledge and Data Engineering, IEEE Transactions on*, 10(2):209–221, 1998.

[14] F. K. Deeb and L. Niepel. A methodology for discovering spatial co-location patterns. In *Computer Systems and Applications, 2008. AICCSA 2008. IEEE/ACS International Conference on*, pages 134–141. IEEE, 2008.

[15] P. J. Diggle and A. G. Chetwynd. Second-order analysis of spatial clustering for inhomogeneous populations. *Biometrics*, pages 1155–1163, 1991.

[16] P. M. Dixon. Ripley's k function. *Encyclopedia of Environmetrics*, 2006.

[17] P. J. Dong Guozhu. *Sequence Data Mining*. Springer, 2007.

[18] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37, 1996.

[19] G. Greenman. Turning a map into a cake layer of information. *New York Times*, 2000.

[20] R. H. Güting. An introduction to spatial database systems. *The International Journal on Very Large Data Bases*, 3(4):357–399, 1994.

[21] K. M. Han Jiawei and P. Jian. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2006.

[22] Y. Huang, H. Xiong, S. Shekhar, and J. Pei. Mining confident co-location rules without a support threshold. In *Proceedings of the 2003 ACM Symposium on Applied Computing*, pages 497–501. ACM, 2003.

[23] Y. Huang, L. Zhang, and P. Yu. Can we apply projection based frequent pattern mining paradigm to spatial co-location mining? In *Advances in Knowledge Discovery and Data Mining*, pages 719–725. Springer, 2005.

[24] Y. Huang, P. Zhang, and C. Zhang. On the relationships between clustering and spatial co-location pattern mining. *International Journal on Artificial Intelligence Tools*, 17(01):55–70, 2008.

[25] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Principles of Data Mining and Knowledge Discovery*, pages 13–23. Springer, 2000.

[26] G. K. Kumar, P. Premchand, and T. V. Gopal. Mining of spatial co-location pattern from spatial datasets. *International Journal of Computer Applications*, 42(21):25–30, 2012.

[27] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 313–320. IEEE, 2001.

[28] H.-C. Lai, C.-T. Li, Y.-C. Lo, and S.-D. Lin. Exploiting and evaluating mapreduce for large-scale graph mining. In *Advances in Social Networks Analysis and Mining (ASONAM), 2012 IEEE/ACM International Conference on*, pages 434–441. IEEE, 2012.

[29] T.-R. Li, Y. Xu, D. Ruan, and W.-m. Pan. Sequential pattern mining. In *Intelligent Data Mining*, pages 103–122. Springer, 2005.

[30] X. Lin, C. Liu, Y. Zhang, and X. Zhou. Efficiently computing frequent tree-like topology patterns in a web environment. In *Technology of Object-oriented Languages, International Conference on*, pages 440–440. IEEE Computer Society, 1999.

[31] Y. Morimoto. Mining frequent neighboring class sets in spatial databases. In *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 353–358. ACM, 2001.

[32] A. Nanopoulos and Y. Manolopoulos. Finding generalized path patterns for web log data mining. In *Current Issues in Databases and Information Systems*, pages 215–228. Springer, 2000.

[33] A. Raorane and R. Kulkarni. Data mining techniques: A source for consumer behavior analysis. *arXiv preprint arXiv:1109.1202*, 2011.

[34] S. U. Rehman, A. U. Khan, and S. Fong. Graph mining: a survey of graph mining techniques. In *Digital Information Management (ICDIM), 2012 Seventh International Conference on*, pages 88–92. IEEE, 2012.

[35] B. D. Ripley. The second-order analysis of stationary point processes. *Journal of Applied Probability*, 1976:255–266, 1976.

[36] B. D. Ripley. Modelling spatial patterns. *Journal of the Royal Statistical Society. Series B (Methodological)*, 1977:172–212, 1977.

[37] J. F. Roddick and M. Spiliopoulou. A bibliography of temporal, spatial and spatio-temporal data mining research. *ACM SIGKDD Explorations Newsletter*, 1(1):34–38, 1999.

[38] Y. H. Shashi Shekhar, Pusheng Zhang. *Spatial Data Mining*. Springer, 2005.

[39] S. Shekhar and S. Chawla. *Spatial Databases: a Tour*, volume 2003. Prentice Hall Upper Saddle River, NJ, 2003.

[40] S. Shekhar, S. Chawla, S. Ravada, A. Fetterer, X. Liu, and C.-t. Lu. Spatial databases-accomplishments and research needs. *Knowledge and Data Engineering, IEEE Transactions on*, 11(1):45–55, 1999.

[41] S. Shekhar and Y. Huang. Discovering spatial co-location patterns: A summary of results. In *Advances in Spatial and Temporal Databases*, pages 236–256. Springer, 2001.

[42] M. Steinbach, P.-N. Tan, H. Xiong, and V. Kumar. Objective measures for association pattern analysis. *Contemporary Mathematics*, 443(2007):205, 2007.

[43] P. Stolorz, H. Nakamura, E. Mesrobian, R. R. Muntz, J. R. Santos, J. Yi, and K. Ng. Fast spatio-temporal data mining of large geophysical datasets. In *In Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 300–305. AAAI Press, 1995.

[44] D. Stoyan and A. Penttinen. Recent applications of point process methods in forestry statistics. *Statistical Science*, 15(1):61–78, 2000.

[45] K. Streib and J. W. Davis. Using ripley's k-function to improve graph-based clustering techniques. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2305–2312. IEEE, 2011.

[46] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[47] K. Wang and H. Liu. Discovering typical structures of documents: a road map approach. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 146–154. ACM, 1998.

[48] W. Wang, J. Yang, and R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proceedings of the 23rd Very Large Data Bases (VLDB) Conferences*, volume 97, pages 186–195, 1997.

[49] M. Worboys and M. Duckham. *GIS: A Computing Perspective, 2Nd Edition*. CRC Press, Inc., Boca Raton, FL, USA, 2004.

[50] Y. Xinran and K. Turgay. Super-sequence frequent pattern mining on sequential dataset. In *Big Data, 2013 IEEE International Conference on*, pages 52–59. IEEE, 2013.

[51] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 721–724. IEEE, 2002.

[52] J. S. Yoo and S. Shekhar. A joinless approach for mining spatial colocation patterns. *Knowledge and Data Engineering, IEEE Transactions on*, 18(10):1323–1337, 2006.