Student thesis series INES nr 327

Spatial analysis for the distribution of cells in tissue sections

Ruibin Xu

2014
Department of
Physical Geography and Ecosystems Science
Lund University
Sölvegatan 12
S-223 62 Lund
Sweden

Ruibin Xu (2014).
Spatial analysis for the distribution of cells in tissue sections
Master degree thesis, 30 credits in Geomatics
Department of Physical Geography and Ecosystems Science, Lund University

Level: Master of Science (MSc)

Course duration: *January* 2014 until *June* 2014

Spatial analysis for the distribution of cells in tissue sections

Ruibin Xu

Master thesis, 30 credits, in Geomatics

October 2014

Supervisor:

Lars Harrie, Lund University

Exam committee

Jonathan Seaquist, Lund University

Andreas Persson, Lund University

Department of Physical Geography and Ecosystem Science

Lund University

Abstract

Spatial analysis, playing an essential role in data mining, is applied in a considerable number of fields. It is because of its broad applicability that dealing with the interdisciplinary issues is becoming more prevalent. It aims at exploring the underlying patterns of the data.

In this project, we will employ the methodology that we utilize to tackle spatial problems to investigate how the cells distribute in the infected tissue sections and if there are clusters existing among the cells. The cells that are neighboring to the viruses are of interest. The data were provided by the Medetect Company in the form of 2-dimensional point data.

We firstly adopted two common spatial analysis methods, clustering methods and proximity methods. In addition, a method for constructing a 2-dimensional hull was developed in order to delineate the compartments in tissue sections. A binomial test was conducted to evaluate the results.

It is detectable that the clusters do exist among cells. The immune cells would accumulate around the viruses. We also found different patterns near and far away from viruses. This study implicates that the cells are interactive with each other and thus present the spatial patterns. However, our analyses are restricted in a planar circumstance instead of treating them in 3-dimensional space. For the further study, the spatial analysis could be carried out in three dimensions.

Keywords: spatial analysis, clustering methods, proximity methods, 2D hull

Acknowledgements

This project is a master thesis in the master program Geomatics at the department of Physical Geography and Ecosystem Science at Lund University. The study was carried out in cooperation with Medetect AB in Lund.

I would like to sincerely thank my supervisor Lars Harrie in the department of Physical Geography and Ecosystem Science at Lund University. His directive suggestions, guidance and impressive kindness are very treasurable.

My special thanks are also given to Jonas Erjefält and Caroline Sanden at Medetect AB, for providing me with this interesting project and the opportunity to deal with the spectacular data.

I would also like to appreciate my family and friends who are supporting and encouraging me all the time.

Table of contents

1. Introduction

1.1 Background

Spatial analysis has been widely used since it emerged in the 1960s (Maguire, 2005). During the first years, spatial analysis was mainly applied to focus on the distribution of points, lines and polygons. Then it started evolving to emphasize on the complex spatial systems. Nowadays the development of the computer science enables various formats for the geographical information, broadening the field of applications.

The purpose of the spatial analysis is to tackle the spatial problems through analyzing and exploiting the potential information among the data. Undoubtedly the information contains the location, distance, direction, topology and other spatial elements. In a more general way, the spatial analysis essentially discovers the underlying patterns in the data so that the patterns could be built for the future research.

Some methods in the spatial analysis are e.g. classification and neighborhood analysis. Classification, popular in the data mining, assembles the objects with similar characteristic and simplifies the structure of the data. Neighborhood analyses are solved using e.g. buffers or triangular data structures.

Cluster analysis, which is one branch of the classification, is concerned with grouping data as well as identify the underlying patterns of the spatial data. This technique is performed depending on the similarities between the elements without knowing the specific classes. Cormack (1971) and Gordon (1999) clarified the meaning of a cluster both from an internal facet (homogeneity) and from an external facet (separation) respectively. The clustering methods attempt to maximize the differences between the groups and minimize the with-in group variations.

Proximity analysis, as another analytical means of spatial analysis, describes the degree of closeness between objects. It is common to use the Euclidean distance as a metric to tell the vicinity. Besides, Delaunay triangulation is broadly applied in the proximity analysis currently (Fisher et al, 2007).

Spatial information always composes three characteristics: the spatial, temporal and attributes aspects. All of these could be used for clustering. This study deals with the position attribute, taking the spatial relationships into consideration; clustering based on position attribute is denoted as spatial clustering.

Spatial clustering is often applied in geographical analysis. But the methodologies in spatial analysis could also be applied in other context, such as astronomy, medicine and chemistry. In this study we apply spatial analysis methods on medical data, more specifically on finding spatial patterns in tissue sections.

The spatial distribution of cells could help and inspire medical researchers to make diagnoses. Basically, we would assume what kind of the physiological changes happen when certain types of cells clump. We would also doubt that what kind of cell is near the virus-infected cells. By studying the cell behavior, the medical people could give treatments to the patients. That is why we aspire to explore the patterns of cells.

1.2 Research problems

In this project, we will focus on the study on spatial patterns of cells in the tissue sections. Based on the latest technologies, with the microscopic analysis of a tissue sections from a creature, we are able to take “pictures” of tissues where cells and viruses are visible (figure 1.1). This allows us to research on how the cells are spatially distributed in the tissues.

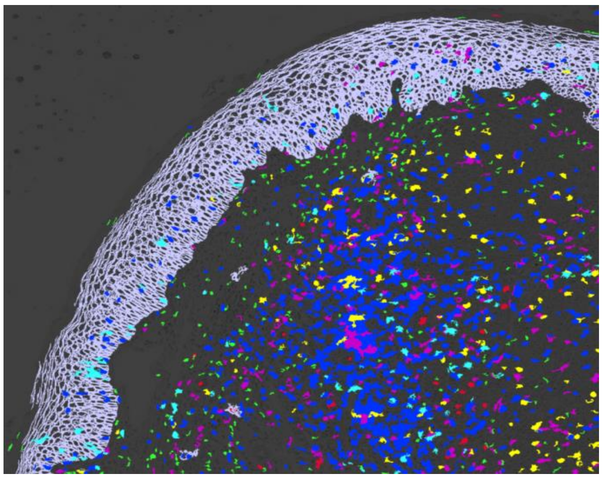


Figure 1.1 A picture for the tissue data. The colorful dots stand for cells.

Questions would be arisen as below:

1) Are there any clusters existing in certain types of cells?
2) What type of cells could be assumed to be neighbors to the virus-infected cells?
3) Do the different types of cells interact with each other?

1.3 Aim

The general aim of this study is to detect spatial patterns of cells and viruses in the tissue sections. The specific aims are:

1) To evaluate which methods are most suitable to detect the clusters of cells and viruses in tissue sections.
2) To evaluate proximity methods to define neighbors.
3) To develop a method for generating 2D hulls.
4) To test medical hypotheses based on methods found in aim 1-3.

1.4 Method

Figure 1.2 illustrates the outline of the whole study. A literature study is carried out to obtain the basic theoretical background of the clustering and proximity methods. Four main kinds of clustering approaches are evaluated, the partitioning methods, the hierarchical methods, density-based spatial clustering of applications with noises and the model-based methods. In addition, a proximity analysis, which is composed of closeness analysis and Delaunay triangulation based analysis, is conducted as well. The delineation of 2D hull is alternatively chosen for the further study. With the extracted data provided by the medical research partner, we conduct a couple of case studies. The outcomes will be compared and evaluated to validate the proposed medical hypotheses.

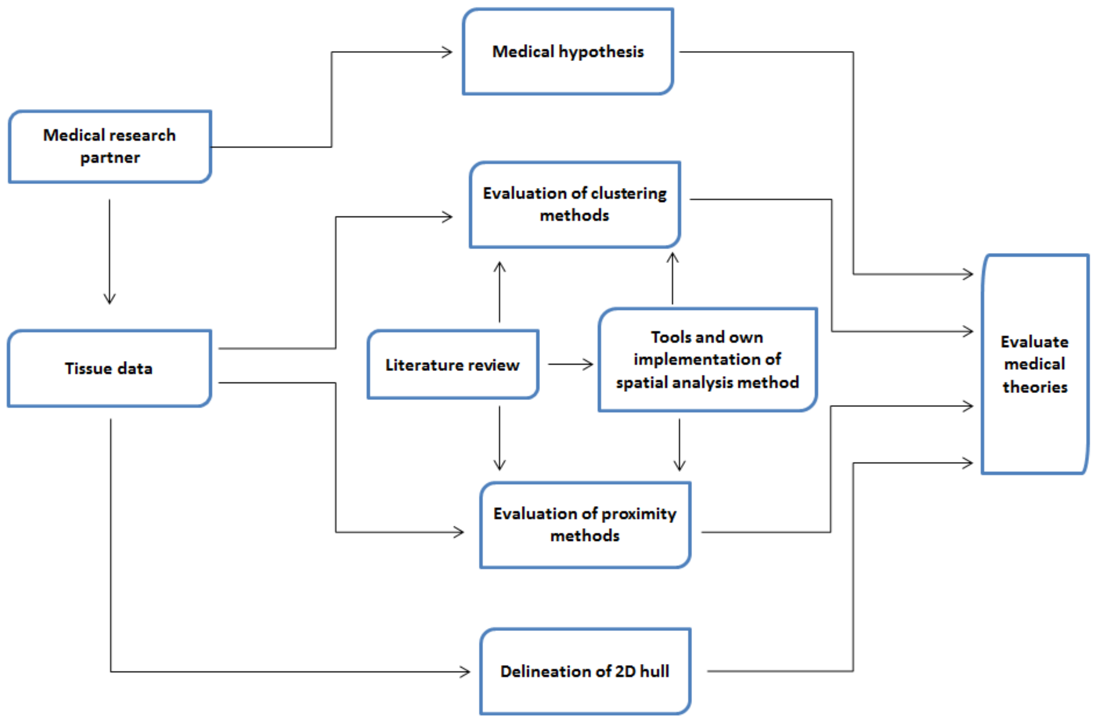

Medical hypothesis
Medical research partner
Evaluation of clustering methods
Tissue data
Literature review
Tools and own implementation of spatial analysis method
Evaluate medical theories
Evaluation of proximity methods
Delineation of 2D hull

Figure 1.2 Outlines for this project

1.5 Disposition

Chapter 2 describes the medical background of tissue sections. The basic medical theories are introduced here.

Chapter 3 presents the clustering methods. The theoretical background of the clustering approaches is included and the relevant practical work is introduced as well. An evaluation is made based on several common methods in the end of this chapter.

Chapter 4 introduces the proximity analysis. Both the theory and implementation of proximity methods are described. An evaluation is conducted by comparing the proposed proximity methods.

Chapter 5 introduces the algorithm of constructing a 2D hull. A brief description of other relevant algorithms is given in order to compare with our improved approach. Finally we carry out an implementation to evaluate this method.

Chapters 6-7 firstly verify the algorithm proposed in chapter 5 with the real data. A provision of the problems and hypotheses is offered by a biotechnology company named Medetect. According the previous evaluations of spatial analysis, a most proper method selected proceeds to verify the hypotheses.

Chapters 8-9 discuss what we have done in the whole project and conclude what we obtained from the case studies.

The report also includes some appendixes, including e.g. the algorithms we used in the methodology part.

2. Need for spatial analysis of tissue sections

In the past decades, spatial analyses have been carried out in different fields. And a variety of spatial analytical methods has been applied. A literature review with respect to the spatial analytical method is first introduced.

2.1 Related work

Korolev et al (2000) made a research about the spatial and temporal distribution of solutes in the carrot taproot. The spatial distribution of solutes was studied by comparing the concentration of solutes. They thought that the higher the concentration is, the more solutes exist. Simply, although the concentration cannot give a visual result about the solutes distribution, it is possible to tell the spatial distribution in a quantitative way.

Gottwald et al (1995) analyzed the spatial pattern of sharka disease in apricot and peach. A beta-binomial index of dispersion, which is associated with the disease incidence, was used to investigate whether spatial aggregation is present when plotting the data. In addition, geostatistical analyses and correlation analyses were carried out to confirm the scarcity of significant association among neighboring trees and figure out in which direction the adjacent trees were infected. By analyzing the disease patterns, the protection could be conducted in case of more trees infected.

Similar to the study above, an analysis of spatial patterns of virus-diseased tobacco plants was made by Madden et al (1987). The distribution of diseased plants was assessed using point pattern and spatial autocorrelation methods. The study area was divided through a modified Greig-Smith procedure. The whole analysis was implemented in a statistical way.

Lecoustre et al (1988) studied the spatial spread of African cassava mosaic virus. Geostatistics was employed to detect spatial dependence by measuring the variation of regionalized variables. Besides, kriging, which is a method of interpolation, was applied to estimate optimal and unbiased regionalized variables at unsampled places. It was also used to reconstruct the spatial patterns of spread of this disease.

A spatial proximity analysis was conducted by Roix et al (2003) to study the proximity of translocation-prone gene loci in human lymphomas. The average physical distances were measured between translocation partners. And the frequency of loci pairs was analyzed to characterize the spatial relationships of the translocation partners. This study proved that high-order spatial genome organization could partly determine the formation of specific translocation in human lymphomas.

2.2 Necessity of spatial analysis

The microscopic changes of distribution of cells happen all the time, especially when the cells get infected. These changes could assist in making diagnoses for patients. It is interesting and meaningful to explore if there are any spatial patterns existing during the life cycle of the cells. An example of a tissue section is given in figure 2.1.

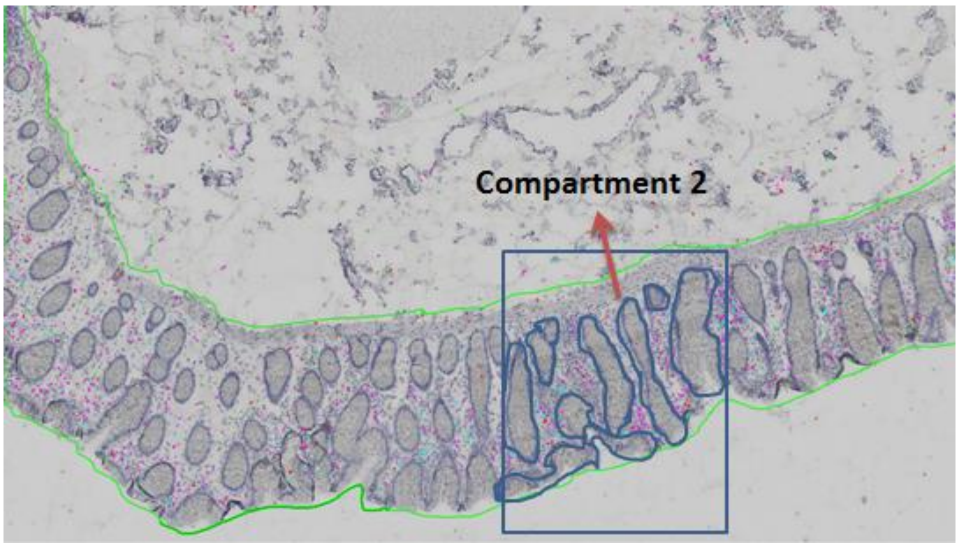


Figure 2.1 An example for the tissue section. The highlighted objects in the box represent the compartments and the dots stand for cells (provided by Medetect)

The coordinates of cells were extracted from a planar coordinate system which is symmetric to the image coordinate system. A plot is made below for presenting the cells in figure 2.1.

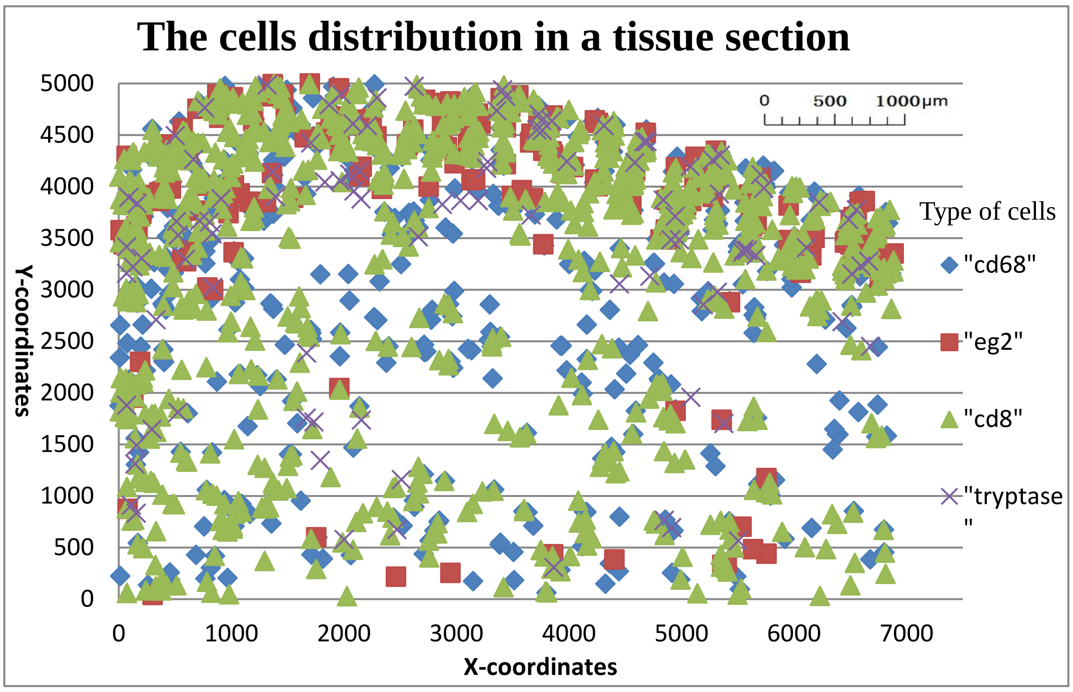


Figure 2.2 The cells distribution in a tissue section. There are four types of cells presented. The scale bar of the figure is only approximate.

It is not difficult to distinguish the dense areas from the sparse areas in figure 2.1. And some cells with the same color tend to be close to other certain type of cells. Hence, we deduce that there exist some spatial patterns in the tissue sections, which inspires us to look into the data with the spatial analysis approaches.

2.3 Selection of methods

The analytical methods mentioned in the section 2.1 could be characterized as the quantitative means to uncover the spatial patterns. However, it is difficult to be visualized in a 2D or 3D space. As for the research problems proposed in the section 1.2, only utilizing the quantitative approaches is not sufficient. To obtain visually good results, we would like to adopt clustering methods which could present the graphical clusters. In order to investigate the neighbors of viruses, the physical distance, which is capable of directly telling the proximity, is preferable to be considered. As we are not faced with scarcity of data, it is not necessary to use some geostatistical methods, such as kriging. Moreover we would also carry out the quantitative analyses. In addition, the highlighted area (see figure 2.1) motivates the use of 2-dimensional hull that delineates the border of point data.

3. Clustering methods

Clustering analysis has been becoming a popular multivariate statistical approach. It is quite widely employed in data mining. Basically, clustering analysis is performed according to the properties of the sample, organizing data into different groups based on the similarities and dissimilarities among the sample.

Spatial clustering, as a special case of clustering, is an important research topic in data mining. It identifies the spatial patterns by distinguishing the similarities and dissimilarities between observations. The process of spatial clustering could be regarded as an unsupervised method without knowing the details of clusters in advance.

The methods of clustering analysis could be broadly divided into four classes (Jain, 2010): hierarchical methods, partitioning methods, density-based methods and grid-based methods.

The selection of methods is related to the characteristics of the data. The first thing we need to consider is the properties of the data with which the particular methods and furthermore the criterion could be determined. Secondly, the dimensionality is another characteristic when it comes to the processing time and the quality of the results. The last but not the least aspect concerns the amount of the noise in data. Here the noise is defined as the object that is obviously far away from the other objects.

As mentioned above, the characteristics of the data play a vital role in analyzing the structure of the data. The data sometimes would not be static in realistic problems. The dynamic changes should be taken into consideration if necessary. Chung and Mcleod (2005) conducted a dynamic clustering analysis in news streams mining, exploring the meaningful patterns (e.g. news events and news topics).

Moreover, the integrity of the data is another problem we need to think about before the analysis. How to deal with the missing data will impact on the outcomes. In some situations, data are preferred to be removed in order to improve the analysis if few noises or the outliers do influence the whole grouping process. For example, in K-means method, the noise will change the shape of clusters.

3.1 Theory

3.1.1 Partitioning methods

A standard partitioning method is conducted as follows. Firstly, defining a specific number of clusters is the prerequisite before partitioning a dataset. Secondly, the partitioning algorithms organize the dataset into the defined number of clusters so that the sum of distance from each object to the cluster centers is minimized.

However, it seems manual to specify the number of clusters, and somehow it is probable to neglect the underlying patterns of the datasets. That the results tend to be spherically shaped clusters could probably be another weakness of partitioning methods.

Two common algorithms are introduced in the following sections, K-means and K-medoids. Each of them has its own way to represent clusters. K-mean employs the mean of the objects while the most centrally located object is used as the cluster center in K-medoids (Han et al., 2000).

3.1.1.1 K-means

Undoubtedly, the most well-known partitioning procedure is K-means. This technique partitions a set of data into a given number of groups through minimizing some specific criterions such as the sum of square error. The K-means method is relatively scalable and efficient in processing large datasets (Han et al., 2001).

As for the determination of the number of clusters, an “elbow method” is adopted. This method plots the sum of within-group errors against the number of clusters. The trend would tell us how the similarities between each object change. The higher the sum of errors is, the more dissimilar the objects are. Thus, the point where the trend tends to be steady should be the suitable number of clusters.

The mean values of each initialized cluster are calculated as the cluster centers. The squared error function is used as the criterion to iterate until the function does not change after the mergences and splits of clusters. The function could be formulated as:

$$E= \sum_{i=1}^{k} \sum_{x \in c_i} |x - m_i|^2$$

Where k is the specified number of clusters, x is the point in a certain cluster, and m_i is the mean value of the cluster c_i.

The mean value is used in K-means method such that the influence arose from the noise and outliers cannot be ignored. However, this algorithm works quite conveniently only with the numerical attributes.

3.1.1.2 K-medoids

Different from the K-means method, K-medoids uses the most centrally located object as the cluster center, making it less sensitive to the noise and outliers. No limitation on attribute types of the dataset is another advantage of K-medoids. However, finding the cluster centers is computationally complex. It is only effective for small datasets.

The algorithm of K-medoids could be summarized as:

i. Determine the number of clusters
ii. Randomly choose the center of each cluster
iii. Repeat finding and assigning the nearest points to each center and update the clusters
iv. Iterate until no replacement could be found

Like the K-means method, a squared error function is used as the criterion for assessing the clustering process. It differs from K-means method in that the cluster center is not fixable after each iteration. In this sense, it is witnessed that the change of the centers could contribute to the decrease of the criterion, which means the improvement is made through this replacement.

3.1.1.3 Other partitioning methods

To deal with the large dataset, Kaufman and Rousseeuw developed a method called CLARA (Clustering Large Applications) (Kaufman and Rousseeuw, 1990). This method iterates to take a portion of data as representative into consideration instead of processing the whole dataset directly.

3.1.1.4 Applications of partitioning methods

Boley et al (1999) proposed two improved methods based on the partitioning clustering for the retrieval of information on the web. These two methods group the most frequently occurring objects instead of counting on the distance matrix or similarities. They tried to categorize the web information for the later-on retrieval.

Another application of partitioning methods is to detect the locations where the earthquakes could occur (Scitovski and Scitovski, 2013). It is complicated to find out the activity centers of the seismic areas. A combination of the well-known DIRECT algorithm and K-means is raised to find either a globally optimal partition or a locally optimal one. Finally, an optimal partition with 3 and 13 clusters is determined. The resulting graphs are shown below.

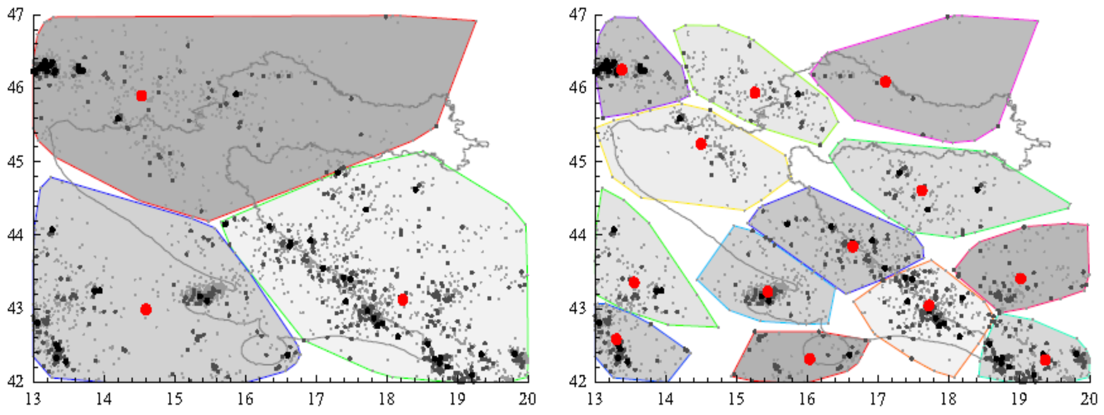


Figure 3.1 An optimal partition with 3 (left) and 13 (right) clusters (red dots) (Scitovski and Scitovski, 2013, Section4.1, 2013, pp.128)

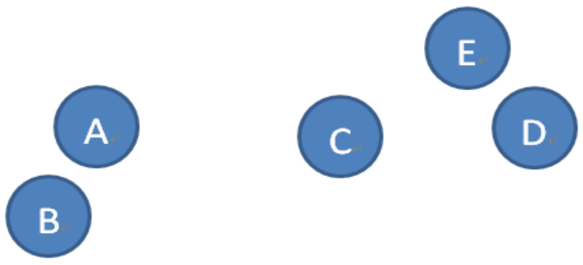
3.1.2 Hierarchical methods

As for the hierarchical methods, it is not necessary to know what number of clusters there are in the sample. Instead this approach partitions the sample iteratively. The hierarchical methods could be divided into two types of methods. One is the agglomerative method, which groups each individual by some certain criterion. And the other is divisive method, which performs an opposite way as the agglomerative one does, dividing a cluster into each individual (Everitt et al., 2011).

Depending on the sequence of hierarchy partitioning, hierarchical clustering has two subdivisions, agglomerative method and divisive method. The general idea of agglomerative method is: to group each individual into classes, calculate the distances between classes and merge the classes with the shortest distances. Each time after the mergence, the distances should be updated between the new classes. This procedure would be ended until all the samples are classified into one group.

The illustration of divisive method is: to group all the objects into one large cluster, and then successively partition the cluster until that each object is classified into one cluster or that the terminative requirements are satisfied.

A dendrogram visually represents how the agglomerative and divisive methods work. For example, the use of a dendrogram in the agglomerative methods is given in figure 3.2. We first assign each observation to individual clusters. The individual clusters on the left side are referred to as leaf nodes. Clusters are formed by joining the leaf nodes or existing clusters. The whole process will not be ended until all the observations are grouped into one cluster. The divisive method operates in the opposite way (figure 3.2).



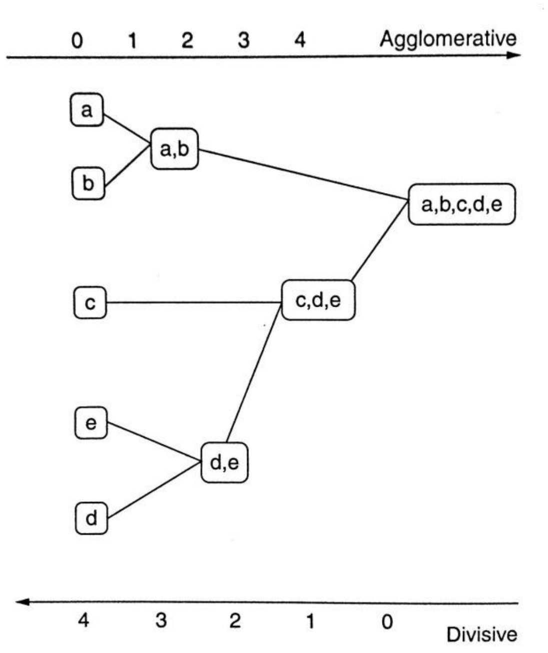


Figure 3.2 The distribution of five observations and an example of the hierarchical tree structure. The arrows in the lower figure instruct the two subdivisions in the hierarchical methods.

It is noteworthy that both the agglomerative method and divisive method are irrevocable (Everitt, 2011). That is, when the fusions have been done, it is impossible to divide the fusions to the former status, and vice the versa. Evidently, the disadvantage of the hierarchical methods lies in that if either the fusions or the divisions are done, this procedure cannot be undone.

3.1.2.1 Hierarchical methods based on different distance metrics

Within the agglomerative and divisive methods, three subdivisions of the methods are presented dependent on the distance between the clusters (Everitt et al., 2011). (a) Single linkage, also called the nearest neighbor clustering, calculates the shortest distance between two points in two different clusters. Here the distance means the similarities, that is, the shorter the distance, the more similar the two points are. This method prevails in dealing with the non-ellipsoidal structure and is sensitive to the noises and outliers. The equation is formulated as below:

$$d_{\min}(c_i,c_j) = min_{p \in c_i, p' \in c_j} |p - p'| \tag{1}$$

Where c_i, c_j are clusters, and $|p - p'|$ is the distance between object p and object p'. Distance could be e.g. Euclidean distance.

(b) Complete linkage, which is also called the furthest neighbor clustering, is opposite to the single linkage. The longest distance between two different clusters is considered. This approach is less sensitive to the noises and tends to find out the compact classifications. The distance formulation is given below:

$$d_{\min}(c_i,c_j) = max_{p \in c_i, p' \in c_j} |p - p'| \tag{2}$$

(c) Centroid linkage is first to calculate the mean of each cluster. The distance between each two clusters is the differences between the mean of the corresponding clusters. The equation is:

$$d_{\min}(c_i,c_j) = |m_i - m_j| \tag{3}$$

Where m_i, m_j are the means of c_i, c_j respectively.

3.1.2.2 Ward's hierarchical method

The Ward's hierarchical clustering is an agglomerative method. It is similar to the linkage methods mentioned above, starting with individual clusters and ending with one single cluster. However, the difference lies in that the linkage methods are based on the distances while the Ward's method groups clusters by calculating the total within-group sum of squares (SSE) (Ferreira and Hitchcock, 2009). The formulation can be defined as:

$$\text{SSE}=\sum_{i=1}^{K}\sum_{j=1}^{n_i}(y_{ij}-\overline{y_\iota})*(y_{ij}-\overline{y_\iota}) \qquad (4)$$

Where y_{ij} is the j^{th} object in the i^{th} cluster, $\overline{y_\iota}$ is the mean value of the i^{th} cluster and n_i is the number of object in the i^{th} cluster.

3.1.2.3 Other hierarchical methods

Other hierarchical methods like BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) and CURE (Clustering using Representatives) are useful for handling large databases. BIRCH is able to cluster data of high dimensionalities dynamically and generates a high quality result even with a single scan of the data (Zhang et al., 1996). CURE is sophisticated in identifying outliers and non-spherical clusters (Guha et al., 1998).

3.1.2.4 Applications of hierarchical methods

A new hierarchical clustering strategy is proposed to detect the interactivity of the proteomic data (Arnau and Marin, 2003). They attempt to see the strength of interaction which could be evaluated by the frequency of grouping any two elements. Therefore, this method defines a primary distance for each pair of two elements. The elements with similar primary distance will be clustered.

Another application of hierarchical methods is aiming at constructing an Audio Event Detector system that has s hierarchical structure (Pellegrini et al., 2009). They identify the similarities and dissimilarities between sound samples. The clustering is carried out to group the similar sound samples.

3.1.3 Density-based methods

Density-based methods commence discovering the underlying patterns of data by grouping observations through specific density functions (Gaonkar and Sawant, 2013). The dense region would be clustered. It is inevitable to get the spherically shaped clusters if using the distance-based procedures. Nevertheless, the density-based methods could detect clusters with arbitrary shapes.

3.1.3.1 DBSCAN

A common density-based method is called DBSCAN (Density-based Spatial Clustering of Application with Noise) (Ester et al., 1996). The clusters are identified according to two input parameters in this method:

- Minimum number of points (*MinPts*).
- Neighborhood distance (*Eps* –neighborhood): the neighborhood within a radius *Eps* of an object.

Additionally, there are several basic definitions described below.

Definition 1 *Core object*

A core object is the object that has more than *MinPts* of objects within its *Eps* -neighborhood (Han et al., 2000). If object *p* has a minimum number (*MinPts*) of objects in the radius of *Eps, p* is called a core object.

Definition 2 *Border object*

If the number of objects is smaller than *MinPts* in the *Eps* of object *p*, and *p* is within the *Eps* of other core objects, the object *p* is called a border object. It is noteworthy that the noise here differs from those in other clustering methods. Simply, the noise mention especially means the object that is not grouped into any clusters.

Definition 3 *Noise*

If the number of objects is smaller than *MinPts* in the *Eps* of object *p*, and *p* is not within the *Eps* of any other core objects, the object *p* is delineated as noise.

Definition 4 *Directly density-reachable*

If object *p* is within the *Eps* of object *q*, and object *q* is a core object, object *p* is directly density-reachable from object *q* (see figure 3.3).

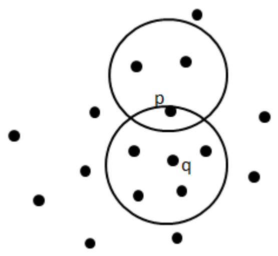


Figure 3.3 Object *p* is directly density-reachable from object *q*.

Definition 5 *Density-reachable*

If there is a chain of objects p_1, p_2… p_n, p_1=*q*, p_n=*p* such that p_{i+1} is directly density-reachable from p_i, object *p* is density-reachable from object *q* with respect to *Eps* and *MinPts* (see figure 3.4).

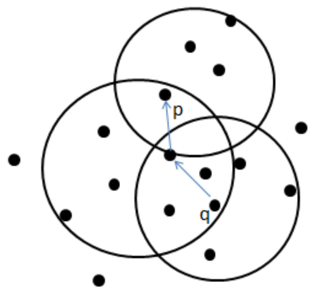


Figure 3.4 Object *p* is density-reachable from object *q* and not vice versa.

The general procedure of DBSCAN could be described as below:

1. The DBSCAN checks each object in the dataset. If the *Eps-neighborhood* of an object *p* has more than *MinPts* objects, a new cluster is created with *p* as the core object.
2. The procedure repeats looking for the objects which are directly density-reachable from these core objects. This process concerns the combination of density-reachable objects.
3. The iteration will not be ended until no new objects are added to the current clusters.

A couple of rules are needed to be followed in order to form the clusters:

1) An object is only within a cluster on the condition that it is inside the *Eps* -neighborhood of some core object.
2) If there is one core object *a* within the *Eps* -neighborhood of another core object *b*, *a* could be grouped into *b*.
3) If a non-core object *p* is within the *Eps* -neighborhood of several core objects at the same time, *p* must belong to at least one of these core objects.

DBSCAN is efficient enough to deal with the large datasets. However, it is not capable of finding out any meaningful clusters with varied densities (Gaonkar and Sawant,2013).

The parameters can be set by prior knowledge or by an investigation of the data. *Eps* can be determined by plotting a sorted *k-distance* matrix which is calculated between each object and its k^{th} nearest neighbor. An experiment with different values of *k* could result in the adoption of a range of values for *Eps*. And it is general to choose *k+1* as the value of *MinPts* because each cluster should at least have *k+1* objects.

3.1.3.2 Other density-based methods

Besides these mentioned clustering methods, there are several other approaches presented. For instance, OPTICS (Ordering Points to Identify the Clustering Structure), a density-based method, takes advantage in finding out the outliers and discovering clusters of arbitrary shapes.

VDBSCAN (Varied Density-based Spatial Clustering of Application with Noise) is derived from DBSCAN. It differs in that several values of *Eps* are selected for different densities. With different *Eps*, this method clusters objects with respect to the corresponding densities (Liu et al., 2007).

3.1.3.3 Applications of density-based methods

A new version of DBSCAN called r-ray DBSCAN is applied in the r-ray astrophysical images (Tramacere and Vecchio, 2012). They have a trial to use the statistical characterization of DBSCAN to identify the regions with better detection efficiency and lower inaccuracy ratio. A wide range of input parameters is investigated. The searching radius (*Eps*) is correlated to the point spread function (PSF). And the minimum number of objects in each cluster is determined by the Poissonian statistics.

The application of DBSCAN clustering in emergency plan compilation was raised by Wu et al (2012) in order to detect the arbitrary shape of categories. A normalization of plans is conducted to transform to feature vector. Then the threshold of input parameters is decided through a least square fit where a similarity curve is plotted.

3.1.4 Grid-based methods

To facilitate the efficiency of processing, grid-based methods were put forward. These methods utilize a grid-based structure to quantize the data into a finite number of cells.

The fast processing mechanism becomes the main advantage of the grid-based methods, which is dependent on the size of the grid instead of the number of data objects (Liao et al., 2004).

The whole process of the grid-based algorithms could be generally illustrated as followed:

1. Partition the data into a grid.
2. Cast the data objects to the corresponding cells and calculate the density of each cell at the same time.
3. A threshold is determined and the cells whose density is lower than the threshold should be deleted.
4. Group the dense cells into clusters.

A couple of grid-based approaches have been developed with. STING (Statistical Information Grid) constructs a hierarchical structure with different levels of cells to statistical information regarding the attributes of each cell. WaveCluster makes the transformation of the initialized feature space by a *wavelet transformation*.

CLIQUE (Clustering high-dimensional space) is based on the density-based and grid-based clustering. Its advantage lies in finding clusters in high dimensional data which are too sparse to form clusters (Han et al., 2000).

3.1.4.1 Applications of grid-based methods

A combined approach of grid mapping and DBSCAN is utilized in the stream data clustering (Quan et al., 2011). There are three requirements that should be met when clustering stream data: (1) compressed data capacity. (2) improvement of data processing efficiency. (3) easy detection of noises. In this sense, a grid data structure is employed to facilitate the processing efficiency. Meanwhile, it is appropriate to use

DBSCAN to find outliers.

The grid-based clustering could also be applied in the mobile wireless sensor network. A robust clustering proposed by Ali and Madani (2009) uses the location of a sensor node as the cluster head in the initial step. The location information could be readily obtained from the grids. Due to the mobility of the sensor, the sensor field is logically divided into zone. Each sensor node would send information to its neighboring cluster head and each cluster head would contact with its close cluster heads as well. Hence the aggregation is made to form clusters.

3.1.5 Model-based methods

A model-based method could not only locate the clusters by constructing the density-based distribution of the datasets, but also identify the number of clusters based on the statistics. Noise and outlier are both taken into consideration as well such that the model-based method generates a robust clustering method.

The selection of models is based on some criteria. Bayes Information Criterion (BIC) is a criterion for choosing a model from a finite set of models. It is partially based on the likelihood function that is parameterized by a statistical model. And Expectation Maximization (EM) is an algorithm for maximizing a likelihood function when there are some unobserved variables in a model. Both BIC and EM play an important role in selecting a model.

3.1.5.1 Mixture models

DBCLASD (Distribution-Based Clustering Algorithm for Clustering Large Spatial Datasets) (Xu et al., 1998) integrates model-based, density-based and grid-based methods. Basically, the identification of clusters in this algorithm is based on the density. It is distinguishable that the distance between the nearest neighboring points within some area is shorter than that between points outside the area. A probability distribution of the nearest neighboring distances accounts for the characteristics of the clusters formed by the points and identify if the nearest points are inside the cluster as well.

3.1.5.3 Applications of model-based methods

A model-based clustering is employed to visualize the navigation patterns on a website (Cadez et al., 2000). The clusters of site users are formed according to the similar navigation paths. In each cluster, user behavior is displayed by sequence with the first-order Markov models.

Handcock et al.(2006) proposed a latent position cluster model to deal with the social networks. In this model, the persons' latent positions correspond to clusters. The probability of a relationship is dependent on the Euclidean distances between clusters.

3.1.6 Aspects of clustering

3.1.6.1 High dimensionality

The traditional clustering algorithms succeed in solving the problems with low dimensionality. However, due to the complexity of some realistic data, the existing algorithms are ineffective, especially for the data with high dimensionality or large capacity. There are two main problems that the traditional algorithms are faced up with:

1) The probability of the existence of the clusters is nearly 0 because of a large number of irrelative attributes in the high-dimensional datasets.
2) Comparing with the low-dimensional data, the distribution of the high-dimensional data is quite sparse. It is common that the distance between

objects is almost equal, which makes it difficult to construct clusters because the traditional methods are performed based on the calculation of distance.

3.1.6.2 Spatial indexation

Clustering methods require that spatial search, such as finding the closest point in a large point set, is fast. To enable fast search, spatial indexation is introduced. Example of spatial indexation for point data are KD-tree, R*-tree (Beckmann et al., 1990) and X-tree (Berchtold et al., 1996). All clustering methods must utilize spatial indexation to be efficient if large datasets are used.

3.2 Implementations of clustering methods in R

3.2.1 Introduction to R

R is an open source environment for statistical programming (GPL license: GNU operating system). S language, as the root of R, was developed by John Chambers and his partners (Becker et al., 1988, Chambers and Hastie, 1992, Chambers, 1998) at Bell Laboratories. And R is software integrating the statistical analysis and graphical display as well. R is widely accessible to the public and can be run under the UNIX, Windows or Macintosh operating systems. A very instructive help system is embedded in R.

R has a large toolbox for handling data analysis. Additional modules are widespread to deal with particular tasks. Because of its extensibility to incorporate other packages, analysts could enable their own coding for specific purposes.

3.2.2 Clustering in R

R also has a powerful toolbox for spatial clustering and was therefore selected for our project. Its capability of handling data and displaying graphics would generate a satisfied outcome in both statistical and graphical ways. Below we describe the following cluster methods in R: K-means, Ward's hierarchical clustering, DBSCAN and model-based clustering. The codes to run these methods are provided in Appendix A.1- A.4.

3.2.2.1 K-means

In K-means algorithm, you have to state the number of clusters. To estimate how many numbers of clusters there are, it is recommendable to use a plot of within-group sum of squares for the extracted clusters. Then we choose an "elbow" where we are able to detect the great difference of the slope. After deciding the number of clusters, the K-means method is then performed.

To run K-means in R, we apply the function *kmeans* that is embedded in package *stats*. There are two input parameters, one is the data and the other one is the determined number of clusters. The number of clusters is obtained at the first step, according to which the function randomly set the initial cluster centers.

3.2.2.2 Hierarchical clustering

Ward's hierarchical method is an agglomerative procedure. The package used here is still *stats*. Firstly, it is necessary to calculate a Euclidean distance matrix with the function *dist*, which constitutes a dissimilarity matrix. Then the function *hclust* initially assigns each object to an individual cluster. And then the algorithm iterates to join the most two similar clusters into new clusters until there is a single cluster. The input parameters for *hclust* are the created dissimilarity matrix and the type of method.

3.2.2.3 DBSCAN

To perform the DBSCAN algorithm, an additional package called *fpc* is installed (*CRAN*). This package contains various methods for clustering and clustering validation. The function *dbscan* is used with two parameters, *Eps* and *MinPts* which are determined by a plot of sorted distances from the k^{th} nearest neighbors.

3.2.2.4 Model-based clustering

The function which is used to perform model-based method is *Mclust*. It is included in the package *mclust*. The function could directly experiment with the input data without determining other arguments. The number of clusters is determined and the best fit is selected automatically. Otherwise, analysts could also specify their own models that are provided in the function.

3.3 Case study I: evaluation of clustering methods

3.3.1 Introduction

In this section, we will implement the clustering methods and present the results graphically. The evaluation of clustering methods for tissue data (see Chapter 2) is given in the end. Based on this evaluation, we would on the one hand discuss which method is comparably suitable for the further studies. And on the other hand, we would investigate whether there are clusters existing in certain types of cells, which is the first question we raised in the section 1.2.

3.3.2 Data

The data to be evaluated, which mainly focus on human cells in specific tissues, is provided by the Medetect Company. The data was collected from the tissue of tonsil. Different sorts of cells and viruses are extracted and labelled. Meanwhile, an analyst assigned the coordinates for each cell and virus. A rough description of the distribution for these objects is presented as a plot (figure 3.5).

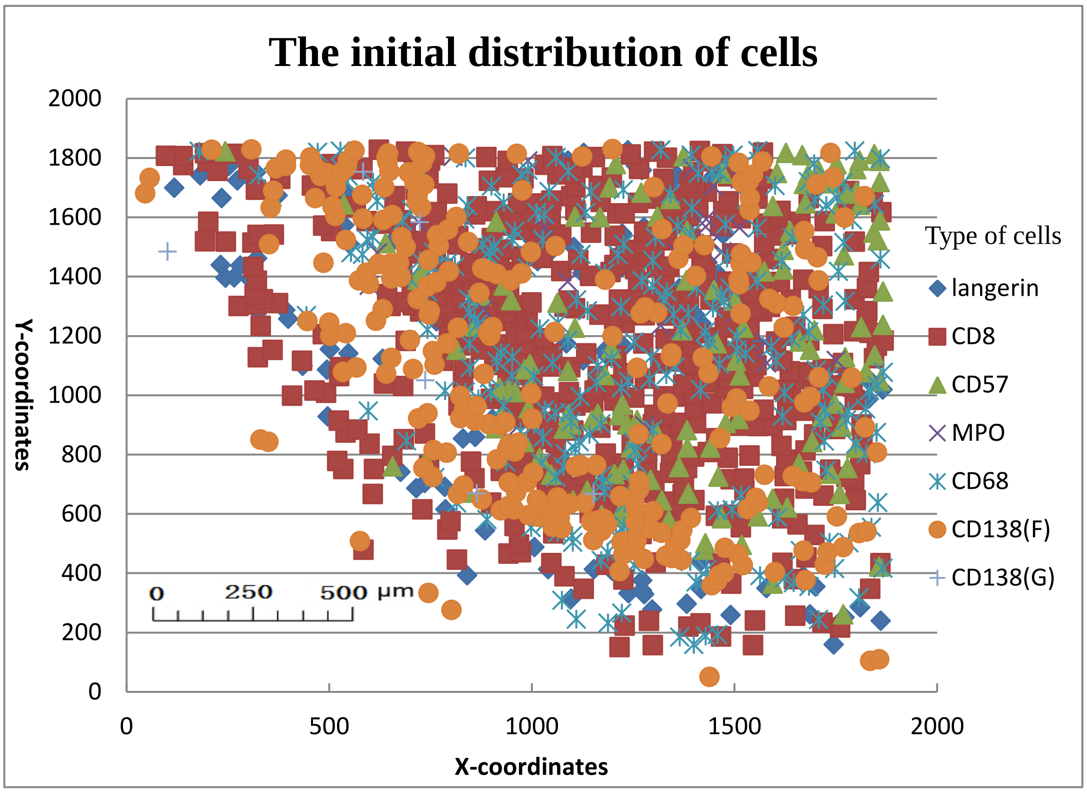


Figure 3.5 An initial distribution of the immune cells and viruses, where langerin, CD8, CD57, MPO, CD68 and CD138(F) are all immune cells while CD138(G) is a virus. The scale bar of the figure is only approximate.

3.3.3 Method

Our analyses are generally divided into two parts, one is to test the data with a predefined number of clusters and the other is based on the statistics of the data. All computations are made in R (see Section 3.2.2 and Appendix A.1-A.4). The evaluations are then carried out either in an intuitive way or according to accuracy assessments.

In part 1, we choose 25 as the number of clusters. Algorithms except the model-based methods and the DBSCAN were performed using the 25 clusters. In part 2, the determination of number of clusters is made in a statistical way, based on which we experiment each approach with the data.

In K-means clustering, firstly we decided the number of clusters by calculating the within-group sum of squares. The point where the variation within groups is minimized is selected as the number of clusters. Finally, the *kmeans* function was used to perform this algorithm.

As for the hierarchical methods, we chose Ward's hierarchical clustering as the analytic approach. A dissimilarity matrix is first initialized by calculating the distances between each object. Each object is taken as an individual cluster. As the algorithm precedes iteratively, the most two similar objects are joined until one single cluster exists. At each aggregation, the distance is recomputed and updated.

To perform the DBSCAN algorithm, two parameters have to be decided in advance, *Eps* and *MinPts*. However, there is not a general way to determine the values of the parameters. A *k-dist* plot could be a heuristic that the k^{th} nearest neighbors are graphically presented. A distance matrix is created in Matlab (See Appendix A.6). The Euclidean distance is calculated from each point to other remaining points. We sort these distances in an ascending order. The plot is drawn with the k^{th} nearest distances. A curve is simulated due to the shape of the point set. The levels of density could be distinguished from the plot, thus identifying whether the clusters have varied densities. The *y* value of the point where a sharp change occurs is the optimal value for *Eps* (Gaonkar and Sawant, 2013). Note that although the value of *Eps* relies on *k*, it does not change dramatically as *k* increases (Liu et al., 2007). We would test the data with an initial value of k, and we take k+1 as the minimum number of objects in a cluster. The DBSCAN algorithm is executed with each pair of *Eps* and *MinPts*. Afterwards, we will adopt the most suitable result for this clustering.

When it comes to the model-based method, *Mclust* package provides us with a number of models. Based on the specific criterions and the characteristics of data, the best fit model is selected. Then the number of clusters is determined with the largest BIC value.

3.3.4 Result

3.3.4.1 K-means with a predefined number of clusters

Cluster analyses were carried out with a determined number of clusters. The result from the partitioning methods is presented below.

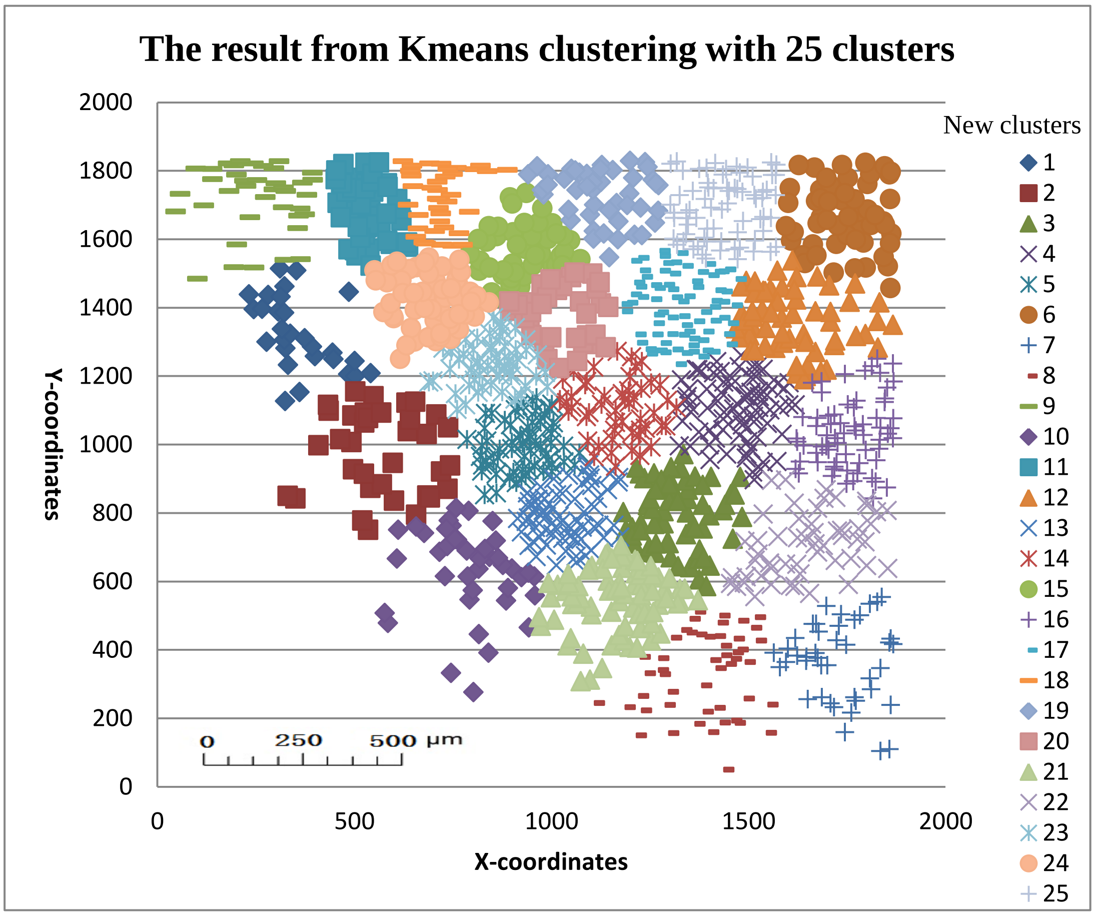


Figure 3.6 Result from K-means method using 25 clusters. A series of number is given to the new clusters. The scale bar of the figure is only approximate.

3.3.4.2 Ward's hierarchical method using a predefined number of cluster

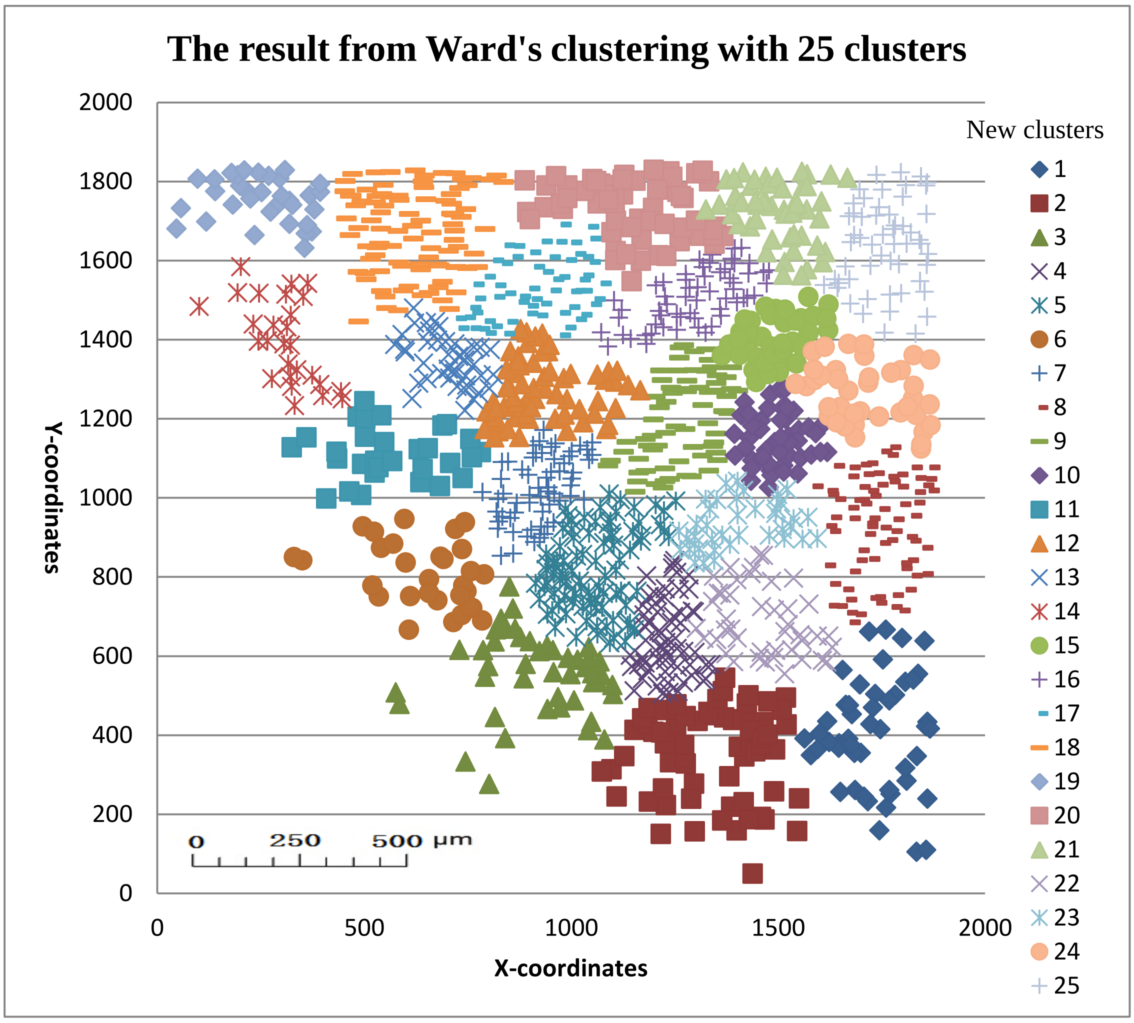


Figure 3.7 Result from the Ward's hierarchical method using 25 clusters. The scale bar of the figure is only approximate.

3.3.4.3 K-means

A statistical method was used to determine the number of clusters. The plot of within-group variations was shown below (figure 3.8). 7 is adopted as the number of clusters.

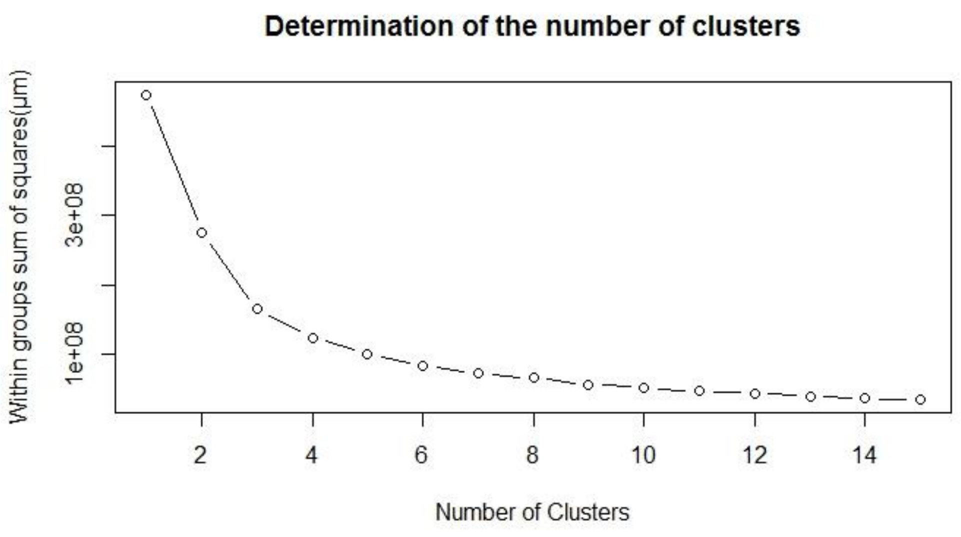


Figure 3.8 The plot of within-group sum of squares

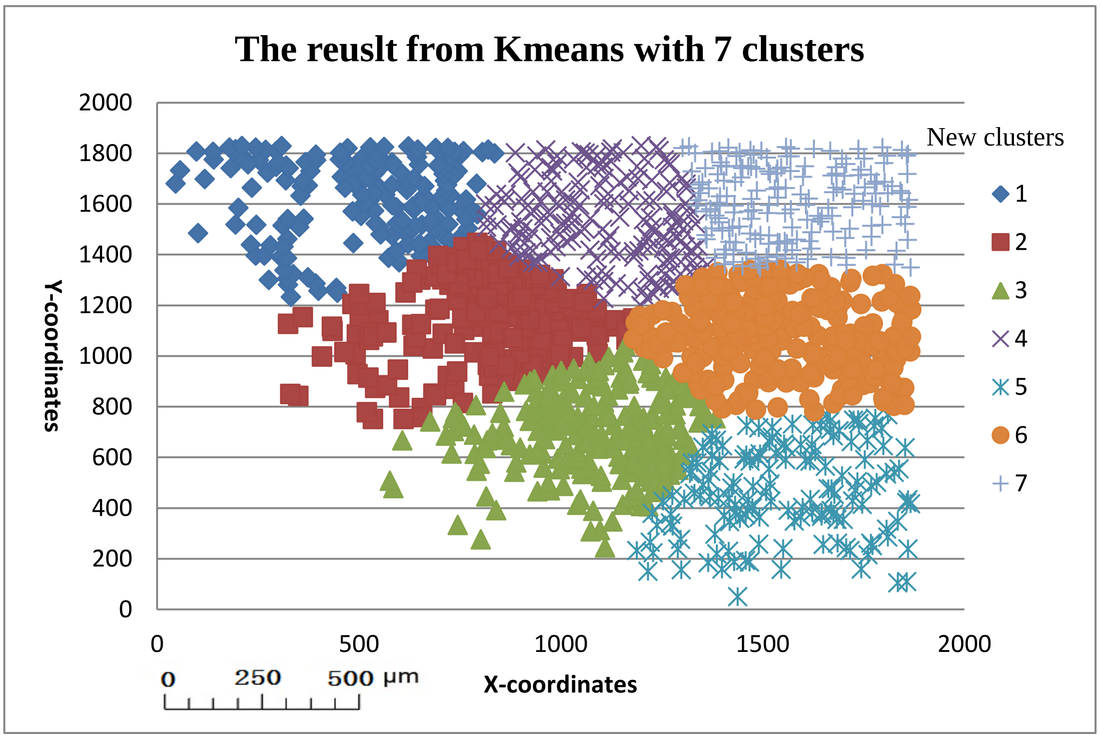


Figure 3.9 Result from K-means method using 7 clusters. A series of number is given to the new clusters. The scale bar of the figure is only approximate.

3.3.4.4 Ward's hierarchical method

In the hierarchical method, we adopt 7 as the number of clusters because of the number of cell types. The cluster dendrogram is plotted (figure 3.10).

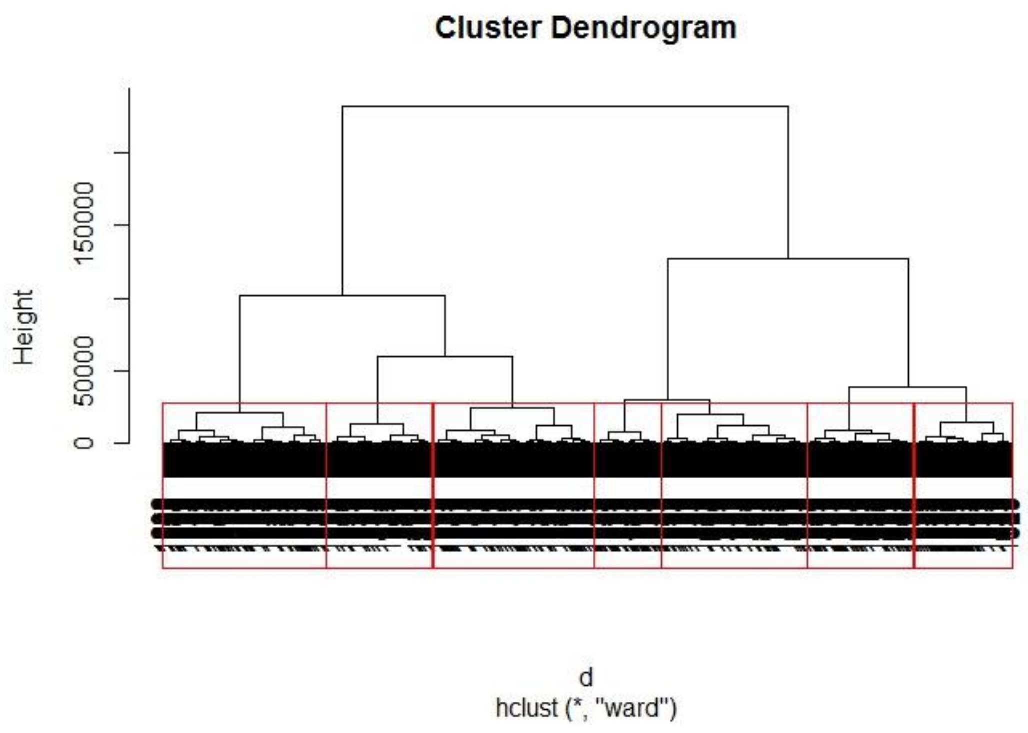


Figure 3.10 Ward's hierarchical clustering. The red boxes divide the objects into 7 clusters. The height means the distance between objects.

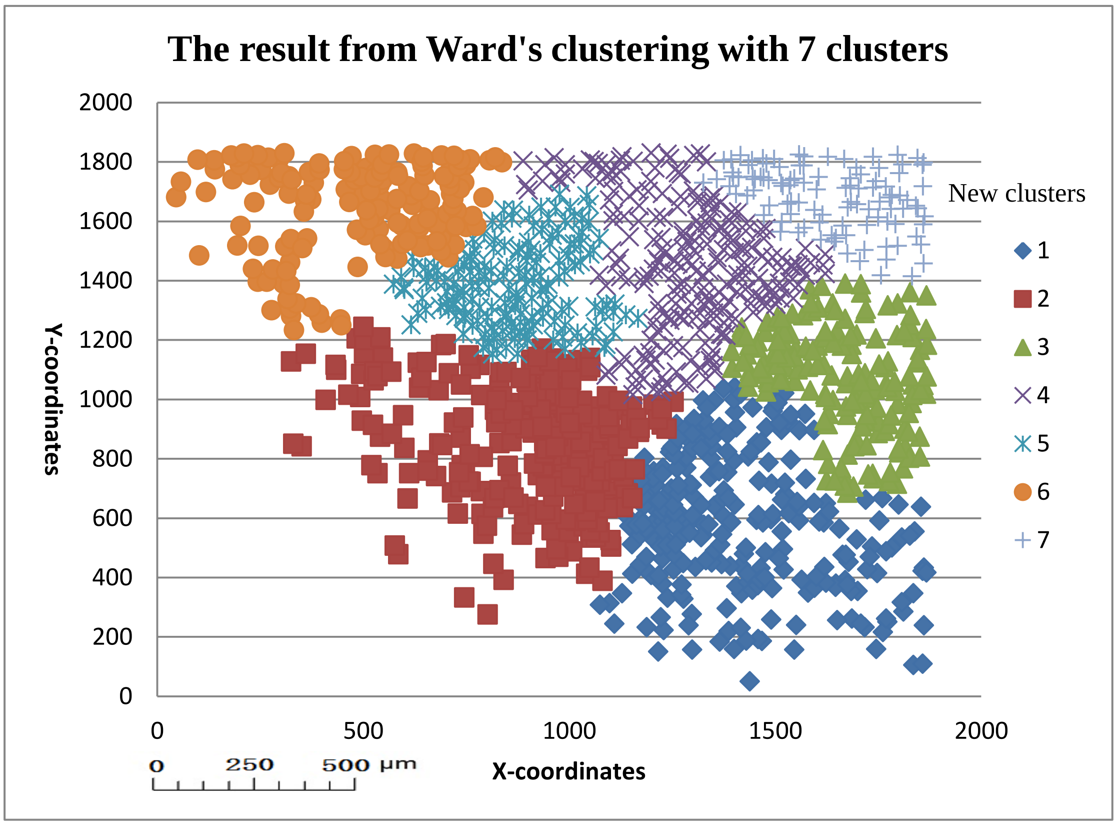


Figure 3.11 Result from Ward's hierarchical method using 7 clusters. The series of number is given to the new clusters. The scale bar of the figure is only approximate.

3.3.4.5 DBSCAN

The DBSCAN algorithm is performed with different values of k. The *k-dist* plot could be like figure 3.12 ($k \in (4,5,6,7,8)$).

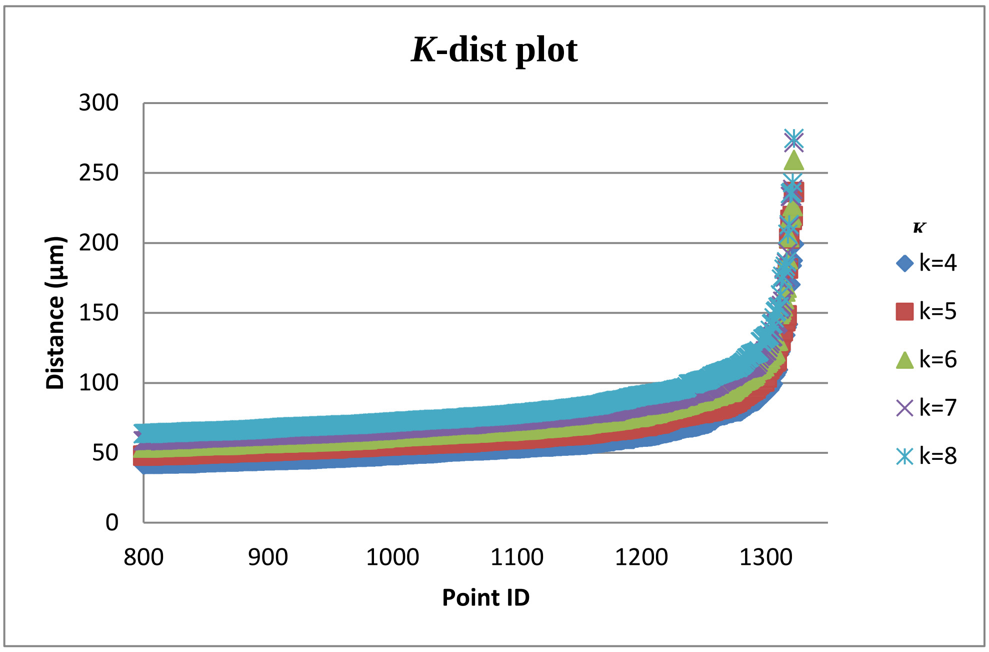


Figure 3.12 The plot for the points with different values of *k*. The *y* axis is the distance from each point to its k^{th} nearest neighbor while the *x* axis is the point ID.

The following graphs displayed the results with different input of *Eps* and *MinPts* (shown in table 3.1).

Table 3.1 Different values for the two input parameters

k	*Eps*	*MinPts*
4	[55, 65]	5
5		6
6		7
7		8
8		9

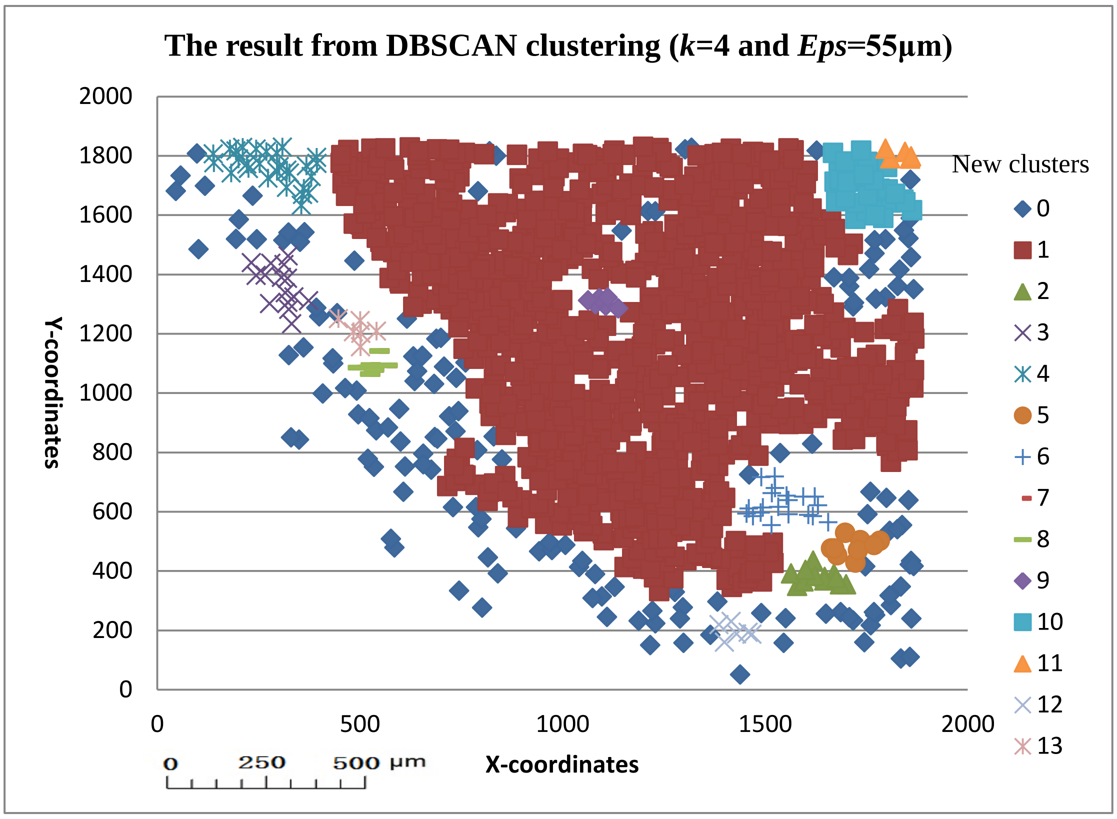


Figure 3.13 The plot for clustering from DBSCAN algorithm when *k*=4 and *Eps*=55. The number 0 mean the noises. The scale bar of the figure is only approximate.

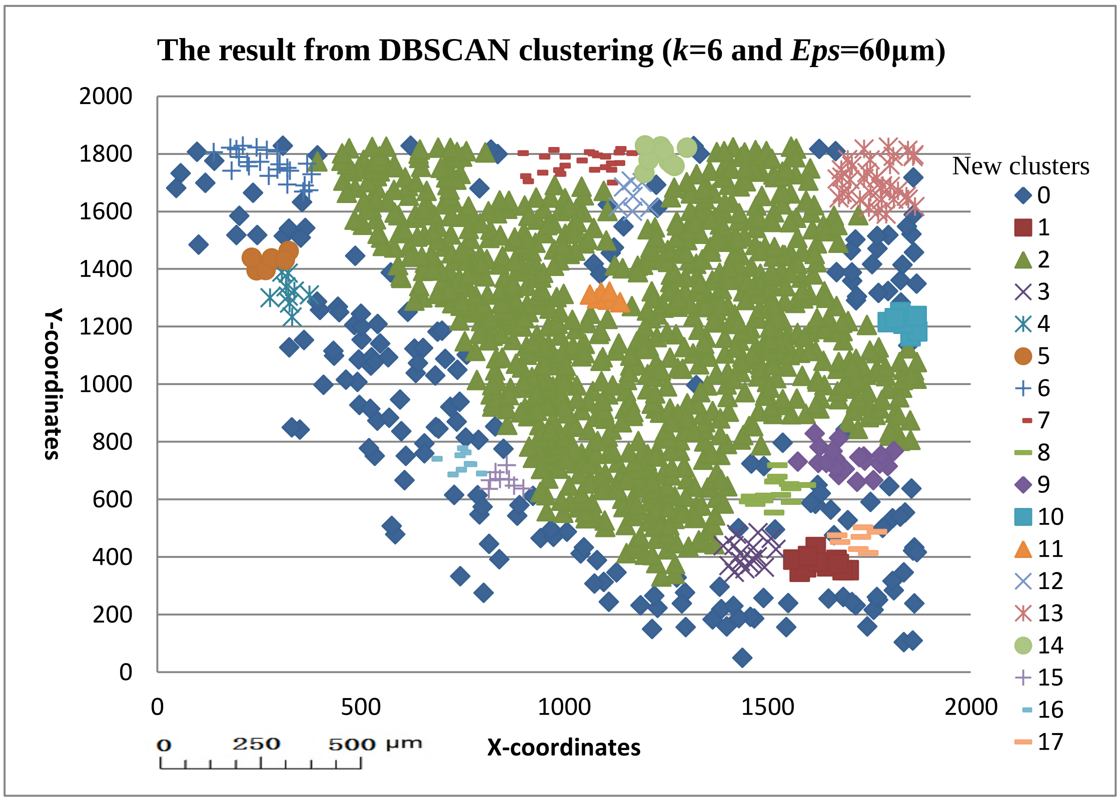


Figure 3.14 The plot for clustering from DBSCAN algorithm when *k*=6 and *Eps*=60. The number 0 mean the noises. The scale bar of the figure is only approximate.

3.3.4.6 Model-based method

The model-based method presents several plots, illustrating how it chooses the number of clusters and what the clusters look like. 9 is determined as the number of clusters according to the largest BIC value. The clusters distributed as figure 3.15 shows.

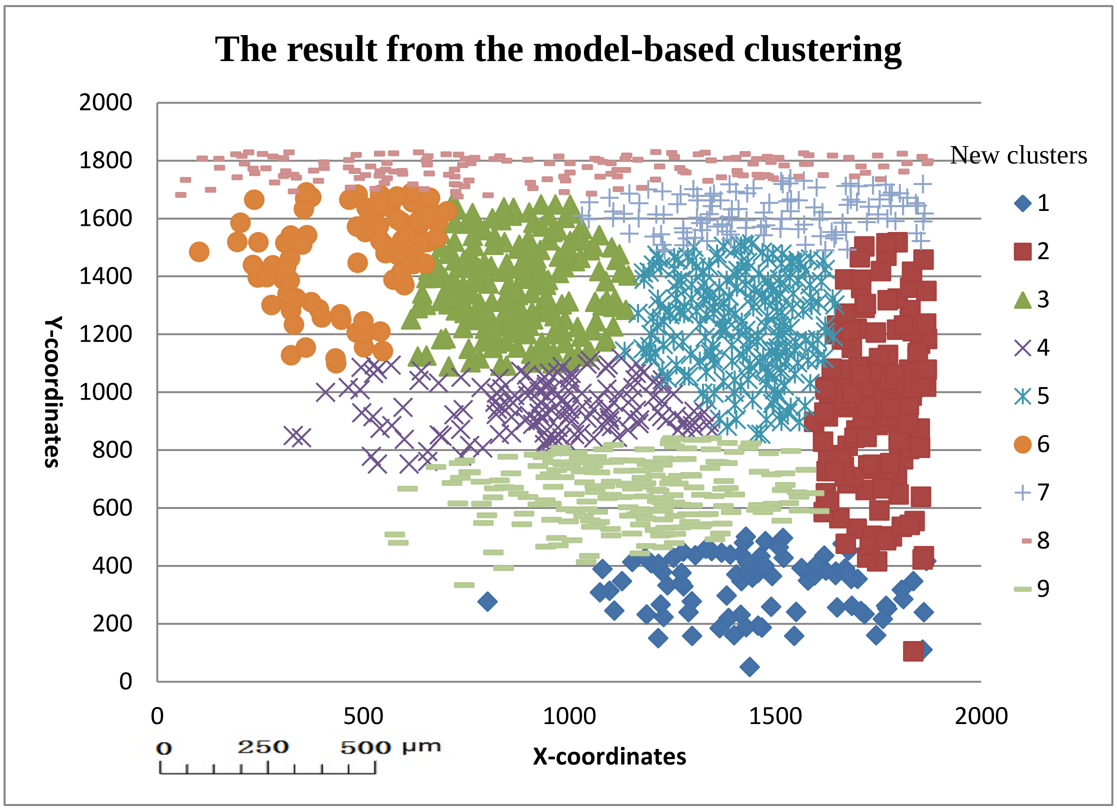


Figure 3.15 The distribution of the clusters when using the model-based clustering. The scale bar of the figure is only approximate.

3.4 Evaluation on methods

As the results show above, the properties of the clustering method are summarized. For the K-mean clustering, the result is dependent on the initial set of cluster centers which are randomly selected. The number of clusters must be defined before performing this method. The clusters tend to be circularly shaped and are of similar size.

Unlike the K-means method, the Ward's clustering could always provide the same results. The number of clusters can be defined by users or it could be decided by a threshold value of the variance within the groups. The shape of clusters is not necessarily circular.

When it comes to the DBSCAN clustering, it is capable of identifying the outliers which are isolated with other objects. The determination of the number of clusters is indirectly made via the parameters *Eps* and *MinPts*. The size of clusters varies substantially and the clusters have irregular shapes.

The model-based method always provides the same result. The clusters often have ellipsoidal form. One element in a cluster could be much closer in space to elements in another cluster than to elements in its own cluster.

3.5 Discussion

According to the results presented above, it has been seen that the clustering methods generate clusters of different shapes. The existence of clusters is confirmed. To know what types of cells cluster together, we could investigate the detailed information of each cluster. That would be meaningful to know which kind of cells interacts with each other. However, the methods except DBSCAN generate regularly shaped clusters. These shapes would probably not correspond to the real situation of the clusters of cells. The DBSCAN is possible to be the most suitable means to detect clusters to some certain extent.

4. Proximity methods

Proximity analysis is an important analytical way to describe the closeness between the selected objects and their neighbors, which is also a crucial means of spatial analysis (Arya and Mount, 2005). As the distances between objects directly reflect the degree of adjacency, most cases of analyzing proximity are based on distances.

4.1 Theory

Proximity tells about the degree of vicinity from one location to its neighboring objects. Neighborhoods can be defined by different metrics. In our studies, proximity analysis is conducted based on the Euclidean distance.

4.1.1 Closeness analysis

Closeness analysis can be graphically delineated in figure 4.1. The calculation of distances relies on the geometry type of the features. Concerning that we are dealing with point objects in our project, the distances are computed by the Euclidean distance from one point to another. Afterwards, a radius is set as a threshold to search for the near features.

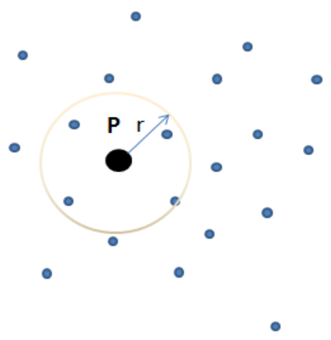


Figure 4.1 Radius searching for neighborhoods where *r* is the radius from the target point *P*

The computational complexity of a distance matrix depends on the amount of data. As the amount of the data in our project is not considerably large, there is no necessity that no other structures of spatial indexation should be used. The index of each point can assist us in obtaining its values.

4.1.2 Proximity analysis based on Delaunay triangulation

The data structure *triangulation* is widely used to store proximity information. Given a set of points in a 2-D plane, the edges can be created by connecting two points. Three non-collinear points constitute a triangle; the sum of all triangles constitutes a triangulation. After the establishment of a triangulation, the vertices in each triangle can be treated as first-order neighbors (see figure 4.2). The vertices that are directly connected to the first-order neighbors are defined as the second-order neighbors.

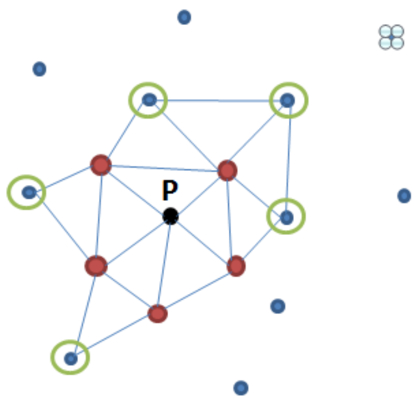


Figure 4.2 Neighborhoods searching based on triangulation network. Red points are first-order neighbor and the ones within the green circle are second-order neighbors.

A triangle can be arbitrarily created with three points; the triangulation is not unique. However, sometimes it is required that the triangles possess certain properties. Delaunay triangulation is one of the most common triangulations. The two most distinguishable properties of a *Delaunay triangulation* are (Worboys and Duckham, 2004):

Circumcircle property: the circumcircle of a triangle in the Delaunay triangulation must not contain any other points. Basically, if we draw a circle that passes through three points, no other points are inside this circle. Figure 4.3 shows an illegal triangle.

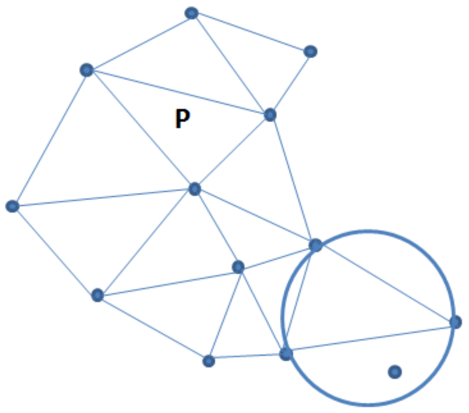


Figure 4.3 The circumcircle property of Delaunay triangulation

Equilateral property: The Delaunay triangulation maximizes the minimum angles. The triangles in the Delaunay triangulation should be as equilateral as possible (Figure 4.4). There would not be any long and thin triangles.

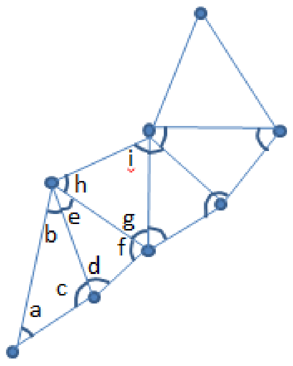


Figure 4.4 A description for the equilateral property of Delaunay triangulation. Assume that we have a set of angles: a, b, c …, and if we organize these angles in an ascending order, for example, a, e, b, d, f … the minimum angle is maximized.

4.1.3 Combined closeness and Delaunay triangulation analysis

A combination of closeness analysis and Delaunay triangulation analysis is made to look for neighborhoods. The first-order (or any order) points within a certain distance can be regarded as neighbors (figure 4.5).

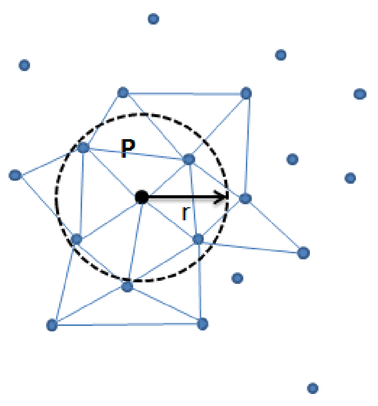


Figure 4.5 A combination searching for neighborhoods

4.1.4 Definition of neighborhoods

Neighborhoods hereby are defined corresponding to the methods introduced above.

1. Neighborhoods within a distance threshold: the points which are in a radius of a target point are taken as neighborhoods (referring to section 4.1.1).
2. *1st* degree neighbor: the points that are directly close to the target point or share an edge with the target point are regarded as neighbors (see red dots in Figure 4.2).
3. *N*-degree neighbor: a point that shares an edge with an *n-1* degree neighbor (see dot with green circles in figure 4.2).

4.2 Implementation of proximity methods in R

4.2.1 Closeness method

As the closeness method is solely dependent on the Euclidean distance between objects, the primary task is to find the searching radius. A histogram will be made for the distances and the frequency of each distance interval is presented in a tabular way. Afterwards, a cumulative percentage of the distance interval is calculated in order to determine a threshold. In the end, neighborhoods are able to be found in a searching radius from the target points.

The proposed closeness method is inspired by the *near* function in ArcGIS, which defines the shortest Euclidean distances from the input features to other features (ArcGIS help). Figure 4.6 delineate how the *near* function performs.

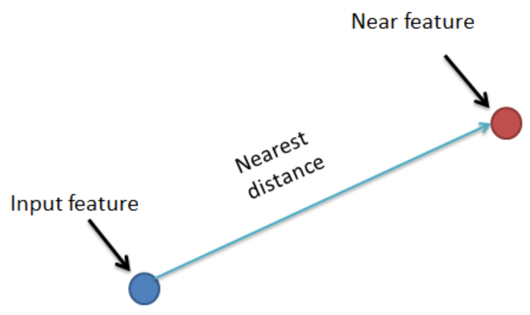


Figure 4.6 An explanation for *near* function

4.2.2 Delaunay triangulation analysis

Delaunayn, which is enclosed in the package *geometry*, is usually used to create the Delaunay triangulation network in *N*-dimensions. A point dataset is necessary to be the input parameter. The resulting matrix will present an *n* by 3 matrixes where each column stores the indexes of vertices. Then the *1st* degree as well as *n*-degree neighbors can be found through the vertices matrix.

4.2.3 Combined closeness and Delaunay triangulation analysis

In the combined method, the first process of data is to create the Delaunay triangulation. *N*-degree neighbors within a certain distance from a point can be identified.

4.3 Case study II: evaluation of neighborhood methods

4.3.1 Introduction

The neighbors are always defined as the close objects in the macroscopic world. However, in the microscopic world, whether the close enough cells can be treated as neighbors is what we are about to investigate in this case study. We carry out the closeness method and the combined closeness and Delaunay triangulation method separately. Both of the methods are focusing on the immune cells around the viruses. As the second question proposed in the section 1.2, the neighbors of the viruses would be known.

4.3.2 Data

The data to be evaluated are the same as the one in case study I (see figure 3.5).

4.3.3 Method

The procedure of proximity analysis is performed in R and is arranged using Excels. There are two methods to test data; one is conducted by calculating the Euclidean distance from points to points. The other one is the combined closeness and Delaunay triangulation method.

The known coordinates of each object are imported into R. We are mainly interested in how the viruses influence the immune cells. Therefore it is necessary to extract the virus cells from the whole datasets. The calculation of distance between the immune cells and viruses is done with their locations. Another calculation of distance between the viruses is necessary to see if a virus is adjacent to the other.

In the combined method, we utilize the *Delaunayn* function in R. A matrix of vertices of each triangle is provided by this function. A computation of each edge is done according to the point ID of each vertex. The detail commands could be referred to in Appendix A.7.

All the distances, either in the closeness method or in combined closeness and *Delaunay triangulation* method, are separately sorted out in the histograms. It could be clear to detect the frequency of each distance interval when we transfer the histograms into tables. A further process of these distances is finalized in the cumulative percentage tables, through which we determine the influential distance that is the threshold used to search for neighbors. In the end, a visualization of neighbors is provided.

4.3.4 Result

4.3.4.1 Closeness method

A table (table 4.1) is created to store the frequency of each distance interval between viruses and immune cells and we plot the cumulative percentages against the distance intervals in figure 4.8. It is noticeable that the distances are computed from the viruses to their first-order “neighbors”.

Table 4.1 The frequency table of distance intervals

Range	Count	Cumulative frequency	Percentage
(0, 50)	24	24	0.304%
[50, 100)	79	103	1.303%
[100, 200)	281	384	4.860%
[200, 400)	899	1283	16.236%

We set the distance threshold as 50 μ m and visualize the neighborhoods of the viruses in figure 4.7.

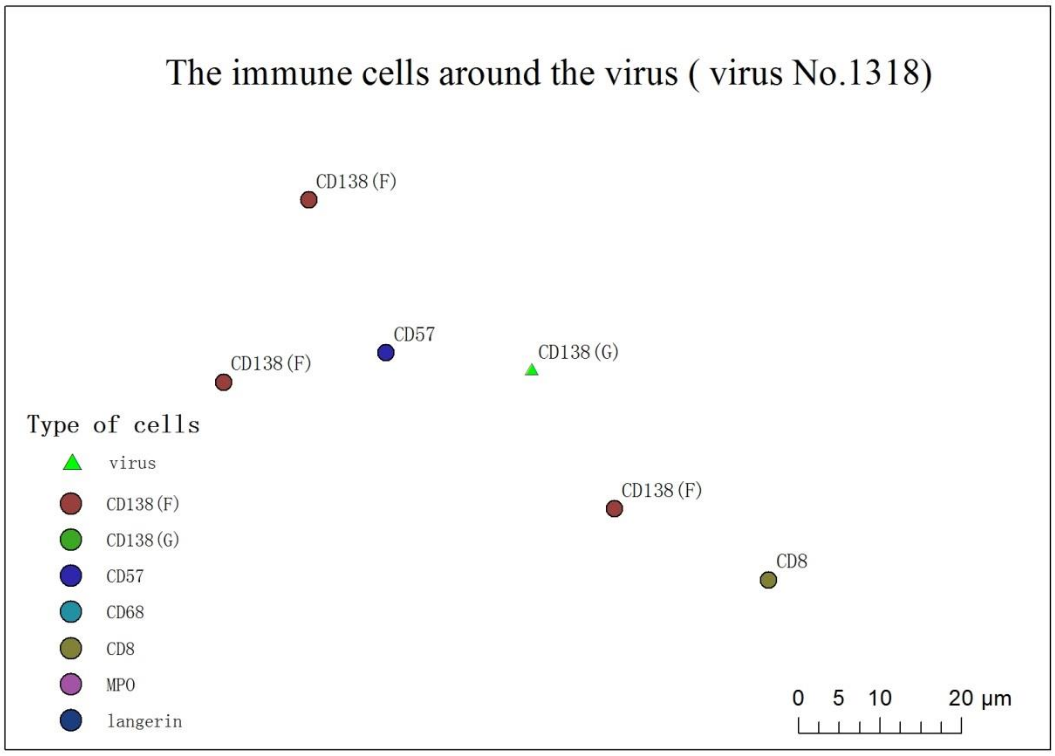


Figure 4.7 A visualization of distribution of neighborhoods (Virus No.1318). The scale bar of the figure is only approximate.

4.3.4.2 Combined closeness and Delaunay triangulation method

Delaunayn function is first performed to obtain the vertices of triangles. Afterwards, an extraction of edges with viruses is accomplished. The distances between viruses and their *1*st degree neighbors of immune cells are divided into 4 intervals and the frequency of intervals is described in table 4.2. The cumulative percentage is calculated by the number of certain distance interval divided by the total number of edges.

Table 4.2 The frequency of each distance interval and their percentages

Range	Count	Cumulative frequency	Percentage
(0,50)	16	16	47.059%
[50,100)	10	26	76.471%
[100,200)	4	30	88.235%
[200,400)	2	32	94.118%

According to the cumulative percentage plot above, the value of threshold is set to be

90μm. The edges larger than 90 are removed. The final visualization of the Delaunay triangles is presented in figure 4.8.

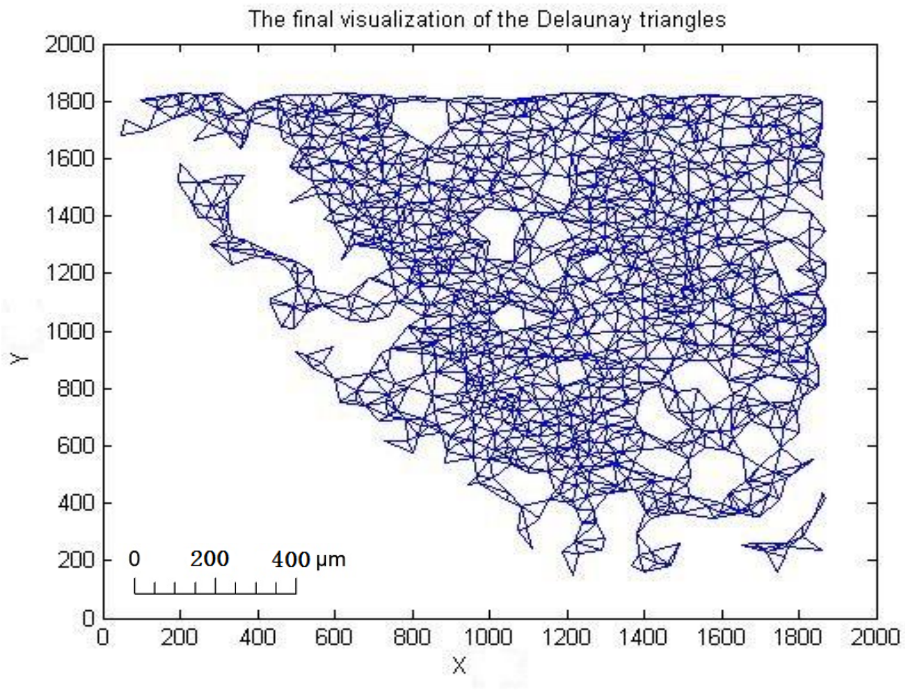


Figure 4.8 The final visualization of the Delaunay triangles. The scale bar of the figure is only approximate.

An example of the neighborhoods of viruses is delineated in figure 4.9.

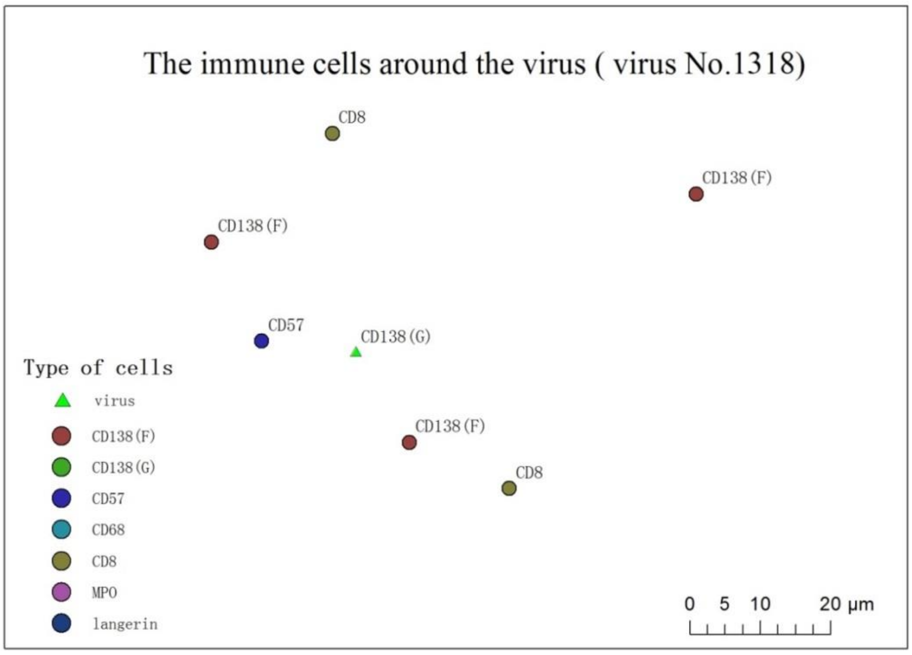


Figure 4.9 A part of neighborhoods of the viruses (No. 1318). The scale bar of the figure is only approximate.

4.4 Evaluation on methods

Hereby we summarize the properties of the proximity methods. The closeness method only utilizes a Euclidean distance threshold to define neighbors. However, the determination of the threshold is not convincible here. In the combined method, objects that are visually close enough are not necessarily accounted as neighbors. Neighbors should be the vertices sharing the same edges with the viruses.

The main difference of these two methods lies in that the closeness method directly calculates the distances between viruses and immune cells while the combined method firstly creates the Delaunay triangulation network. Obviously, the comparison between table 4.1 and table 4.2 shows that even with the same distance interval, the number of *1st* degree neighbors in the combined method is fewer than that of the closeness method, which means not all the immune cells within a certain distance are first-order neighbors in the Delaunay triangulation.

4.5 Discussion

Comparing the two tables (table 4.1 and table 4.2), it seems contradictory that the larger number of frequency accounts for a smaller percentage. This phenomenon could be explained as: both of the methods look into the *1st* order "neighbors". The closeness method calculates the distance from the immune cells to the viruses. Hence all the viruses are treated as the *1st* order "neighbors" (not all the immune cells are the neighbors of the viruses, for instance, the immune cells far away cannot be defined as neighbors) in this situation. However, the combined method only investigates the *1st* order neighbors which share the edges with the viruses. It is clear that the dividers in these two methods differ; the divider in the closeness method is larger than that in the combined method. For the purpose of getting an explicit comparison, normalization is done by using the divider in the closeness method (the divider is 7902). The new result in the combined method is presented in table 4.3. The same distance interval takes up a lower percentage.

Table 4.3 The new result normalized by the divider in the closeness method

Range	Count	Cumulative frequency	Percentage
(0,50)	16	16	0.202%
[50,100)	10	26	0.329%
[100,200)	4	30	0.380%
[200,400)	2	32	0.405%

From figure 4.7 and figure 4.9, more immune cells are found in the combined method, which is caused mainly by the searching radius.

5. Algorithm of constructing 2D hulls for a point set

5.1 Theory

The aim of this section is to define methods for constructing the hull of a point set (i.e., the exterior boundary). A hull can be regarded as line that circumvents all the points that belong to a cluster of points. The most commonly used hull is the convex hull. The convex hull is the minimum convex polygon that contains the entire given elements. However, convex hull is not properly served for all applications. The circumscription of a given dataset would probably include blank areas where no data exist. To avoid the point-free areas, the outline should be as similar to the supposed shape of points as possible. Furthermore, several point clusters include larger areas where there are no points. In such a case the hull should have holes.

This section starts with some algorithms to compute the convex hulls. Then our methodology to compute the 2D hull is described. This methodology handles both point-free areas along the border and in the middle of the clusters (i.e. holes).

5.1.1 Algorithms to compute the convex hull

The Graham's algorithm first sorts all the points and find out the leftmost point *p*. Then connections are created from the point *p* to all the other points. According to the angles between lines in the polar coordinate system, the hull is generated by connecting points in a counterclockwise order, the point *p* being the starting and ending point (Bo et al, 2007). A brief explanation of the Graham's algorithm is shown in figure 5.1.

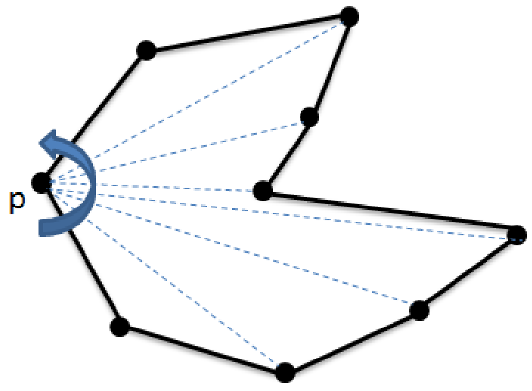


Figure 5.1 An explanation for the Graham's algorithm

Like the Graham's scan algorithm, points are sorted in the Jarvis's marching method and the point with the lowest *y*-coordinate is chosen as the starting point. Basically, the Jarvis's marching is like a gift wrapping which encloses all the points to create a convex hull. Hence, we connect the points from the starting point in a counterclockwise sequence. Only the points where there are no other points on the left side of the line created from the last point to the current point are adopted as the vertices of the hull (Bo et al, 2007). Figure 5.2 illustrates how the Jarvis's marching algorithm performs.

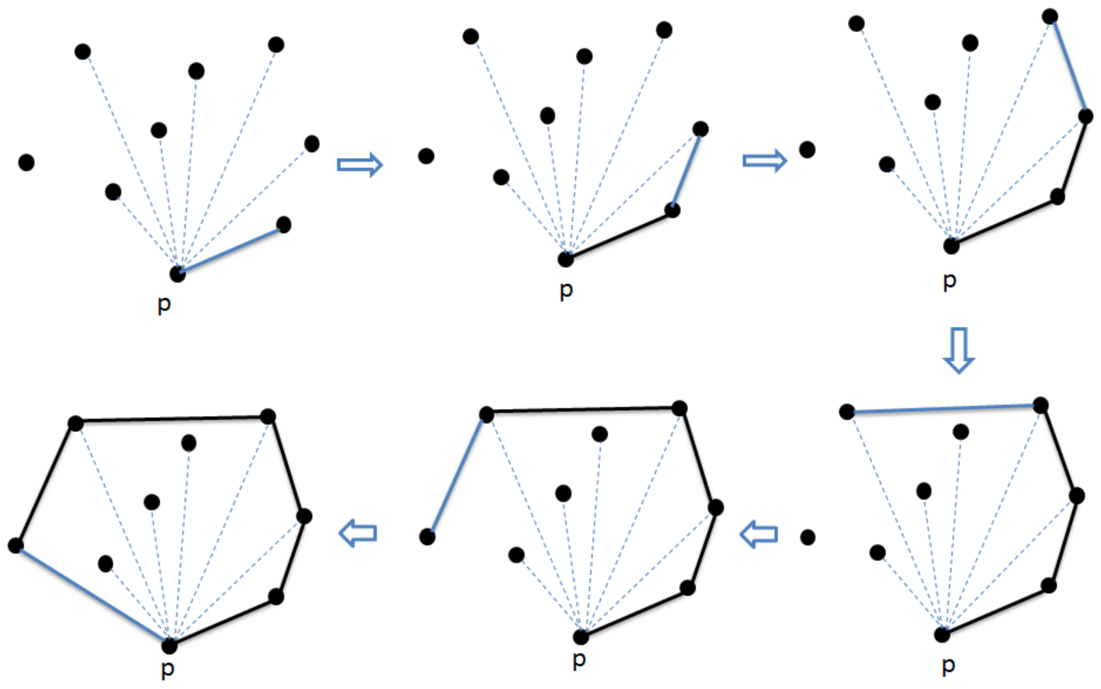


Figure 5.2 An illustration for the Jarvis's marching algorithm.

5.1.2 Our proposed 2D hull method

The method we developed is partly inspired by Joubran and Gabay (2000). The method starts by computing the Delaunay triangulation. Secondly, we create a convex hull by using the Jarvis's marching algorithm (one could also use the property that the outer edges of the Delaunay triangulation equal the convex hull, but in our case it was simpler to use Jarvis's algorithm). Then the point at the bottom is selected as the starting point. All the vertices of the convex hull will be checked and the criterion will be set that if the edge constituted by the last vertex and the current vertex is longer than a certain distance threshold, the current vertex will be replaced according to the vertices of the Delaunay triangles (illustrated in figure 5.3). The replacement is possible to be successively taken until the edge satisfies the criterion. An exception would happen if there are no short enough edges found.

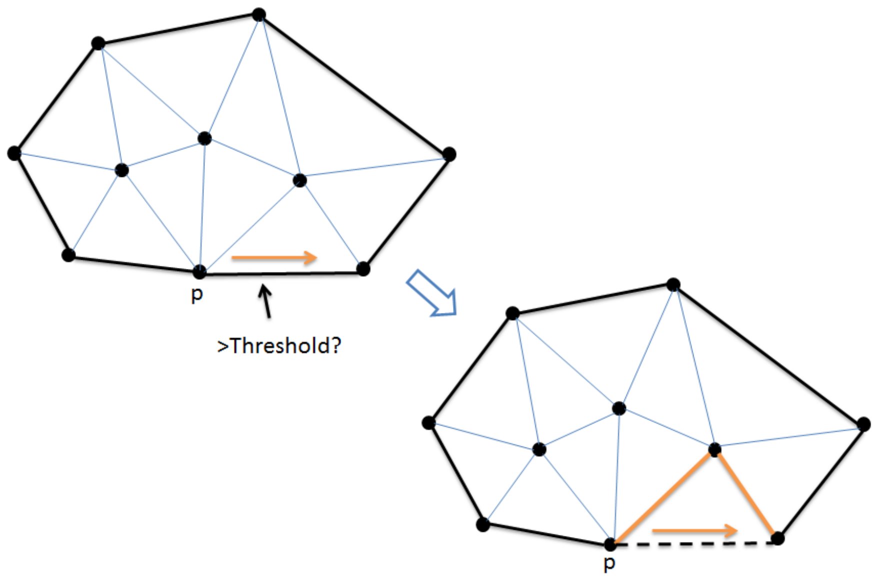


Figure 5.3 A replacement will be taken only when the current edges are larger than the threshold.

The determination of the threshold is accounted by an analysis for the edges of Delaunay triangles. Hereby we will make a histogram for those edges to check the frequency of certain length intervals. A satisfactory threshold is decided when 90% of the edges are with a length interval.

As a matter of fact, the threshold is determined based on a couple of trials. On the one hand, it is meaningless to choose a large value so that the shape of the convex hull will not change too much. On the other hand, a small value would result in an infinite loop because no replaceable edges could be found.

However, the external boundary is optimized by the method described above. The internal problem “holes” is needed to be solved. The holes are constituted by the triangles which at least have two long edges. Using the threshold we discussed before, we investigate and mark the point from which two edges of a triangle are larger than the threshold. Then the single triangles are removed and the adjacent triangles are merged as polygons.

The only thing we know is that the holes are formed by several triangles. But it is unknown that the shape of the holes is either convex or concave. In this sense, the Graham’s scanning method could be a better option to construct the holes.

5.2 Implementation of our 2D hull method

Our proposed method is implemented in Matlab, which prevails in computations especially when we deal with the computational geometry problems while R is capable of handling statistical problems.

The implementation can be roughly divided into two parts. The first part is about creating the Delaunay triangulation and the second part, which is the key part of this algorithm, is to draw the hull of the point set. The detailed script can be referred to in Appendix A.8.

As for the creation of Delaunay triangulation, the detailed introduction and implementation can be referred to in chapter 4. Function *delaunayn* is deployed to constitute the Delaunay triangles. The resulting matrix containing the vertices of each triangle is useful for the next part. Moreover, the edges of the triangles are calculated.

After obtaining the relevant information of the Delaunay triangles, we commence on constructing the 2D hull. Function *chull* is first utilized to get the convex hull of this point set. The vertices of this convex hull are stored in a vector. We start with the first vertex of the convex hull and check if the distance between the starting vertex and the next vertex is larger than a certain value. If the distances between the two neighboring vertices are larger than that value, we will look into the triangle that contains these two vertices and treat the third vertex of this triangle as the next vertex of this hull. This step will proceed and iterate until the distance is smaller than the threshold. The addition of new vertices will update the vector which originally consists of vertices of the convex hull. In the end, the 2D hull is accomplished by connecting the vertices in the vector.

In order to delineate the holes, we go through each point and find out the triangles from this point that have two edges larger than the threshold. The holes are created by multiple triangles such that the single triangles are removed. In addition, the holes which intersect the circumscribed hull are removed as well because the holes are not totally within the hull. It is assumed that the vertices of each triangle would also be the vertices of the holes if no edges are shared by two triangles. With the vertices of the triangles, we defined a function called *graham* to construct the border of the hole. The detailed codes of delineating the holes are attached in Appendix A.9 and the defined function is given in Appendix A.10.

5.3 Case study III: computing the 2D hull

5.3.1 Introduction

This part is to delineate a minimum circumscribed hull of points and furthermore to construct the “hole” inside the hull. The Delaunay triangulation is of necessity to be generated beforehand. The developed method of delineating the hull is based on the Delaunay triangulation. The holes hereby are defined as an individual triangle or a combination of triangles whose two edges are larger than a threshold. This proposed method will be evaluated and utilized in later case studies.

5.3.2 Data

We select one sample of points to test our algorithm. The distribution of points is given in Figure 5.4. The data contain the coordinates of each point.

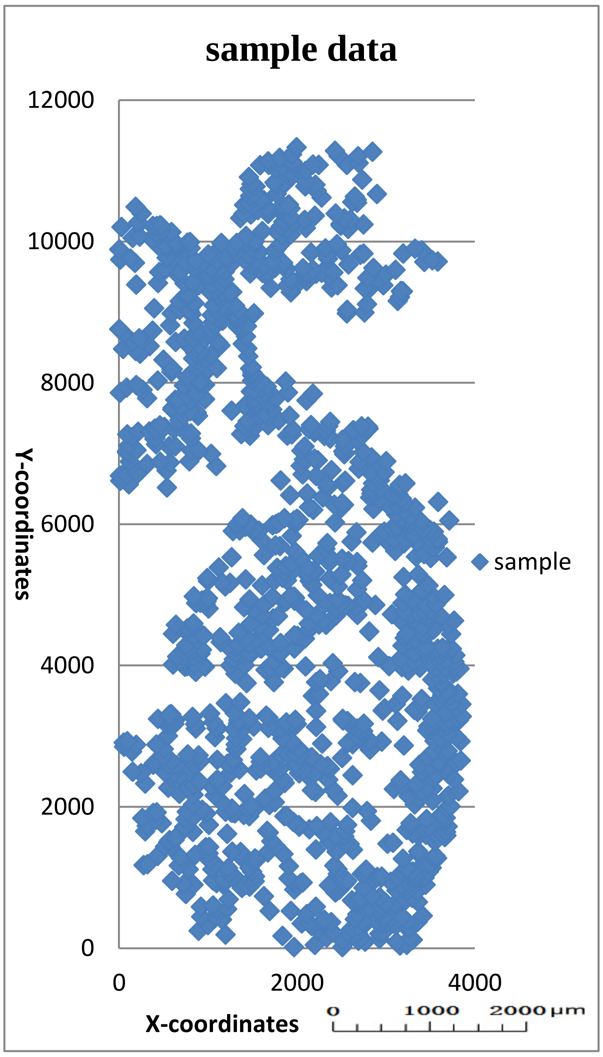


Figure 5.4 The distribution of sample points. The scale bar of the figure is only approximate.

5.3.3 Method

The concrete description can be referred in the section 5.1.2 and the implementation is described in the section 5.2.

5.3.4 Result

According to the edges of the Delaunay triangles (shown in figure 5.5), the value of the threshold is set as 428µm.

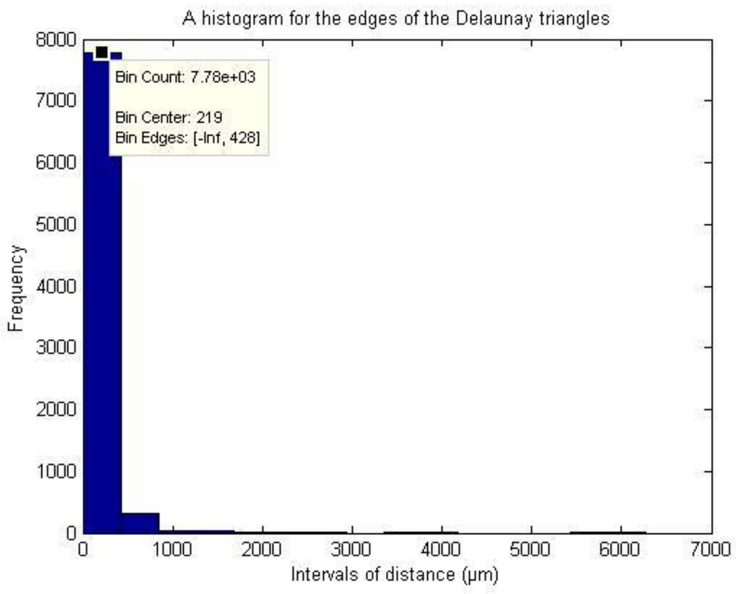


Figure 5.5 A histogram for the edges of the Delaunay triangles, x axis is the length of the edges and y axis is the corresponding frequency

For the convenience of detecting the changes of convex hull into to our expected hull, the convex hull and Delaunay triangles are kept in the figure 5.6.

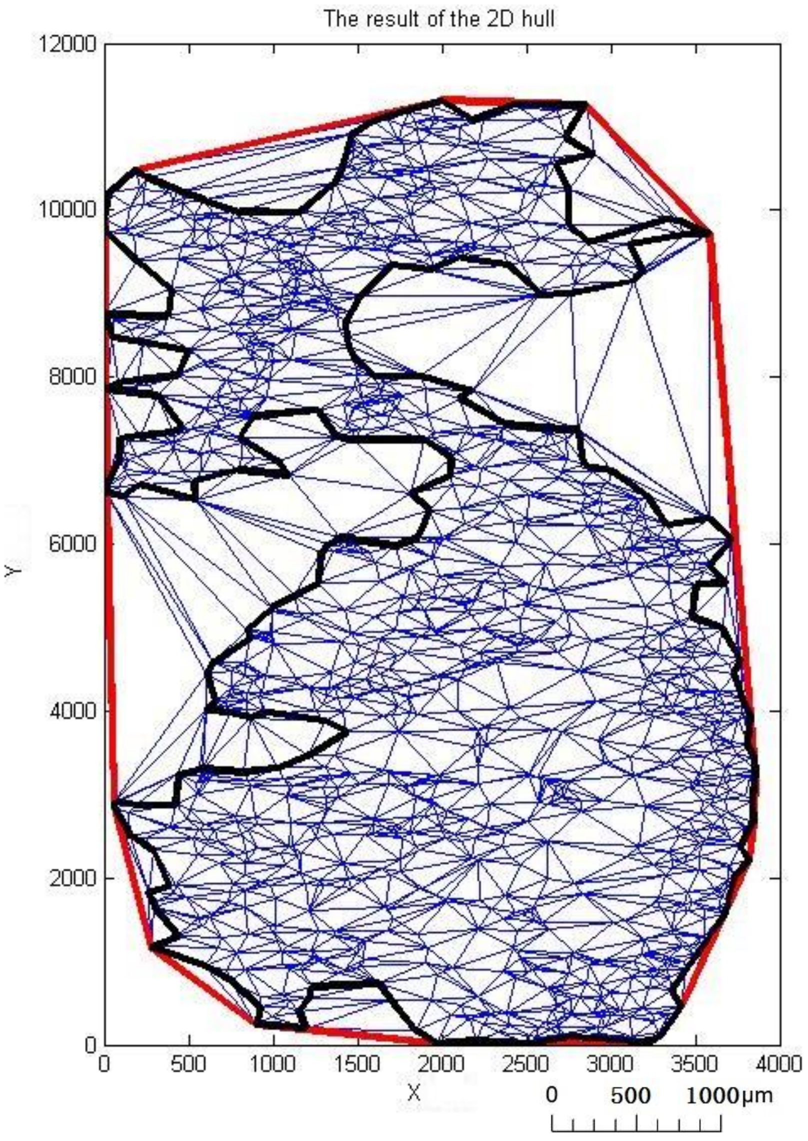


Figure 5.6 The result of the 2D hull. The red thick line is the convex hull and the blue lines are the Delaunay triangles. The black line constitutes the 2-dimensional hull. The scale bar of the figure is only approximate.

The holes are marked by thick blued lines. See in the figure 5.7.

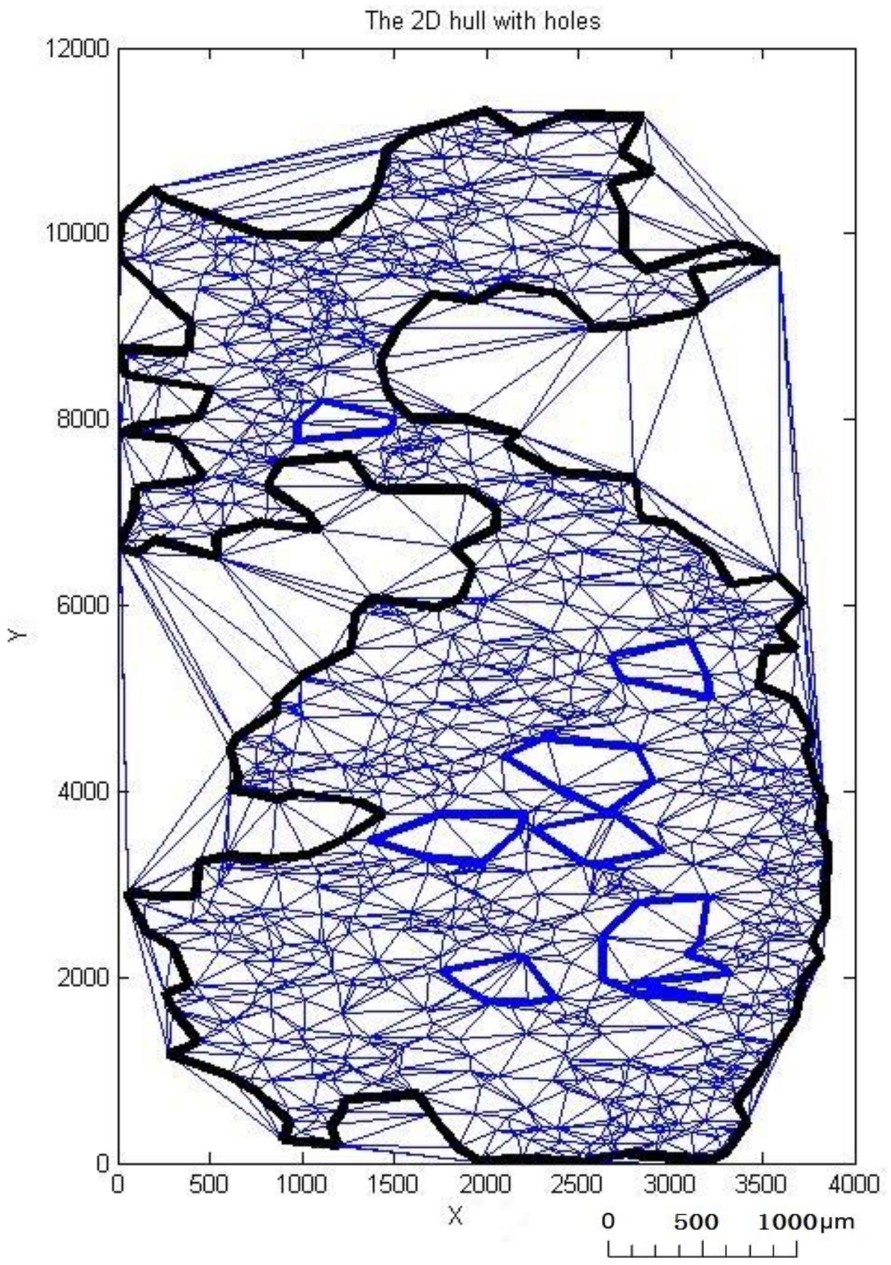


Figure 5.7 The holes are delineated by thick blue lines. The blue lines delineate the Delaunay triangulation and the black line presents the 2-dimensional hull. The scale bar of the figure is only approximate.

5.4 Evaluation on methods

Our proposed method to construct the 2D hull has the following properties: It is based on Delaunay triangulation. The hull is possibly concave by exclusion of blank areas.

In addition, the method can create holes in the hull.

5.5 Discussion

The problem of drawing the 2D hull is about setting a proper threshold. When the threshold is between 280 and 400, some of the hull edges are still larger than this threshold. However, this problem is not able to be avoided. Figure 5.8 is a partial screenshot of the hull.

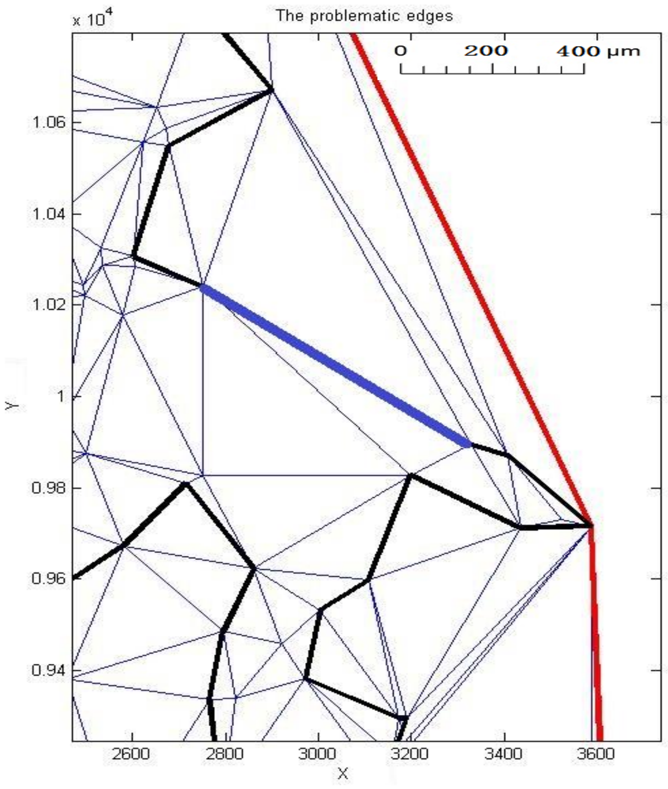


Figure 5.8 The problematic edge highlighted by the thick blue line, The scale bar of the figure is only approximate.

Actually the long edge (in blue) can be replaced by other edges (shown in figure 5.9).

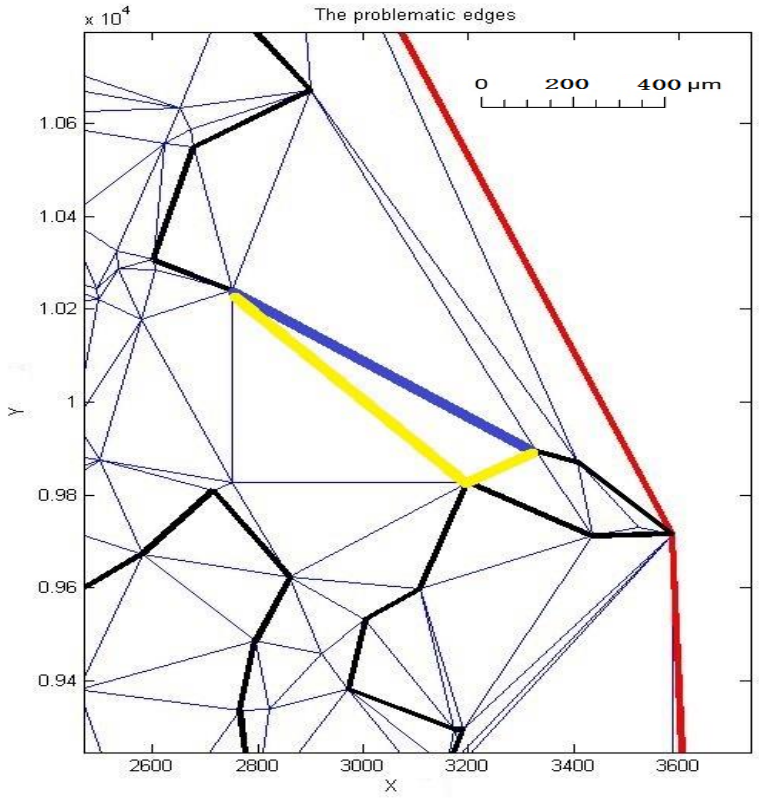


Figure 5.9 The problematic edge in thick blue is supposed to be replaced by the other two edges rendered in thick yellow. The scale bar of the figure is only approximate.

If we replace the long edge with those two edges, the hull is not an irregular shaped polygon; instead it becomes two polygons. As the vertices of the triangles are the points, all the points are divided into two polygons, which is not our purpose.

When we merge the adjacent triangles to form the holes, another problem occurs. If there is a triangle sharing its edge with another two triangles, the Graham's scanning is not capable of handling correctly (see figure 5.10). The green lines delineate such triangles. In the Graham's scanning algorithm, the hull is created by all the points used. Apparently this hole is incorrect.

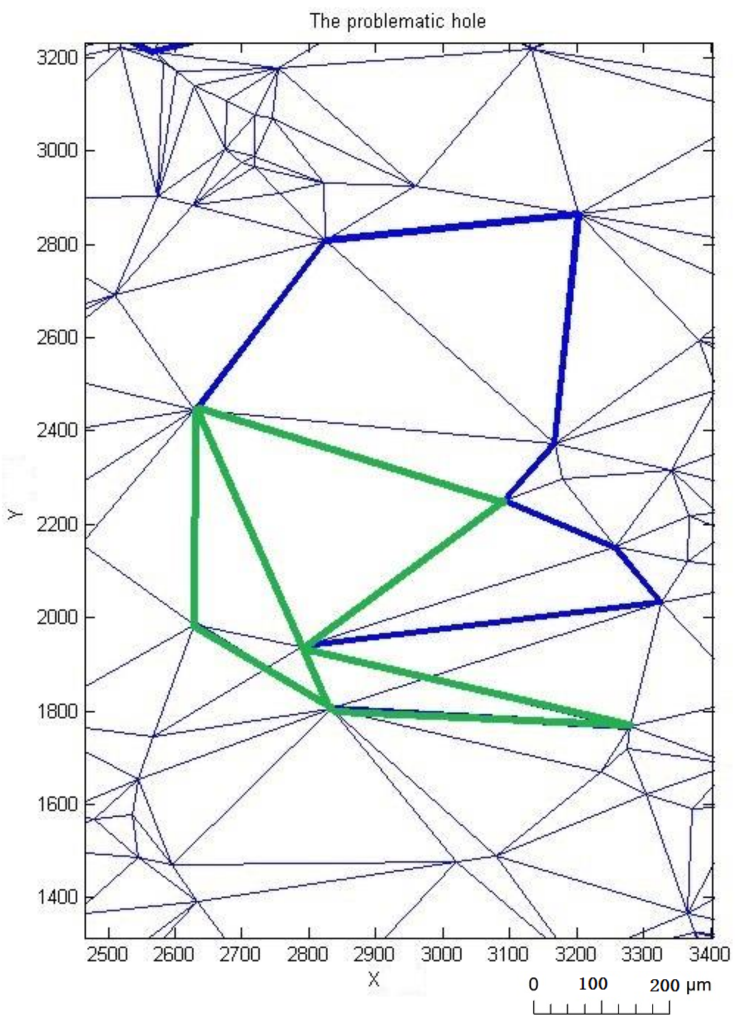


Figure 5.10 The problematic hole: one triangle shares an edge with two other triangles (triangles in green). The scale bar of the figure is only approximate.

6. Case study IV: Delineating compartments in tissues

6.1 Introduction

The compartments exist in the tissue section. In this chapter, we will attempt to delineate the compartments by using the developed algorithm in chapter 5. In order to obtain a more convincible consequence, the quality of the proposed algorithm is evaluated based on the real data provided by the Medetect. The medical partner manually delineates the compartments in the tissue sections.

The workflow can be generally formulated as below:

1. Data preparation: the data that are necessary to be processed are extracted.
2. Data processing: the data are firstly processed by using the DBSCAN clustering.
3. Hull generation: the hulls are generated by delineating the clusters.
4. Comparisons: a comparison is carried out by overlaying the manually made compartments with the automatically made hulls. Then we visually investigate the similarities between these two layers.

An evaluation is followed by the comparisons made between these two kinds of hulls. If the generated compartments well correspond to the manually made ones, it will be feasible to use this method to implement relevant studies.

6.2 Data

The data are presented in figure 6.1. The colorful dots stand for the cells the medical partner extracted. The manually made compartment was delineated by the yellow circle. The location of all the cells was specified in a standard text file (XLS-file).

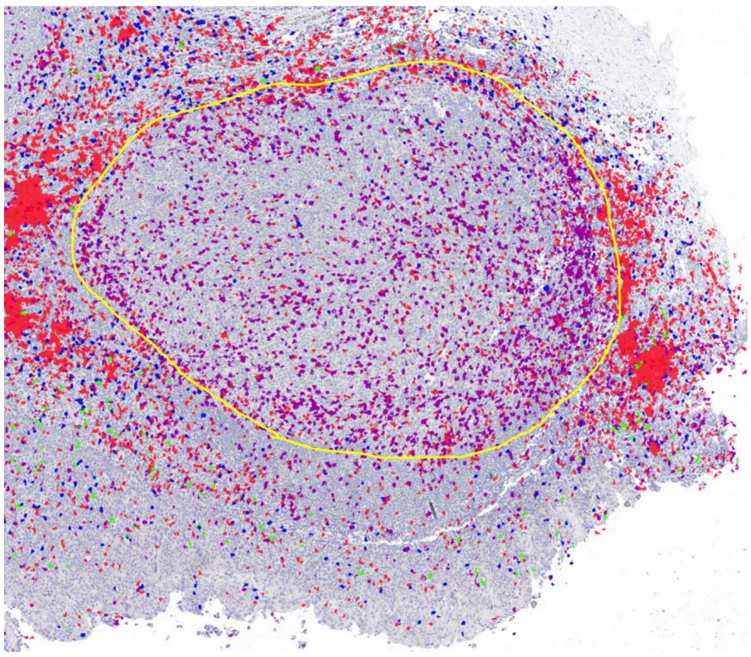


Figure 6.1 The manually delineated compartment in yellow by Caroline

However, our analysis only focuses on two specific types of cells instead of the entire dataset, CD57 and CD3 which are recommended by Medetect. Figure 6.2 shows the plot of these two types of cells by their coordinates.

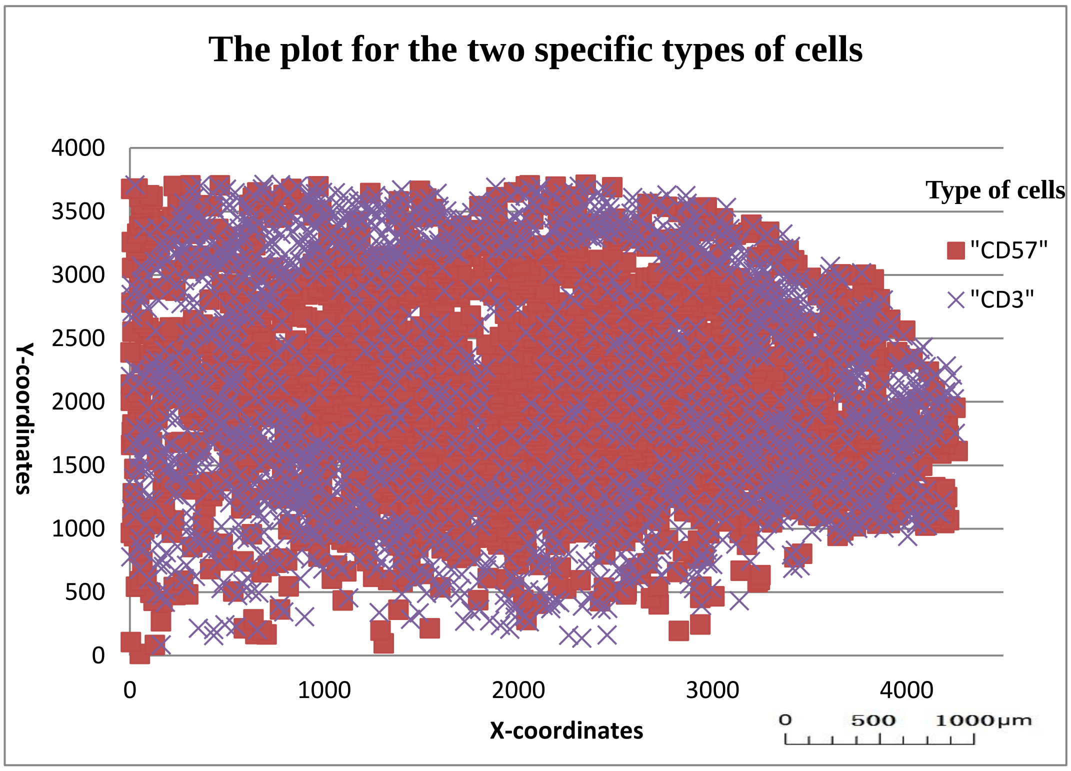


Figure 6.2 The plot of the two specific types of cells. The scale bar of the figure is only approximate.

6.3 Implementation

As shown in figure 6.1, the compartments can be regarded as tiny clusters. Due to the fact that the DBSCAN clustering could obtain irregular shaped clusters, DBSCAN is adopted to generate clusters. A challenge here is to find suitable values for the parameters *Eps* and *MinPts*. Generally, small values for these parameters result in a larger number of clusters. Based on the *k-dist* plot (shown in figure 6.3), we select 60 and 4 for *Eps* and *MinPts* respectively. With the proposed hull algorithm (described in chapter 5), we finally construct the hull for each compartment.

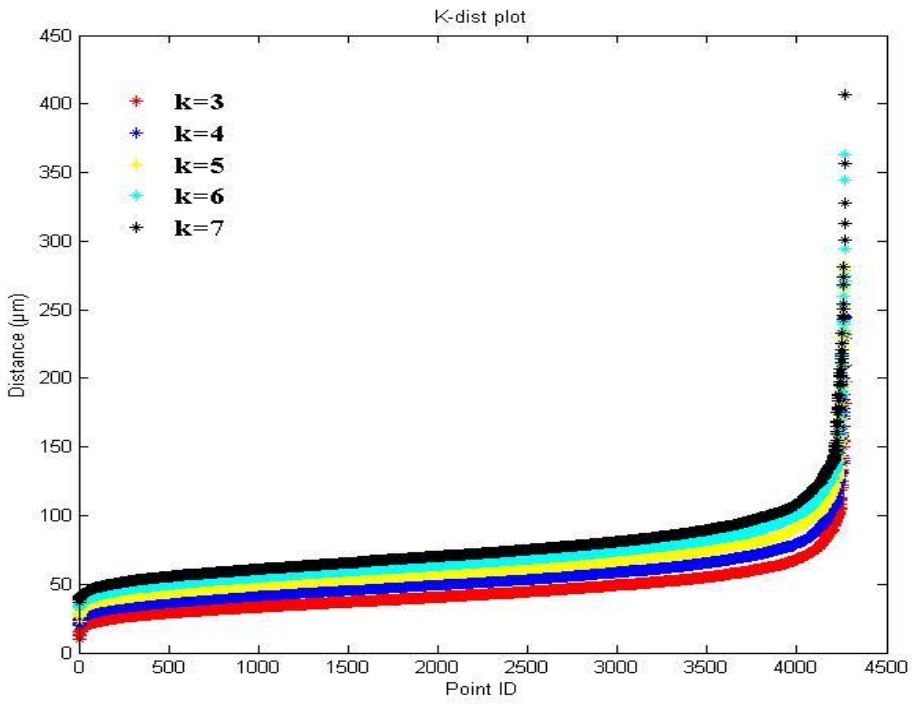


Figure 6.3 The *k*-dist plot made with type CD3 and CD57 cells

6.4 Result

The hulls for the compartments are constructed by delineating the borders of the above clusters in figure 6.4. An overlay is accomplished with the figure 6.1.

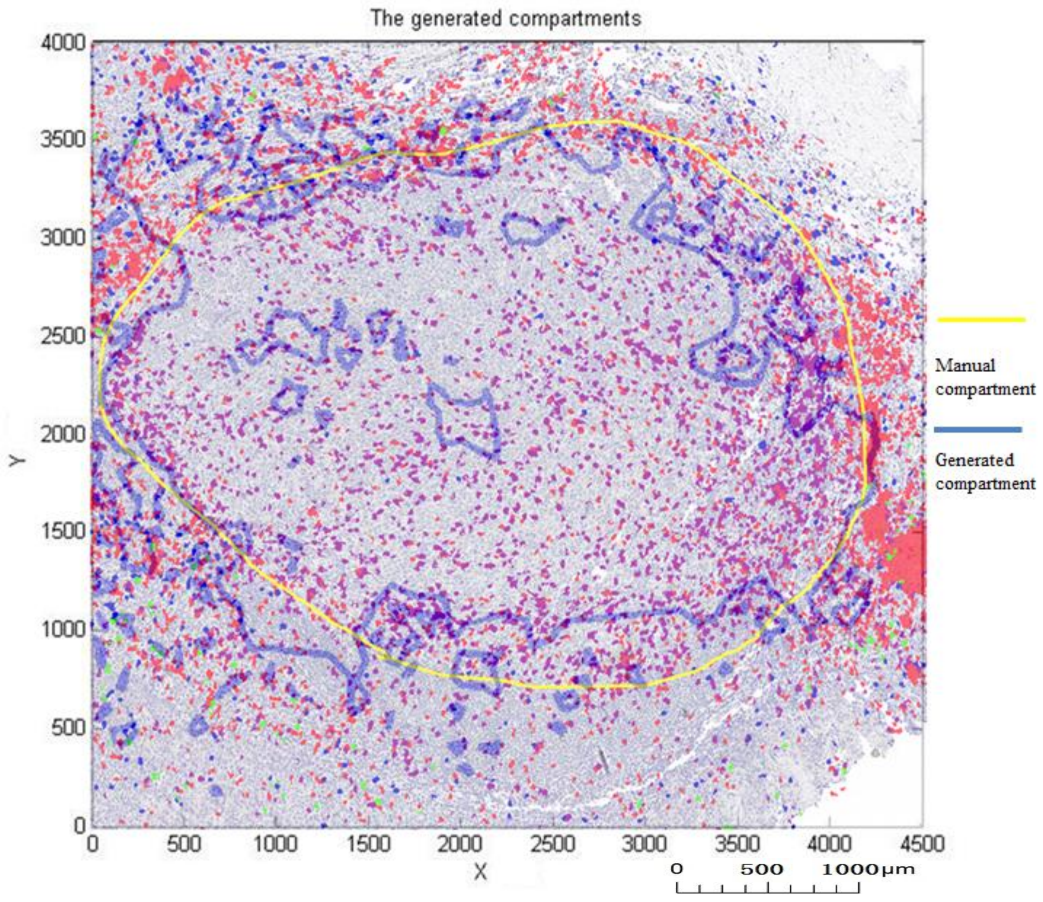


Figure 6.4 The resulting hulls made by the proposed method. The blue lines are the borders of the generated clusters and the yellow line is the compartment. The scale bar of the figure is only approximate.

6.5 Discussion

It must be possible to compare the manually generated compartments with the automatically created ones. From an overview of the figure 6.5, we can almost obtain a cluster that is similar to the compartment depicted in the figure 6.1. The automatically made hulls generally delineate the shape of the compartments. However these clusters do not give a good correspondence to the manually made compartments. It is detectable that there are several holes within the largest hull. It could be resulted from that the cells inside the holes are more densely distributed than the ones outside the holes. Hence, these holes cannot be integrated into the largest hull. Otherwise, the largest hull is arbitrarily shaped. It could be highly possible that the arbitrary shape tends to be more realistic to the actual compartment.

7. Case study V: Investigating immune cell distribution

7.1 Introduction

The dynamic changes are necessary to be considered in order to analyze how the cells are distributed before and after inflammations. Basically, for instance, when an infection occurs, the immune cells would accumulate to eliminate the viruses. The clusters of immune cells are then formed. The whole process is time-dependent and the spatial distribution of immune cells would change. However, the point-like data only provides us with the static position of cells. Our analysis is restricted to detect the distribution at a certain time.

The Medetect proposed a hypothesis about the immune cell distribution, which can be summarized as below:

1. Does the distribution or pattern of the immune cells B220 and IL33 differ in the area with viruses compared to the area without viruses?

According to the hypothesis mentioned above, we here evaluate the hypothesis by two studies, the neighborhood analysis and the circumscribed hull analysis. Both of these analyses are conducted in a quantitative way. In the neighborhood analysis, we introduce two indexes and calculate the percentage of the neighboring immune cells to the viruses. In addition, for the sake of evaluating the result, we set up a statistic analysis for the indexes. As it has been investigated that the hull generation algorithm is able to make a relatively good result, hereby the hull generation method is utilized to obtain the boundary of the viruses. The number of the immune cells within and outside the boundary is then calculated.

7.2 Data

The data were obtained from a mouse lung which was infected by viruses. An overview of the data distribution is presented in figure 7.1. Here different types of cells are rendered with different symbols.

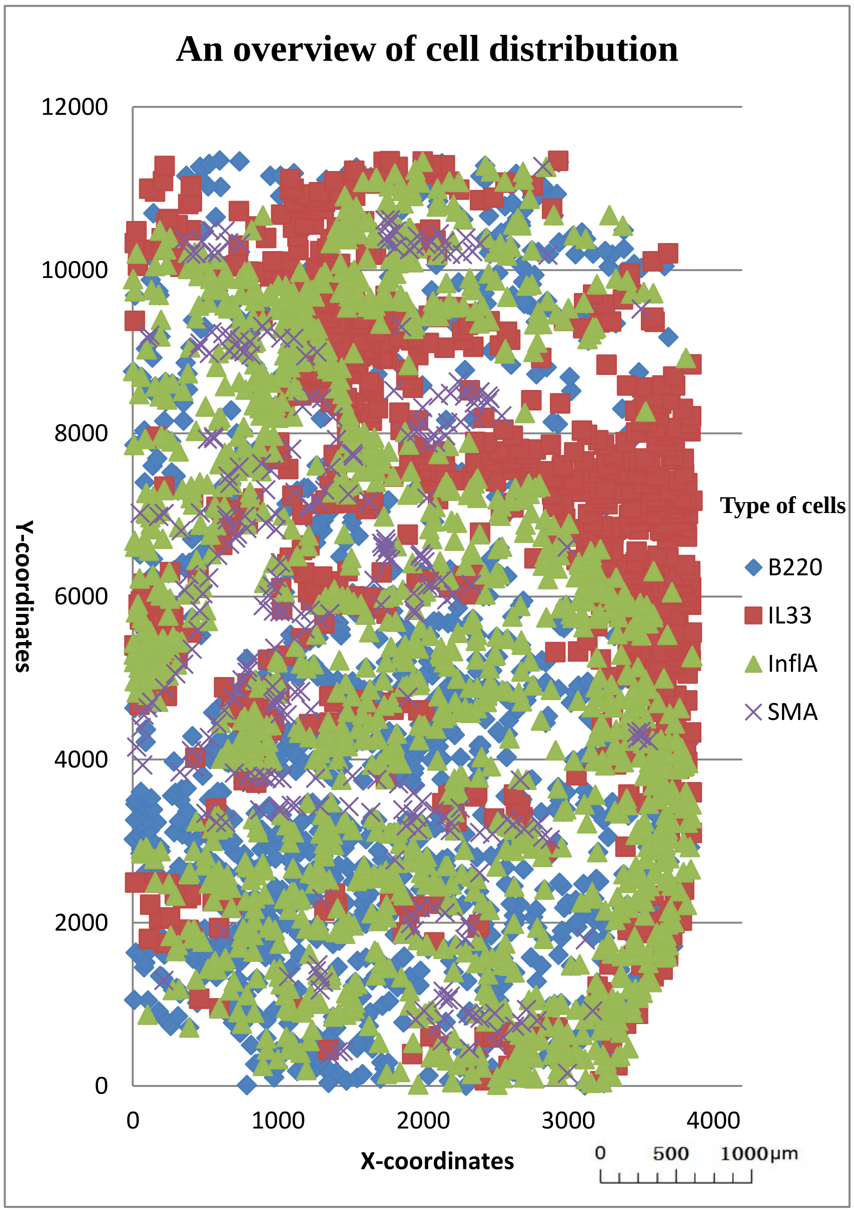


Figure 7.1 An overview of cell distribution. B220 is the immune cell and IL33 is another type of cell. InflA stands for viruses and SMA is a marker of a "place". The scale bar of the figure is only approximate.

7.3 Neighborhood analysis

7.3.1 Method

In the neighborhood analysis, we calculate the number of the neighboring immune cells to the viruses and divide them by the total number of the neighboring B220 and IL33 cells. This percentage is what we introduce below, the nearest neighbor index. The overall index is provided here in equation 5. In the calculation of the nearest neighbor index, we separately calculate the indexes for the nearest neighbor and three close neighbors.

$$Overall\ index = n_A/(n_A + n_B) \quad (5)$$

$$Nearest\ neighbor\ index = n_{\bar{A}}/(n_{\bar{A}} + n_{\bar{B}}) \quad (6)$$

where n_A and n_B mean the number of cell type A and type *B*, $n_{\bar{A}}$ and $n_{\bar{B}}$ are the number of both types of cells that are close to the viruses. Here cell type A is B220, and cell type B is IL33.

At the end of this study, we utilize a statistical analysis to evaluate what we acquired from the above indexes. A null hypothesis should be made beforehand, which is that B220 cells tend to be closer to the viruses. A binomial test is implemented by evaluating the overall index and the nearest neighbor index in Matlab. There are three input parameters that need to be specified in the function *binofit*, the number of success and trials and the confidence level.

7.3.2 Result

The tabular result below presents the calculation of the two types of index mentioned in section 7.4. This can be regarded as a quantitative way to describe the density of cells.

Table 7.1 The calculation of two indexes

Type	Value	
Overall	0.6314	
Nearest neighbor	1 closest neighbor	0.6640
	3 close neighbors	0.6647

We choose 95 percent as our confidence level and the probability of the occurrence of B220 cells ranges from 0.6189 and 0.6685 at this confidence level. Evidently the nearest neighbor index is within this interval. Hence it is 95 percent sure that the null hypothesis is denied and B220 cells do not tend to be closer to the viruses.

7.3.3 Discussion

The overall index in table 7.1 illustrates the percentage of B220 cells quantitatively. It is clear that the number of B220 cells is larger than that of IL33 cells. Although the nearest neighbor indexes are larger than the overall index, it is not able to tell if the distribution of B220 cells differs in these situations.

In the binomial test, we investigate whether the distribution of B220 cells is denser to the viruses. The overall index is regarded as the probability for the occurrence of B220 cells. Based on the confidence level we chose, the probability of B220 cells should be between 0.6189 and 0.6685. In this sense, the null hypothesis is not accepted. The distribution of B220 cells does not densely approach the viruses.

7.3 Circumscribed hull analysis

7.3.1 Method

The basic idea is to delineate the distribution of the viruses and investigate the discrepancy of distribution of the immune cells based on the delineation we made on the viruses. The clustering method and the hull generation method are utilized in delineating the spatial distribution of cells. Additionally, a quantitative analysis is conducted to investigate the difference of the immune cell distribution.

A couple of requirements should be fulfilled with regard to the selection of methods. The method should consist of the following properties: (1) the cluster method should be computationally efficient due to the amount of data, (2) the cluster method should be able to identify clusters with arbitrary shapes, and (3) the method should be insensitive to noises and outliers.

In chapter 3, we evaluate the clustering methods. As stated there, it is necessary to define the number of clusters in both partitioning clustering and hierarchical clustering. The model based method results in ellipsoidal and overlapped clusters. Hence, the DBSCAN would be more proper since it meets all the requirements.

DBSCAN enables that the spatial distribution of cells can be detected with noises. Clusters are defined with a specific density and are able to tell us how these objects are dispersed from a global perspective. On the other hand, in order to investigate the different distribution of cells with and without the viruses, the hull generation method could assist in roughly constructing an area of viruses based on the clusters. The number of the immune cells inside and outside the virus area is then calculated.

7.3.2 Result

The clusters of viruses and the hulls of clusters are presented in figure 7.2.

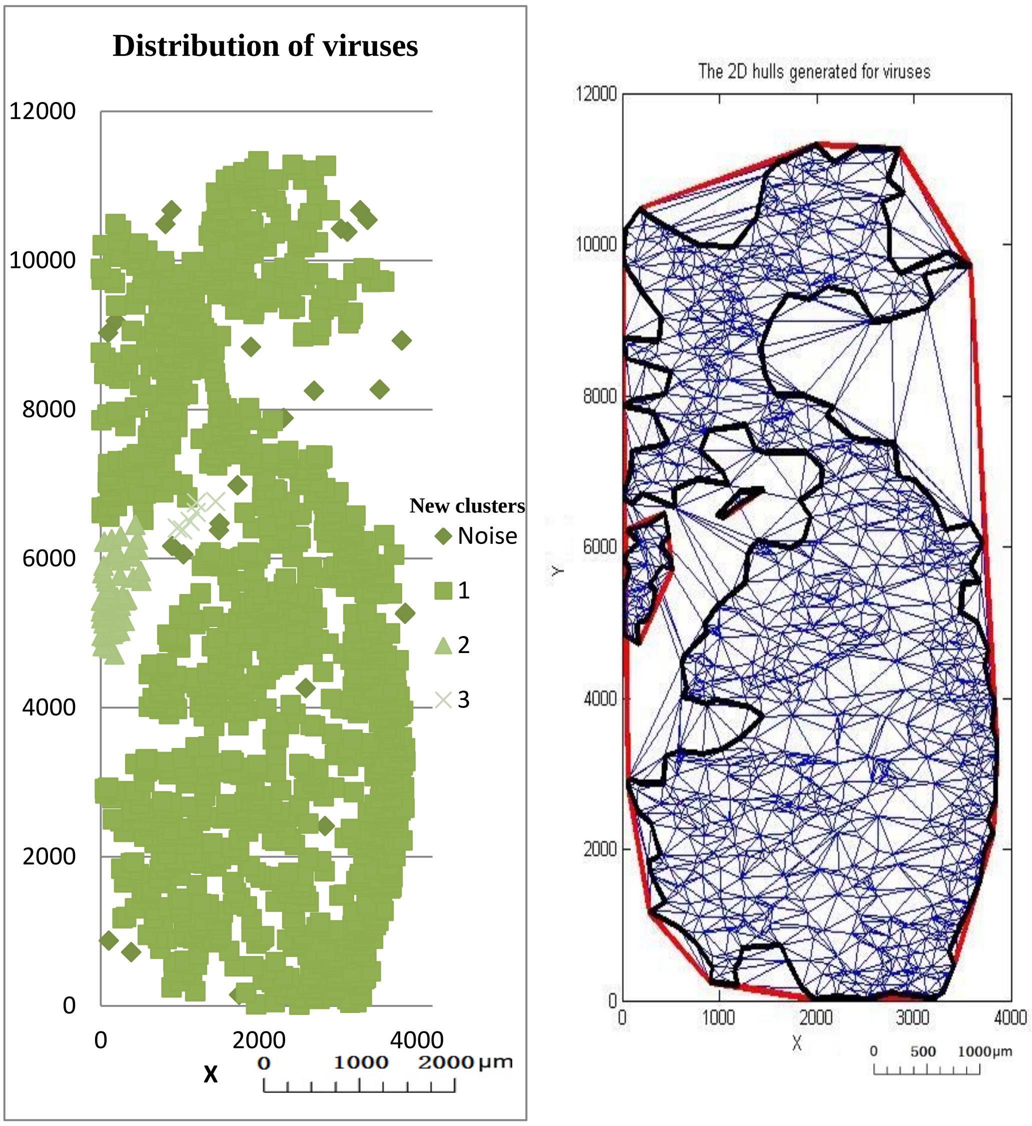


Figure 7.2 The clusters of viruses and the hulls of each cluster (the thick black lines). The scale bar of the figure is only approximate.

The number of cells (B220 and IL33) is recorded in the table 7.2.

Table 7.2 The number of B220 and IL33 inside and outside the hulls respectively

Type	Inside	Outside
B220	1255 (79.89%)	316 (20.11%)
IL33	491 (53.54%)	426(46.46%)

7.3.3 Discussion

For the purpose of testing the hypothesis, we perform the clustering method on the viruses and generate the circumscribed hull for these clusters. Hence it becomes distinguishable to investigate the areas with and without the viruses. From table 7.2, over three quarter of B220 cells are within the hull, which indicates most of B220 cells are adjacent to the viruses. The number of IL33 cells existing inside and outside the hulls does not differ a lot. However, unlike the test 1, the quantitative analyses in this part are only carried out on the individual type of cells. It is difficult to tell which kind of cells is more “affectionate” to the viruses.

8. Discussion

In this project we have conducted several case studies with spatial analysis approaches, namely, clustering methods and proximity methods. In addition, the development of the algorithm for constructing a 2D hull is helpful for studying the shape of our point-like data.

8.1 Spatial analysis methods

8.1.1 Clustering methods

As described in the sections 3.1 and 3.2.2, we present both a theoretical background of the clustering methods and their implementations in R. Each clustering type contains a couple of methods. Here we are about to emphasize on the clustering methods used in the case study I.

The K-means method has to define the number of clusters beforehand. The determination of this number is dependent on the characteristic of the data (e.g. the sum of within group error). Afterwards, the K-means algorithm randomly places the cluster centers which could result in different clusters.

In Ward's hierarchical method, it is not necessary to define an expected number of clusters. A pre-process is to create a distance matrix to tell the similarities and dissimilarities, which is computationally complex. Then the clusters are formed by grouping the objects with the same similarities.

When it comes to the DBSCAN clustering, the key problem lies in determining those two parameters with the *k-dist* plot which is made by the discrete points. The specific values of k correspond to the values of *MinPts*. As we determine the value of k, the *k-dist* plot presents all the k^{th} neighboring distances. The *MinPts* is decided by investigating the turning point of the plot. However, as shown in figure 3.12, it is necessary to simulate the trend by fitting a curve on the plot. As a matter of fact, finding the proper function is difficult because of two factors. One factor is that we cannot utilize a continuous function fitting these discrete points. Instead a piecewise function could be found to fit this trend. And the other one is that the neighboring distances are increasing exponentially, hence the turning point does not actually exist.

Even though the model-based methods could produce a better clustering result, it is highly dependent on the adoption of models. However, we hereby did not specify any models in the experiment; and instead the selection of models is automatically done by the function.

Taking a panoramic view of all the clustering methods discussed above, each type of the methods has its advantages and disadvantages. It is important to judge which method is applicable according to the characteristics of data. In our case studies, DBSCAN is the priority both in detecting the outliers and in obtaining arbitrarily shaped clusters.

8.1.2 Proximity methods

In the case study for proximity analysis, we mainly compare with two methods, the closeness method and the combined closeness and Delaunay triangulation analysis. The closeness method just uses the Euclidean distance as the metric to specify the degree of adjacency. The combined method makes the procedure more complicate since the analysis is based on the Delaunay triangulation network. It is noteworthy that the distances in the closeness method are not calculated from the viruses to their neighbors because some immune cells that are far away from the viruses are not regarded as neighbors.

8.1.3 Algorithm for constructing a 2D hull

Although our proposed method succeeds in constructing the hull for the data, problems still exist that the determination of the threshold is based on trials. The data were processed by using the DBSCAN clustering. Hence the threshold should be at least larger than the *Eps* adopted for the clustering, or no replacement of edges would be taken. If the threshold is set larger, it is meaningless to delineate the border of the data. In this case, we investigate the possible values of this threshold from two aspects. As we have the minimum value of the threshold, we evaluate the larger values. Then the edges of this hull are checked if they are all smaller than this threshold. If some edges are incorrect (still larger than the threshold), the threshold must be increased. Meanwhile, a histogram for all the edges in the Delaunay triangles is made. In our study (where *Eps* was 280) an investigation shows that 90% of the edges are within the length interval between 0 and 428, which could alternatively explain that 428 is selected as the threshold. In fact, when the value goes up to 400, all the edges of the hull are correct.

Regarding the holes inside the hull, they are constituted by the triangles. The different combination of triangles would result in different shapes of the holes. That is why the Graham's scanning method is more suitable for delineating the boundary of the holes.

9. Conclusions

This project aims at exploring the spatial patterns of cells in tissue sections on a microscopic level. With the state of art technologies, cells are treated as point like data. In the study we investigated if there is any clustering phenomenon or any other interactions existing in the cells.

The spatial analyses are carried out from two aspects, clustering analysis and proximity analysis. And in each analysis, we introduce a couple of methods that are then compared. The selection of a proper analytical method depends on both the characteristics of the data and the aim of the study.

From the comparisons of the clustering methods, DBSCAN is more appropriate for our analysis. K-means and Ward's hierarchical clustering are not able to detect noises and the shape of their clusters tends to be ellipsoidal. Even though the model-based method provides a large number of models that enables a relatively complicate analysis, it makes difficult that we need to choose a suitable model. Additionally no noises can be found in the model-based method either.

The methods in proximity analysis show the different results via two ways, a simple distance analysis and a network analysis on Delaunay triangulation. The different determinations of searching radius result in different consequences. If the definition of the neighbors could be specified in the closeness method in the further study, we would obtain a more comparable result.

Although the algorithm for constructing the 2D hull is able to generate visually good results, there are also some problems with regard to determining the threshold and forming the holes. When dealing with the realistic data, the outcomes are not capable of perfectly corresponding to the actual situation. An improvement could be made to precisely determine the threshold so that the hull will fit the used data better. Otherwise, the problems occurred in delineating the holes should be treated individually.

The methods have been applied in the case study V on real data provided by Medetect. The neighborhood analysis indicates that B220 cells are not closer to the viruses than IL33 cells. The circumscribed hull analysis reveals that the distribution of immune cell type B220 differs in the areas with viruses from the areas without viruses. However, the pattern of the immune cell IL33 does not make it different in the areas with and without viruses.

To summarize, we applied the spatial analysis methodology in tackling medical problems. It confirms that everything is related to each other. For the further work, there could be some advanced methods that are developed based on the methods we used. For instance, a trial can be made to utilize the varied density-based clustering method.

Reference

Ali, S. and Madani, S. 2011. Distributed grid based robust clustering protocol for mobile sensor networks. *The International Arab Journal of Information Technology*, Vol.8, No.4.

Arnau, V and Marin, I. 2003. A hierarchical clustering strategy and its application to proteomic interaction data. *F.J. Perales et al. (Eds.): IbPRIA* 2003, LNCS 2652, pp.62-69.

Arya S, Mount D. 2005. *Handbook of Data Structures and Applications*, 63-1.

Boley, D., Gini, M., Gross, R., Han, E.H., Hastings, K., Karypis, G., Kumar, V., Mobasher, B., Moore, J. 1999. Partitioning-based clustering for web document categorization. *Decision Support System*, pp.329-341.

Becker, R. A., Chambers, J. M., and Wilks, A. R. 1988. *The New S Language*, London, UK: Chapman & Hall.

Beckmann, N., Kriegel, H.-P., Schneider, R., Seeger, B. 1990. The R*-tree: An efficient and robust access method for points and rectangles. *Proc.1990 ACM-SIGMOD Int. Conf. Management of Data (SIG-MOD'90)*, pp.322-331.

Berchtold, S., Keim, D., Kriegel, H.-P. 1996. The X-tree: an efficient and robust access method for points and rectangles. *Proc.1996 Int. Conf. Very Large Data Bases (VLDB'96)*, pp.28-39.

Cadez, I., Heckerman, D., Meek, C., Smyth, P., White, S. 2000. Visualization of navigation patterns on a web site using model based clustering. *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp.280-284.

Chambers, J. M. 1998. Programming with Data, New York, USA: Springer.

Chambers, J. M. and Hastie, T. J. 1992. *Statistical Models in S*, London. UK: Chapman & Hall.

Chung, S., Mcleod, D. 2005. Dynamic pattern mining: an incremental data clustering approach. *Lecture Notes in Computer Science*, vol. 3360, pp.85-112.

Cluster analysis. 2012. Quick-R. Retrieved April 2014, from http://www.statmethods.net/advstats/cluster.html.

Convex hull in 2D. 2013. Fudan education home page. Retrieved August 2014, from http://www.tcs.fudan.edu.cn/rudolf/Courses/Algorithms/Alg_cs_07w/Webprojects/Zhaobo_hull/index.html

Cormack, R. M. 1971. A review of classification. *Journal of the Royal Statistical Society*, A 134: 321-367.

CRAN (2014-1-29). "fpc: Flexible procedures for clustering" from http://cran.r-project.org/web/packages/fpc/index.html.

Everitt, B.S., Landau, S., Leese, M., Stahl, D. 2011. *Cluster Analysis*, 5th edition. John Wiley & Sons.

Ester, M., Kriegel, H.P., Sander, J., Xu, X.W. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining*. KDD.96.

Ferreira, L. and Hitchcock, D.B. 2009. A comparison of hierarchical methods for clustering functional data. *Communications in Statistics-Simulation and Computation*, vol.38, pp.1925-1949.

Fisher, M. and Springborn, B. and Schröder, P. and Bobenko, A. I. 2007. An algorithm for the construction of intrinsic Delaunay triangulations with applications to digital geometry processing. *Computing*, 81 (2-3). pp. 199-213.

Frigui, H., Rhouma, M. 2001. Self-organization of Pulse-coupled Oscillators with Application to Clustering, *Transactions on Pattern Analysis and Machine Intelligence*, pp.180-195.

Gaonkar, M.N. and Sawant, K. 2013. AutoEpsDBSCAN: DBSCAN with Eps automatic for large dataset. ISSN:2319-2526, vol.2.

GNU operating system. 2014. GNU general public license. Retrieved 2014, from https://www.gnu.org/copyleft/gpl.html.

Gordon, A. D. 1999. Classification (2nd edn). Chapmen and Hall/CRC. Boca Raton, FL.

Gottwald, T. R., Avinent, L., Llácer, G., de Mendoza, A. H., Cambra, M. 1995. Analysis of the spatial spread of sharka (plum pox virus) in apricot and peach orchards in eastern Spain. *Plant Disease*, 79(3), pp.266-278.

Guha, S., Rastogi, R., Shim, K. 1998. CURE: an efficient clustering algorithm for large databases. *Proc.1998 ACM-SIGMOD Int. Conf. Management of Data (SIG-MOD'98)*, pp.73-84.

Han, J.W., Kamber, M., Tung, A.K.H. 2001. Spatial clustering methods in data mining: a survey. *Geographic Data Mining and Knowledge Discovery, Research*

Monographs in GIS.

Handcock, M.S., Raftery, A.E., Tantrum, J.M. 2007. Model-based clustering for social networks. *J.R.Statist.Soc.A*, Part 2, pp.301-354.

Introduction to ArcGIS. 2014. ArcGIS Resources. Retrieved 7 May 2014, from http://resources.arcgis.com/en/help/getting-started/articles/026n00000014000000.html.

Jain, A. K. 2010. Data clustering: 50 years beyond K-means. Pattern Recognition Letters, pp.651-656.

Joubran, J. and Gabay, Y. 2000. A method for construction of 2D hull for generalized cartographic representation. *International Archives of Photogrammetry and Remote Sensing*. vol. XXXIII, part B4.

Kaufman, L. and Rousseeuw, P.J. 1990. Finding groconstruups in data: an introduction to cluster analysis. John Wiley & Sons.

Korolev, A. V., Tomos, A. D., Bowtell, R., Farrar, J. F. 2000. Spatial and temporal distribution of solutes in the developing carrot taproot measured at single‐cell resolution. *Journal of Experimental Botany*, 51(344), pp.567-577.

Lecoustre, R., Fargette, D., Fauquet, C., de Reffye, P. 1989. Analysis and mapping of the spatial spread of African cassava mosaic virus using geostatistics and the kriging technique. Phytopathology, 79(9), pp.913-920.

Liao, W.K., Liu, Y., Choudhary, A. 2004. A grid-based clustering algorithm using adaptive mesh refinement. *7th Workshop on Mining Scientific and Engineering Datasets of SIAM International Conference on Data Mining*.

Liu, P., Zhou, D., Wu, N.J. 2007. VDBSCAN: varied density based spatial clustering of applications with noise.

Madden, L. V., Pirone, T. P., Raccah, B. 1987. Analysis of spatial patterns of virus-diseased tobacco plants. *Phytopathology*, 77(10), pp.1409-1417.

Maguire, D.J., Batty, M., Goodchild, M.F. 2005. GIS, spatial analysis and modeling, Esri Press.

Near anlysis. 2014. ArcGIS Resource. Retrieved 7 May 2014, from http://resources.arcgis.com/en/help/main/10.1/index.html#//00080000001q0

00000.

Pellegrini, T., Portelo, J., Trancoso, I., Abad, A., Bugalho, M. 2009. Hierarchical clustering experiments for application to audio event detection. P*roceedings of the 13th International Conference on Speech and Computer*.

Quan, Q., Xiao, C.J., Zhang, R. 2013. Grid-based data stream clustering for intrusion detection. I*nternational Journal of Network Security*, vol.15, no.1, pp. 1-8.

Roix, J. J., McQueen, P. G., Munson, P. J., Parada, L. A., & Misteli, T. 2003. Spatial proximity of translocation-prone gene loci in human lymphomas. *Nature Genetics*, 34(3), 287-291.

Sander, J., Ester, M., Kriegel, H.P., Xu, X.W. 1998. Density-based clustering in spatial databases: the algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery 2*, pp.169-194.

Scitovski, R. and Scitovski, S. 2013. A fast portioning algorithm and its application to earthquake investigation. *Computer & Geosciences*, pp.124-131.

Tramancere, A. and Vecchio, C. 2013. γ-ray DBSCAN: a clustering algorithm applied for Fermi-LAT γ-ray data. *Astronomy & Astrophysics 549*, A138.

Wu, H.G., Lin, Q., Jin, B.H., Liu, Q. 2012. Application of improved DBSCAN algorithm in the plan compilation management. *International Journal of Database Theory and Application*, vol.5, no.4.

Xu, X., Ester, M., Kriegel, H., Sander, J. 1998. A Distribution-based Clustering Algorithm for Mining in Large Spatial Databases, *In 14th International Conference on Data Engineering, Computer Society Press*, pp.324-331.

Zhang, T., Ramakrishnan, R., Livny, M. 1996. BIRCH: an efficient data clustering method for very large databases. *Proc.1996 ACM-SIGMOD Int. Conf. Management of Data (SIG-MOD'96)*, pp.103-114.

Appendix A

Below the detailed commands in R used to experiment with are given, ton_data is the data to be processed.

A.1 K-means algorithm

```
kmeans(x, centers, iter.max = 10, nstart = 1,
          algorithm = c("Hartigan-Wong", "Lloyd", "Forgy",
                              "MacQueen"))
```

x	**numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns).**
centers	**either the number of clusters, say k, or a set of initial (distinct) cluster centres. If a number, a random set of (distinct) rows in x is chosen as the initial centres.**
iter.max	**the maximum number of iterations allowed.**
nstart	**if centers is a number, how many random sets should be chosen?**
algorithm	**character: may be abbreviated.**

```
# Determine number of clusters
wss <- (nrow(ton_data)-1)*sum(apply(ton_data,2,var))
for (i in 1:15) wss[i] <- sum(kmeans(ton_data,centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters",ylab="Within groups sum of
squares")

# K-Means Cluster Analysis
fit <- kmeans(ton_data, 7) # 7 cluster solution
# get cluster means
aggregate(ton_data,by=list(fit$cluster),FUN=mean)
# append cluster assignment
ep_data <- data.frame(ton_data, fit$cluster)
write.csv(ep_data,"data_fit")
```

A.2 Hierarchical clustering

```
hclust(d, method = "complete", members=NULL)
```

d	**a dissimilarity structure as produced by dist.**
method	**the agglomeration method to be used. This should be (an unambiguous abbreviation of) one**

of"ward", "single", "complete", "average", "mcquitty", "median" or "centroid".

members NULL or a vector with length size of d.

```
# Ward Hierarchical Clustering
d <- dist(ton_data, method = "euclidean") # distance matrix
fit1 <- hclust(d, method="ward")
plot(fit1) # display dendogram
groups <- cutree(fit1, k=7) # cut tree into 7 clusters
# draw dendogram with red borders around the 7 clusters
rect.hclust(fit1, k=7, border="red")
ep_data1<-data.frame(ton_data,groups)
write.csv(ep_data1,"data_fit1")
```

A.3 DBSCAN

```
dbscan(data, eps, MinPts = 5, scale = FALSE, method = c("hybrid", "raw",
    "dist"), seeds = TRUE, showplot = FALSE, countmode = NULL)
```

data	**data matrix, data.frame, dissimilarity matrix or dist-object. Specify method="dist" if the data should be interpreted as dissimilarity matrix or object. Otherwise Euclidean distances will be used.**
eps	**Reachability distance.**
MinPts	**Reachability minimum no. of points.**
scale	**scale the data if TRUE.**
method	**"dist" treats data as distance matrix (relatively fast but memory expensive), "raw" treats data as raw data and avoids calculating a distance matrix (saves memory but may be slow), "hybrid" expects also raw data, but calculates partial distance matrices (very fast with moderate memory requirements).**
seeds	**FALSE to not include the isseed-vector in the dbscan-object.**

```
#DBSCAN
fit3<-dbscan(ton_data,55,MinPts=5,method="hybrid",seeds=TRUE,showplot=TRUE)
ep_data3<-data.frame(ton_data,fit3$cluster)
write.csv(ep_data3,"data_fit3")

fit4<-dbscan(ton_data,60,MinPts=7,method="hybrid",seeds=TRUE,showplot=TRUE)
ep_data4<-data.frame(ton_data,fit4$cluster)
write.csv(ep_data4,"data_fit4")
```

A.4 Model-based clustering

```
Mclust(data, G=NULL, modelNames=NULL, prior=NULL, control=emControl(),
        initialization=NULL, warn=FALSE, ...)
```

data	**A numeric vector, matrix, or data frame of observations. Categorical variables are not allowed. If a matrix or data frame, rows correspond to observations and columns correspond to variables.**
G	**An integer vector specifying the numbers of mixture components (clusters) for which the BIC is to be calculated. The default is G=1:9.**
modelNames	**A vector of character strings indicating the models to be fitted in the EM phase of clustering.**

```
fit2 <- Mclust(ton_data)
plot(fit2) # plot results
summary(fit2,parameters=TRUE) # display the best model
ep_data2<-data.frame(ton_data,fit2$classification)
write.csv(ep_data2,"data_fit2")
```

A.5 Delaunay triangulation

```
delaunayn(p, options = "", full = FALSE)
```

p	**p is an n-by-dim matrix. The rows of p represent n points in dim-dimensional space.**
options	**String containing extra options for the underlying Qhull command.**
full	**Return all information asscoiated with triangulation as a list. At present this is the triangulation (tri) and a list of neighbours of each facet (neighbours).**

A.6 The script for *k-dist* plot

```
% k-dist plot
data=xlsread('E:\thesis\practical part\ton_epitel.xlsx');
m=length(data);
% create a m*m matrix with 0
dist_matrix=zeros(m,m);
% calculate distances between each point to other points and store them in
% the dist_matrix
for i=1:m
    for j=1:m
        dist=sqrt((data(i,1)-data(j,1))^2+(data(i,2)-data(j,2))^2);
          dist_matrix(i,j)=dist;
    end
```

```
end

sort_matrix=zeros(m,m);
for i=1:m
        sort_matrix(:,i)=sort(dist_matrix(:,i));
end

% the values of k range from 4 to 8
p=1:1323;
q1=sort(sort_matrix(4,:));
q2=sort(sort_matrix(5,:));
q3=sort(sort_matrix(6,:));
q4=sort(sort_matrix(7,:));
q5=sort(sort_matrix(8,:));

%plot k-dist with different values of k
plot(p,q1,'r*');
hold on
plot(p,q2,'b*');
hold on
plot(p,q3,'y*');
hold on
plot(p,q4,'r-');
hold on
plot(p,q5,'b-');
```

A.7 The script for *Delaunay triangulation*

```
library("base")
library("geometry")

setwd("E:\\thesis\\proximity")
data<-read.csv("tonsill.csv")
Tri<-delaunayn(data)
dim<-dim(Tri)
rows<-dim[1]
tri_edge=matrix(0,rows,3)
for (i in 1:dim[1]) {
    a<-Tri[i,1]
    b<-Tri[i,2]
    c<-Tri[i,3]
    tri_edge[i,1]=sqrt((data[a,1]-data[b,1])^2+(data[a,2]-data[b,2])^2)
    tri_edge[i,2]=sqrt((data[a,1]-data[c,1])^2+(data[a,2]-data[c,2])^2)
    tri_edge[i,3]=sqrt((data[b,1]-data[c,1])^2+(data[b,2]-data[c,2])^2)
}

d<-matrix(tri_edge,dim[1]*3,1)
hist(d,nclass=15)

g<-0
for (i in 1:dim[1]) {
    if (tri_edge[i,1]>90 |tri_edge[i,2]>90 |tri_edge[i,3]>90){
    g<-c(g,i)
    }
}
Tri<-Tri[-g[2:length(g)],]
data<-as.matrix(data)
trimesh(Tri,data)

dim2<-dim(Tri)
Eds=matrix(0,dim2[1]*3,2)
for (i in 1:dim2[1]) {
    Eds[3*i-2,1]=Tri[i,1]
    Eds[3*i-2,2]=Tri[i,2]
    Eds[3*i-1,1]=Tri[i,1]
    Eds[3*i-1,2]=Tri[i,3]
    Eds[3*i,1]=Tri[i,2]
    Eds[3*i,2]=Tri[i,3]
}
Eds<-Eds[order(Eds[1:6879,1]),]
virus<-data[1318:1323,]
for (i in 1:length(virus)) {
    Eds[Eds[,1]==virus[i],]
    Eds[Eds[,2]==virus[i],]
}
```

A.8 The script for constructing the 2D hull

```
%This algorithm generates the 2D hulls of the clusters.
%The input data include the coordinates of points and the cluster number of
%each point.
%This script was written by Ruibin Xu in August, 2014.
%==========================================================
================
%import the data
data=xlsread('E:\thesis\new_data\ts_virus.xlsx');
ncluster=max(data(:,3));
%plot the points
xcoor=data(:,1);
ycoor=data(:,2);
plot(xcoor,ycoor,'r.')
hold on
%generate the hull for each cluster
for id=1:ncluster
  cluster=data(data(:,3)==id,:);
  Tri=delaunay(cluster(:,1:2));
  length=size(Tri,1);
%create an empty matrix to store the length of edges
  tri_edge=zeros(size(Tri));
%calculate the length of edges of each triangle
  for i=1:length
      a=Tri(i,1);
      b=Tri(i,2);
      c=Tri(i,3);
      tri_edge(i,1)=sqrt((cluster(a,1)-cluster(b,1))^2+(cluster(a,2)-cluster(b,2))^2);
      tri_edge(i,2)=sqrt((cluster(a,1)-cluster(c,1))^2+(cluster(a,2)-cluster(c,2))^2);
      tri_edge(i,3)=sqrt((cluster(b,1)-cluster(c,1))^2+(cluster(b,2)-cluster(c,2))^2);
  end
%combine the data of all columns into one column
  dim=size(tri_edge);
  d=reshape(tri_edge,dim(1)*dim(2),1);
%choose a proper threshold
%plot a histogram for the edges
  fre=0;
  threshold=0;
  [e,f]=hist(d,15);
  for i=1:size(e,2)
        if fre/sum(e)<0.9
                fre=fre+e(i);
                threshold=f(i)+max(d)/30;
        end
  end

%   triplot(Tri,cluster(:,1),cluster(:,2),'b-');
%   hold on
```

```
  vertice1=convhull(cluster(:,1:2));

%   plot(cluster(vertice1,1),cluster(vertice1,2),'r-', 'LineWidth',3);
%   hold on
%calculate the convex hull edges
  ch_edge=zeros(size(vertice1,1)-1,1);
  for i=2:size(vertice1,1)
        e=vertice1(i);
        f=vertice1(i-1);
        ch_edge(i-1)=sqrt((cluster(e,1)-cluster(f,1))^2+(cluster(e,2)-cluster(f,2))^2);
  end

  vertice1=vertice1';
  ch_edge=ch_edge';
  p3_pre=0;

  for i=1:100000
    if i<=size(ch_edge,2)
      while ch_edge(i)>threshold
              [p,q]=find(Tri==vertice1(i));
              [m,n]=find(Tri(p,:)==vertice1(i+1));


              for j=1:size(m,1)
                     index=6-q(m(j))-n(j);
                     p1=Tri(p(m(j)),index);

                   if ismember(p1,vertice1)==0
                          switch index
                                 case 1
                                        vertice1=[vertice1(1:i),p1,vertice1(i+1:end)];

ch_edge=[ch_edge(1:i-1~=0),tri_edge(p(m(j)),1),tri_edge(p(m(j)),2),ch_edge(i+1:end
)];
                                 case 2
                                        vertice1=[vertice1(1:i),p1,vertice1(i+1:end)];

ch_edge=[ch_edge(1:i-1~=0),tri_edge(p(m(j)),3),tri_edge(p(m(j)),1),ch_edge(i+1:end
)];
                                 case 3
                                        vertice1=[vertice1(1:i),p1,vertice1(i+1:end)];
```

```
ch_edge=[ch_edge(1:i-1~=0),tri_edge(p(m(j)),2),tri_edge(p(m(j)),3),ch_edge(i+1:end
)];
                              end
                      end
                end
                if p3_pre==p1
                          break;
                end
                    p3_pre=p1;
      end
    end
  end

plot(cluster(vertice1,1),cluster(vertice1,2),'b-','LineWidth',3);
hold on
end
```

A.9 The script for delineating the holes

```
data=xlsread('E:\thesis\new_data\ts_virus.xlsx');
%cluster1
cluster1=data(data(:,3)==1,:);
Tri1=delaunay(cluster1(:,1:2));
%create an empty matrix to store the length of edges
tri_edge1=zeros(size(Tri1));
%calculate the length of edges of each triangle
for i=1:length(Tri1)
        a=Tri1(i,1);
        b=Tri1(i,2);
        c=Tri1(i,3);

tri_edge1(i,1)=sqrt((cluster1(a,1)-cluster1(b,1))^2+(cluster1(a,2)-cluster1(b,2))^2);

tri_edge1(i,2)=sqrt((cluster1(a,1)-cluster1(c,1))^2+(cluster1(a,2)-cluster1(c,2))^2);

tri_edge1(i,3)=sqrt((cluster1(b,1)-cluster1(c,1))^2+(cluster1(b,2)-cluster1(c,2))^2);
end

dim=size(tri_edge1);
d1=reshape(tri_edge1,dim(1)*dim(2),1);
%[e,f]=hist(d1,15);
```

```
triplot(Tri1,cluster1(:,1),cluster1(:,2),'b-');
hold on

vertice1=convhull(cluster1(:,1:2));

%calculate the convex hull edges
ch_edge1=zeros(length(vertice1)-1,1);
for i=2:length(vertice1)
        e=vertice1(i);
        f=vertice1(i-1);

ch_edge1(i-1)=sqrt((cluster1(e,1)-cluster1(f,1))^2+(cluster1(e,2)-cluster1(f,2))^2);
end

vertice1=vertice1';
ch_edge1=ch_edge1';
p3_pre=0;

for i=1:100000
  if i<=length(ch_edge1)
    while ch_edge1(i)>428
                [p,q]=find(Tri1==vertice1(i));
                [m,n]=find(Tri1(p,:)==vertice1(i+1));


                for j=1:length(m)
                        index=6-q(m(j))-n(j);
                        p1=Tri1(p(m(j)),index);

                      if ismember(p1,vertice1)==0
                              switch index
                                      case 1
                                              vertice1=[vertice1(1:i),p1,vertice1(i+1:end)];

ch_edge1=[ch_edge1(1:i-1~=0),tri_edge1(p(m(j)),1),tri_edge1(p(m(j)),2),ch_edge1(i
+1:end)];
                                      case 2
                                              vertice1=[vertice1(1:i),p1,vertice1(i+1:end)];

ch_edge1=[ch_edge1(1:i-1~=0),tri_edge1(p(m(j)),3),tri_edge1(p(m(j)),1),ch_edge1(i
+1:end)];
```

```
                    case 3
                        vertice1=[vertice1(1:i),p1,vertice1(i+1:end)];

ch_edge1=[ch_edge1(1:i-1~=0),tri_edge1(p(m(j)),2),tri_edge1(p(m(j)),3),ch_edge1(i
+1:end)];
                end
            end
        end
        if p3_pre==p1
            break;
        end
          p3_pre=p1;
  end
 end
end

plot(cluster1(vertice1,1),cluster1(vertice1,2),'black-', 'LineWidth',4);
hold on


g=0;
for i=1:length(cluster1)
    [mp,np]=find(Tri1==i);
    for j=1:length(mp)
        switch np(j)
            case 1
                edge(j,1)=tri_edge1(mp(j),1);
                edge(j,2)=tri_edge1(mp(j),2);
            case 2
                edge(j,1)=tri_edge1(mp(j),1);
                edge(j,2)=tri_edge1(mp(j),3);
            case 3
                edge(j,1)=tri_edge1(mp(j),2);
                edge(j,2)=tri_edge1(mp(j),3);
        end
        if edge(j,1)>428 && edge(j,2)>428
            g=[g,mp(j)];
        end
    end
end

co_vertex=0;
```

```
for i=2:length(g)
        vertices=Tri1(g(i),:);
        intersection=intersect(vertice1,vertices);
        if isempty(intersection)
              co_vertex=[co_vertex,g(i)];
        end
end

hole=zeros(length(co_vertex)-1,3);
for i=1:length(hole)
        hole(i,:)=Tri1(co_vertex(i+1),:);
end

%remove single triangles and rows with the same values
h=0;
for i=1:length(hole)
        [X1,Y1]=find(hole==hole(i,1));
        [X2,Y2]=find(hole==hole(i,2));
        [X3,Y3]=find(hole==hole(i,3));
        if length(X1)==1 && length(X2)==1 && length(X3)==1
                h=[h,i];
        end
end
hole(h(2:end),:)=[];
hole=unique(hole,'rows');
%merge the connected triangles
l=zeros(length(hole),10);
for i=1:length(hole)
        [X1,Y1]=find(hole==hole(i,1));
        [X2,Y2]=find(hole==hole(i,2));
        [X3,Y3]=find(hole==hole(i,3));
        triangles=unique([X1',X2',X3']);
        for j=1:length(triangles)
                l(i,j)=triangles(j);
        end
        l(i,length(triangles)+1:end)=0;
end
l=unique(l,'rows');
%plot the holes
polygon1=unique([l(1,:),l(2,:),l(5,:)]);
p_input=unique(hole(polygon1(2:end),:));
input=cluster1(p_input,1:2);
```

```
points=graham(input);
plot(points(:,1), points(:,2), 'b-','LineWidth',4);
hold on
polygon2=unique([l(3,:),l(4,:),l(6,:),l(7,:),l(8,:),l(9,:),l(10,:)]);
p_input=unique(hole(polygon2(2:end),:));
input=cluster1(p_input,1:2);
points=graham(input);
plot(points(:,1), points(:,2), 'b-','LineWidth',4);
hold on
polygon3=unique(l(11,:));
p_input=unique(hole(polygon3(2:end),:));
input=cluster1(p_input,1:2);
points=graham(input);
plot(points(:,1), points(:,2), 'b-','LineWidth',4);
hold on
polygon4=unique([l(17,:),l(18,:),l(19,:),l(20,:)]);
p_input=unique(hole(polygon4(2:end),:));
input=cluster1(p_input,1:2);
points=graham(input);
plot(points(:,1), points(:,2), 'b-','LineWidth',4);
hold on
polygon5=unique(l(16,:));
p_input=unique(hole(polygon5(2:end),:));
input=cluster1(p_input,1:2);
points=graham(input);
plot(points(:,1), points(:,2), 'b-','LineWidth',4);
hold on
polygon6=unique([l(21,:),l(22,:),l(23,:)]);
p_input=unique(hole(polygon6(2:end),:));
input=cluster1(p_input,1:2);
points=graham(input);
plot(points(:,1), points(:,2), 'b-','LineWidth',4);
hold on
polygon7=unique(l(24,:));
p_input=unique(hole(polygon7(2:end),:));
input=cluster1(p_input,1:2);
points=graham(input);
plot(points(:,1), points(:,2), 'b-','LineWidth',4);
```

A.10 The script for the Graham’s scanning

```
% This function aims at generating the hull by using Graham’s scanning algorithm.
% The input data contain the coordinates of each point and the output would export
% the sorted points through the sequence mentioned in Graham’s scanning method
% This script was written by Ruibin Xu in August, 2014.
function [points]=graham(input)
%use graham's scanning to get the polygon
%find the starting point
start_index= find(input(:,1)==min(input(:,1)));
pstart=input(start_index,:);
%sort the points
for i=1:length(input)
        Xs=pstart(1,1);
        Ys=pstart(1,2);
        Xe=input(i,1);
        Ye=input(i,2);
        input(i,3)=(Ys-Ye)/(Xs-Xe);
end
%sort the input matrix
ms=input(:,3);
[ps,pos]=sort(ms);
input(:,1:2)=input(pos,1:2);
input(:,3)=ps;

points=[input(end,1:2);input(1:end,1:2)];
end
```

Institutionen för naturgeografi och ekosystemvetenskap, Lunds Universitet.

Student examensarbete (Seminarieuppsatser). Uppsatserna finns tillgängliga på institutionens geobibliotek, Sölvegatan 12, 223 62 LUND. Serien startade 1985. Hela listan och själva uppsatserna är även tillgängliga på LUP student papers (www.nateko.lu.se/masterthesis) och via Geobiblioteket (www.geobib.lu.se)

The student thesis reports are available at the Geo-Library, Department of Physical Geography and Ecosystem Science, University of Lund, Sölvegatan 12, S-223 62 Lund, Sweden. Report series started 1985. The complete list and electronic versions are also electronic available at the LUP student papers (www.nateko.lu.se/masterthesis) and through the Geo-library (www.geobib.lu.se)

294 Mattias Spångmyr (2014) Development of an Open-Source Mobile Application for Emergency Data Collection
295 Hammad Javid (2013) Snowmelt and Runoff Assessment of Talas River Basin Using Remote Sensing Approach
296 Kirstine Skov (2014) Spatiotemporal variability in methane emission from an Arctic fen over a growing season – dynamics and driving factors
297 Sandra Persson (2014) Estimating leaf area index from satellite data in deciduous forests of southern Sweden
298 Ludvig Forslund (2014) Using digital repeat photography for monitoring the regrowth of a clear-cut area
299 Julia Jacobsson (2014) The Suitability of Using Landsat TM-5 Images for Estimating Chromophoric Dissolved Organic Matter in Subarctic Lakes
300 Johan Westin (2014) Remote sensing of deforestation along the trans-Amazonian highway
301 Sean Demet (2014) Modeling the evolution of wildfire: an analysis of short term wildfire events and their relationship to meteorological variables
302 Madelene Holmblad (2014). How does urban discharge affect a lake in a recreational area in central Sweden? – A comparison of metals in the sediments of three similar lakes
303 Sohidul Islam (2014) The effect of the freshwater-sea transition on short-term dissolved organic carbon bio-reactivity: the case of Baltic Sea river mouths
304 Mozafar Veysipanah (2014) Polynomial trends of vegetation phenology in Sahelian to equatorial Africa using remotely sensed time series from 1983 to 2005
305 Natalia Kelbus (2014) Is there new particle formation in the marine boundary layer of the North Sea?
306 Zhanzhang Cai (2014) Modelling methane emissions from Arctic tundra wetlands: effects of fractional wetland maps
307 Erica Perming (2014) Paddy and banana cultivation in Sri Lanka - A study analysing the farmers' constraints in agriculture with focus on Sooriyawewa D.S. division
308 Nazar Jameel Khalid (2014) Urban Heat Island in Erbil City.
309 Jessica, Ahlgren & Sophie Rudbäck (2014) The development of GIS-usage in developed and undeveloped countries during 2005-2014: Tendencies, problems and limitations
310 Jenny Ahlstrand (2014) En jämförelse av två riskkarteringar av fosforförlust från jordbruksmark – Utförda med Ekologgruppens enkla verktyg och

erosionsmodellen USPED
311 William Walker (2014) Planning Green Infrastructure Using Habitat Modelling. A Case Study of the Common Toad in Lomma Municipality
312 Christiana Marie Walcher (2014) Effects of methane and coastal erosion on subsea-permafrost and emissions
313 Anette Fast (2014) Konsekvenser av stigande havsnivå för ett kustsamhälle- en fallstudie av VA systemet i Beddingestrand
314 Maja Jensen (2014) Stubbrytningens klimatpåverkan. En studie av stubbrytningens kortsiktiga effekter på koldioxidbalansen i boreal barrskog
315 Emelie Norhagen (2014) Växters fenologiska svar på ett förändrat klimat - modellering av knoppsprickning för hägg, björk och asp i Skåne
316 Liisi Nõgu (2014) The effects of site preparation on carbon fluxes at two clear-cuts in southern Sweden
317 Julian Will (2014) Development of an automated matching algorithm to assess the quality of the OpenStreetMap road network - A case study in Göteborg, Sweden
318 Niklas Olén (2011) Water drainage from a Swedish waste treatment facility and the expected effect of climate change
319 Wösel Thoresen (2014) Burn the forest - Let it live. Identifying potential areas for controlled forest fires on Gotland using Geographic Information System
320 Jurgen van Tiggelen (2014) Assimilation of satellite data and in-situ data for the improvement of global radiation maps in the Netherlands.
321 Sam Khallaghi (2014) Posidonia Oceanica habitat mapping in shallow coastal waters along Losinj Island, Croatia using Geoeye-1 multispectral imagery.
322 Patrizia Vollmar (2014) The influence of climate and land cover on wildfire patterns in the conterminous United States
323 Marco Giljum (2014) Object-Based Classification of Vegetation at Stordalen Mire near Abisko by using High-Resolution Aerial Imagery
324 Marit Aalrust Ripel (2014) Natural hazards and farmers experience of climate change on highly populated Mount Elgon, Uganda
325 Benjamin Kayatz (2014) Modelling of nitrous oxide emissions from clover grass ley – wheat crop rotations in central eastern Germany - An application of DNDC
326 Maxime Rwaka (2014) An attempt to investigate the impact of 1994 Tutsi Genocide in Rwanda on Landscape using Remote Sensing and GIS analysis
327 Ruibin Xu (2014) Spatial analysis for the distribution of cells in tissue sections