Chapter I

Introduction

1.1 Background

Spatial database optimized to store and query data associated with objects in space, including points, and lines polygons[1]. Spatial database has some type of spatial query one of them is Nearest Neighbour. Nearest Neighbour is a spatial query that can estimate the nearest point of finding an object. For example, there are five post offices in the city. How do people know where the post office closest? However, Nearest Neighbour has the disadvantage that the computation results of searches take a long time. This can be remedied by using a partition space that is Voronoi diagram.

Voronoi Diagram method divides the map into smaller regions based on nearest distance to an object, A Voronoi diagram can mimic human visual intuition, whereby humans can easily identity whether an object is located inside or outside a closed shape. Voronoi diagram have many variation, there are Voronoi Diagram Order-1 and Higher Order Voronoi Diagram. These two variations have different construct and and has a deficiency that is not dynamic and possess high computational properties. Highest Order Voronoi Diagram(HSVD) have a solution for it. However Voronoi diagram also has the disadvantage that the data object to the fragmentation (fractional part Voronoi diagram) is sought can not be directly retrieved by the search data. Search fragment using Linear search whereby search squentially each polygon. However it can takes a long time to searching fragment.

Optimal indexing method can optimize or reduce the object to be inspected upon a query process. One of the most significant index structures proposed for indexing spatial data is the R-tree[4]. The main idea of the R-tree is to split the space with hierarchically nested and possibly overlapping subspaces. The hierarchical structure of the R-tree index is height balanced (i.e. all leaf nodes appear at the same level) similar to that of B-tree index.

R*tree is one A variant R-tree which has improvements and handling overflow. R*-Tree focuses on the insertion issues of R-Tree in order to improve search performance. This is achieved by forced re-insert. In R*-Tree, if a node overflows, it is not split right away. Rather p entries are removed from the node, and re-inserted into the tree (called forced re-insert). In practice, parameter p may vary. But experiments show that when p is around 30%, it achieves the best performance[3].

In this final task will be indexing the region Highest Order Voronoi Diagram using R*-tree method. By dividing partition region with MBR. Multiple MBR form a bigger MBR, hence creating a hierarchy of MBRs. The Larger MBR that contains smaller MBRs completely and properly covers all its MBRs that it contains. MBRs recursively grouped into larger MBRs to form a tree, whereby the largest MBR which cover all spatial objects continuously. By using this method it will increase performance of indexing to find a region on an object.

1.2 Problem Definitions

The following final task problem definitions.

1. How to implement R*-Tree on voronoi cell?
2. How performance result to searching fragment voronoi cell using indexing R*-tree method?

1.3 Purpose

Based on the above needs, this Final Task undertakes the following research objectives.

1. Implement R*-Tree method on voronoi cell.
2. Analyze performance result to searching fragment voronoi cell using indexing R*-Tree method.

1.4 Limitation of Problem

The following final task scope problem.

1. System using random uniform distributed dataset, object will spread to all areas voronoi diagram.
2. 2-dimension dataset modeling.
3. Performance test-parameter measured from construct data structure and search single region.
4. R*-Tree Max entry each node 4 (M=4)

1.5 Research Method

- Study of Literature

  Read up the concepts and theories related to this Final Task. The learning process materials research through libraries relating thesis either in the form of books and scientific journals

- Data Collection

  Search data required to support the resolution to this thesis

- Analysis and Design System Requirement

  Analysis and design of the system built, analyzed the methods to be used for resolve the issues, including determining the programming language used, architecture, and functionality of system.

- Result Analysis

  Analysis the result of system built, Analysis of the results will be done by looking at the performance results.

- Documentation Report

  Documentation report writing based on research that has been done.