Efficient Data Modeling and Querying System for Multi-Dimensional Spatial Data

Wei Li
University of Massachusetts Lowell
One University Ave
Lowell, MA 01854, USA
wli@cs.uml.edu

Cindy X. Chen
University of Massachusetts Lowell
One University Ave
Lowell, MA 01854, USA
cchen@cs.uml.edu

ABSTRACT
Multi-dimensional spatial data are obtained when a number of data acquisition devices are deployed at different locations to measure a certain set of attributes of the study subject. How to manipulate these spatial data remains a challenge to the database community, especially when the spatial locations are represented in 3D. In this work, we establish a data model to handle multi-dimensional spatial data with three spatial dimensions. In particular, firstly, a clustering algorithm is applied to group the data set into “point clouds”. Secondly, each cloud is considered as a 3D spatial convex object and triangulated into a set of tetrahedrons. Thirdly, all tetrahedron sets are stored in the database and spatial analysis is performed. In this paper, we focus on defining 3D spatial operations and relationships for 3D spatial elements (points, segments, triangles and tetrahedrons), and further applying these operations on 3D spatial objects, where each object is composed of a set of tetrahedrons.

1. INTRODUCTION

In various GIS research fields, multi-dimensional spatial data are frequently generated. Multi-dimensional spatial data are obtained when a number of data acquisition devices are deployed at different locations to measure a certain set of attributes of the study subject. For example, scientists may be interested in studying the weather conditions in a region. Thus, sensors are distributed to measure air pressures, temperatures, humidity, radio strengths, etc., at different locations. Thousands of data points, with measurements of the different attributes, associated with their spatial signature, pose many challenging questions: How to analyze the data? How to interpret the information? How to manipulate the data to support efficient data querying?

Our recent research is directed toward developing an efficient query system for multi-dimensional spatial data, where each data point comprises a 3D spatial location along with other attributes. The entire data processing consists of three phases. In the first phase, we preprocess the data set and partition it into clusters. In particular, we use an I/O efficient, fast clustering method for high-dimensional data without leaving out any dimension. Due to the “curse of dimensionality”, clustering high-dimensional data is intrinsically difficult. Our recently designed clustering algorithm [6] provides a solid foundation for the next two phases. After the first phase, the original data set will be partitioned into clusters, or say, point clouds, which can be considered as 3D spatial objects. Then, in the second phase, we apply computational geometry algorithms to compute the 3D spatial boundary (convex hull) of each point cloud and triangulate the space enclosed in each boundary into a set of tetrahedrons. These tetrahedrons will be loaded into the database to represent each 3D spatial object. The second phase provides data normalization so that the data are suitable for modeling and querying in the next phase. It also provides data reduction since the number of tetrahedrons of a 3D spatial object is usually much smaller than that of the data points. In the third phase, we establish a 3D spatial data model to manipulate the data acquired after the second phase, where supports for 3D spatial operations and relationships are provided. Since each 3D spatial object is represented by a tetrahedron set, used-defined aggregates can be defined to operate on sets of tetrahedrons, which is also the main focus of this paper.

3D spatial modeling is an important aspect to 3D GIS. Nowadays, spatial locations are typically represented by the three dimensional coordinates of latitude, longitude, and height. However, not until recently, spatial data models have been designed to support 2D spatial data [2, 3, 5], where latitudes and longitudes are presented. Lately, some work on 3D spatial data modeling has emerged [1, 10]. However, so far the focus of this field is on the geological surface of the earth, such as in urban planning [4], 3D cadastre [8], city visualization [4], city modeling, etc. In [7], Molenaar introduced a model called 3D FDS (Formal Data Structure). A geometric object (point, line, surface or body) can only be described using one of the primitives (node, arc, face, or edge) of the same dimension. This concept is called a single-valued map. The approach is trying to isolate the non-overlapping objects in the space. A problem that this model faces is that it lacks the flexibility to manipulate complex spatial objects. In [9], Zlatanova proposed the Simplified Spatial Model, which is suitable for processing online visualization queries. In this model, however, there are only two primitives, node and planar face. One of the advantages of the model is that primitives can be abstracted with ease. In [4], the Urban Data Module (UDM) was described by Coors for
city visualization. Since it applies triangulation on all the surfaces, the primitives are node and triangle. The shortcoming of this model is that it demands greater storage, and it is difficult to perform update.

In this paper, we present our 3D spatial data modeling and spatial analysis on 3D spatial objects.

2. PROPERTIES OF 3D SPATIAL OBJECTS
2.1 3D Spatial Elements and Their Representations

In geometry, n-simplex is often used to represent the simplest polytope in the n-dimensional space. For example, a 0-simplex is a point, and a 1-simplex is a segment. In two dimensions, the simplex is a triangle. In three dimensions, the simplex is a tetrahedron. These four simplices are the building blocks in our 3D spatial database. Plus, we also consider line and plane. A line may be considered as a segment, with each end extended infinitely. Similarly, a plane can be defined by a triangle with unlimited boundaries. Note that line and plane are conceptual in the database. Their existence assists in studying the relationships between the four fundamental spatial elements.

In the 3D spatial database, POINT is the primitive element and used to define other geometric elements. A POINT is represented by its three coordinates in the Cartesian coordinate system. Therefore, a SEGMENT (or LINE) is defined by two POINTs. A TRIANGLE (or PLANE) is composed of three POINTs, and a TETRAHEDRON consists of four POINTs. The representations of the 3D spatial elements are listed in Table 1.

<table>
<thead>
<tr>
<th>Spatial element</th>
<th>Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>POINT</td>
<td>( P(x, y, z) )</td>
</tr>
<tr>
<td>SEGMENT/LINE</td>
<td>( P_1(x_1, y_1, z_1), P_2(x_2, y_2, z_2) )</td>
</tr>
<tr>
<td>TRIANGLE/PLANE</td>
<td>( P_1(x_1, y_1, z_1), P_2(x_2, y_2, z_2), P_3(x_3, y_3, z_3) )</td>
</tr>
<tr>
<td>TETRAHEDRON</td>
<td>( P_1(x_1, y_1, z_1), P_2(x_2, y_2, z_2), P_3(x_3, y_3, z_3), P_4(x_4, y_4, z_4) )</td>
</tr>
</tbody>
</table>

2.2 Spatial Operations and Relationships of Spatial Elements

Our goal is to be able to determine the relationship between tetrahedrons. In order to achieve that, we need to define various operations on the other spatial elements, as well as study the relationships between these spatial elements. The spatial operations and relationships needed are listed in Table 2. Some of the operations and relationships are already studied in different geometry articles. In this section, we discuss the following:

1. intersect_tetrahedron_tetrahedron:
   The intersection between two tetrahedrons is a convex polyhedron, which may be considered as the space enclosed by a collection of points. These points are hence the vertices of the convex polyhedron, which may also degenerate to a polygon, a segment, a point, or null. The vertices of this convex polyhedron are the intersections between the edges of one of the two tetrahedrons and the faces of the other tetrahedron, or the vertices of these two tetrahedrons. As an example, Figure 1 shows the intersection between two tetrahedrons that is a hexahedron. The left figure is the intersection without hidden lines, and the right one highlights the hexahedron. The vertices of this hexahedron is composed of two vertices of the two tetrahedrons, and six intersections between the edges of one tetrahedron and the faces of the other.

2. relation_tetrahedron_tetrahedron:
   The relationship between two tetrahedrons \( T_1 \) and \( T_2 \) is defined as follows:
   - Case 0 (disjoint): \( T_1 \) and \( T_2 \) are disjoint;
   - Case 1 (meet): \( T_1 \) and \( T_2 \) meet at a single point;
   - Case 2 (share): \( T_1 \) and \( T_2 \) share a common segment;
   - Case 3 (adjacent): \( T_1 \) and \( T_2 \) share a common face;
   - Case 4 (overlap): \( T_1 \) and \( T_2 \) overlap.
Besides the above five relationships, we are also interested in the following two relationships that are categorized under overlap:

- contain: $T_1$ is contained by $T_2$ or vice versa;
- equal: $T_1$ and $T_2$ are identical.

3. **contain tetrahedron tetrahedron:**
   TETRAHEDRON $T_1$ is contained by TETRAHEDRON $T_2$ if every vertex of $T_1$ is inside $T_2$.

4. **disjoint tetrahedron tetrahedron:**
   TETRAHEDRON $T_1$ and TETRAHEDRON $T_2$ are disjointed if all the following four conditions are satisfied:
   
   - Every vertex of $T_1$ is outside $T_2$;
   - Every vertex of $T_2$ is outside $T_1$;
   - The relationship between any edge of $T_1$ and any face of $T_2$ is disjoint;
   - The relationship between any edge of $T_2$ and any face of $T_1$ is disjoint;

5. **meet tetrahedron tetrahedron:**
   TETRAHEDRON $T_1$ and TETRAHEDRON $T_2$ meet at a single point in one of the following four cases, as illustrated in Figure 2.

6. **share tetrahedron tetrahedron:**
   TETRAHEDRON $T_1$ and TETRAHEDRON $T_2$ share a common segment in one of the following sixteen cases, as illustrated in Figure 3.

7. **adjacent tetrahedron tetrahedron:**
   TETRAHEDRON $T_1$ and TETRAHEDRON $T_2$ are adjacent, if there exists a pair of triangle faces from each of the two tetrahedrons that overlap, plus, the fourth vertex of $T_1$ and that of $T_2$ are on the different side of the plane defined by the shared faces. As shown in Figure 4, suppose that triangle $P_1P_2P_3$ and triangle $Q_1Q_2Q_3$ overlap. Then, for the figure on the left, $T_1$ and $T_2$ are adjacent since $P_4$ and $Q_4$ are on the different side of the plane where triangle $P_1P_2P_3$ and triangle $Q_1Q_2Q_3$ reside, but for the figure on the right, $T_1$ and $T_2$ are not.

8. **overlap tetrahedron tetrahedron:**
   TETRAHEDRON $T_1$ and TETRAHEDRON $T_2$ overlap if their relationship is not disjoint, or meet, or share, or adjacent.

**2.3 Spatial Operations and Relationships of Spatial Objects**

In the previous section, we have discussed the relationships and operations of the 3D spatial elements, especially those of TETRAHEDRON. In general, these 3D spatial elements (POINT, SEGMENT/LINE, TRIANGLE/PLAN, TETRAHEDRON) are also spatial objects. However, in this section, we focus on the spatial objects that are 3D convex polyhedrons, namely OBJECTs. Since each 3D convex polyhedron may be triangulated into a set of TETRAHEDRONs, an OBJECT is composed of a group of TETRAHEDRONs. This data model enables us to manipulate 3D spatial objects with ease, as we can analyze the relationship between 3D OBJECTs by studying collectively the relationships between the sets of TETRAHEDRONs of the OBJECTs. Moreover, in some modern commercial database management systems, support for user-defined aggregate functions provides us with the freedom of defining the relationships and operations of 3D spatial objects inside the database engine, for example, Oracle PL/SQL.

Let $O_1$ and $O_2$ denote two 3D convex polyhedral OBJECTs, each of which consists of a set of TETRAHEDRONs $S_1$ and $S_2$, respectively. The following relationships and operations may readily be defined:

1. **disjoint:**
   The relationship between $O_1$ and $O_2$ is disjoint if $\forall T_1 \in S_1$ and $T_2 \in S_1$, the relationship between $T_1$ and $T_2$ is also disjoint.

2. **meet:**
Figure 5: Algorithm for calculating the distance between two 3D spatial objects

The relationship between $O_1$ and $O_2$ is meet if $\exists T_1 \in S_1$ and $T_2 \in S_2$ such that the relationship between $T_1$ and $T_2$ is meet AND $\forall \ T_1 \in S_1$ and $T_2 \in S_2$, the relationship between $T_1$ and $T_2 \leq 1$.

3. share:
The relationship between $O_1$ and $O_2$ is share if $\exists T_1 \in S_1$ and $T_2 \in S_2$ such that the relationship between $T_1$ and $T_2$ is share AND $\forall \ T_1 \in S_1$ and $T_2 \in S_2$, the relationship between $T_1$ and $T_2 \leq 2$.

4. adjacent:
The relationship between $O_1$ and $O_2$ is adjacent if $\exists T_1 \in S_1$ and $T_2 \in S_2$ such that the relationship between $T_1$ and $T_2$ is adjacent AND $\forall \ T_1 \in S_1$ and $T_2 \in S_2$, the relationship between $T_1$ and $T_2 \leq 3$.

5. overlap:
The relationship between $O_1$ and $O_2$ is overlap if $\exists T_1 \in S_1$ and $T_2 \in S_2$ such that the relationship between $T_1$ and $T_2$ is overlap.

6. contain:
Let collection $C_1$ be the set that contains all vertices of every $T_1 \in S_1$. Then, $O_1$ is contained by $O_2$ if $\forall$ vertex $P_1 \in C_1$, $\exists$ vertex $P_2 \in S_2$ such that $P_1$ is inside $T_2$.

7. equal:
Since there are multiple ways to triangulate a given OBJECT, to decide whether $O_1$ and $O_2$ are equal, it is insufficient to simply compare whether $S_1$ and $S_2$ are equal. However, if $O_1$ and $O_2$ are equal, the two sets of points that form the boundaries of $O_1$ and $O_2$ must be equal.

Let collection $C_1$ be the set that contains all vertices of every $T_1 \in S_1$. Let collection $C_2$ be the set that contains all vertices of every $T_2 \in S_2$. Then, $O_1$ and $O_2$ are equal if the size of $C_1$ is equal to that of $C_2$ AND $\forall$ vertex $P_1 \in C_1$, $\exists$ vertex $P_2 \in C_2$ such that $P_1$ and $P_2$ are equal.

8. distance:
The distance between $O_1$ and $O_2$ can be calculated using the algorithm in Figure 5.

9. intersect:
Similar to intersect_tetrahedron_tetrahedron, the intersection between $O_1$ and $O_2$ is acquired by computing all the POINTs that constitute the boundary of the intersection. The algorithm is shown in Figure 6.

Figure 6: Algorithm for calculating the intersection of two 3D spatial objects

3. CONCLUSIONS
We presented a methodology to manipulate multi-dimensional spatial data with three spatial dimensions, which consists of three phases summarized as clustering, data normalization, modeling and querying. We studied extensively the 3D spatial relationships and operations of the 3D spatial elements such as points, segments, triangles and tetrahedrons. We also discussed the 3D spatial relationships and operations of the 3D spatial objects, which are composed of sets of tetrahedrons.

4. REFERENCES