A database paradigm for the management of DICOM-RT structure sets using a geographic information system

Weber Shao[1], **Patrick A Kupelian**[1], **Jason Wang**[1], **Daniel A Low**[1], **Dan Ruan**[1]
[1]Department of Radiation Oncology, David Geffen School of Medicine at UCLA

E-mail: druan@mednet.ucla.edu

Abstract. We devise a paradigm for representing the RT structure sets in the database management system, in such way that secondary calculations of geometric information can be done quickly from the existing RT contour definitions. The implementation of the paradigm is achieved using the PostgreSQL database system and the PostGIS extension, a geographic information system commonly used for encoding geographical map data. The proposed paradigm eliminates the overhead of retrieving large data records from the database, as well as the need to implement various numerical and data parsing routines, when additional information related to the geometry of the anatomy is desired.

1. Introduction

DICOM-RT, a widely utilized extension to the Digital Imaging and Communications in Medicine (DICOM) imaging standard, represents many aspects of radiotherapy imaging. One of such is the RT structure sets, a group of definitions of anatomical regions of interest. It is desirable to devise a paradigm for representing the RT structure sets in the database management system, in such way that secondary calculations of geometric information can be performed quickly from the existing RT contour definitions.

To achieve this, there are two possible general approaches. One approach is to rely on client-side processing after record retrieval from the database. Using this method, client-side computation can be carried out swiftly if the routine is implemented using a compiled language such as C, at the expense of having to handle memory management. A bigger drawback of this approach is the relatively long data retrieval time due to the size of the collection of RT contour records.

The alternative approach is to perform computation directly on the database server before the query and retrieval. This method produces shorter data retrieval time than does the client-side processing approach, since only the results, as opposed to the sets of contours, need to be returned by the SQL query. However, the performance would suffer significantly if the mathematical calculations, which would require loops, are carried out purely using a stored procedure language, which is processed through an interpreter.

Therefore, in order for the server-side computation approach to be efficient, it ought to be supplemented with an additional set of SQL functions in the form of pre-compiled binary objects. While it is possible to extend the SQL language by writing and compiling C routines into functions, it is optimal to employ a geographic information system (GIS), which is historically used for storing, managing, and manipulating geographical spatial data. Specifically, we aim to accomplish this goal using PostGIS, an extension of the PostgreSQL database management system (DBMS).

2. Experiment Background

An experiment is conducted to measure the computations of three common quantities pertaining to a structure set: surface area, volume, and geometric centroid. The experiment encompasses the two aforementioned approaches and is administered in two settings for each approach:

2.1. Approaches

We hereinafter refer to the approach of server-side computation using GIS as Approach 1, and the approach of client-side computation as Approach 2. In each approach, the database query operation is wrapped inside a C client program supported by the libpq5 library.

2.1.1. Approach 1. PostgreSQL 9.1 is used as the DBMS, along with PostGIS 2.0.1.

2.1.2. Approach 2. PostgreSQL 9.1 is used as the DBMS.

2.2. Settings

The database server is hosted on a workstation with 12 Intel Xeon E5645 2.4GHz CPUs and 12GB of RAM. Settings 1 and 2 differ in the location of the client program.

2.2.1. Setting 1. The client program resides on the same workstation as that of the database server.

2.2.2. Setting 2. The client program resides on a remote workstation, which has 2 Intel(R) Core(TM) 2 Duo E7400 2.80GHz CPUs and 4GB of RAM, and is connected through a cable modem at a maximum of 30Mbps. The remote workstation is 7 networking hops away from the database server, and physically separated by 23 miles.

3. Structure

In the DICOM-RT specification, one course contains several structures. Each structure is represented by a collection of contours. Each contour is a closed set of vertices which share a common z-coordinate. In Approach 1, the set of vertices is modelled by *polygon*, a two-dimensional PostGIS geometric data structure. In Approach 2, the set of vertices is modelled by an array of floating point numbers. With the exception of the difference in the data type representation of contours, the database tables and their relations are identical in both approaches.

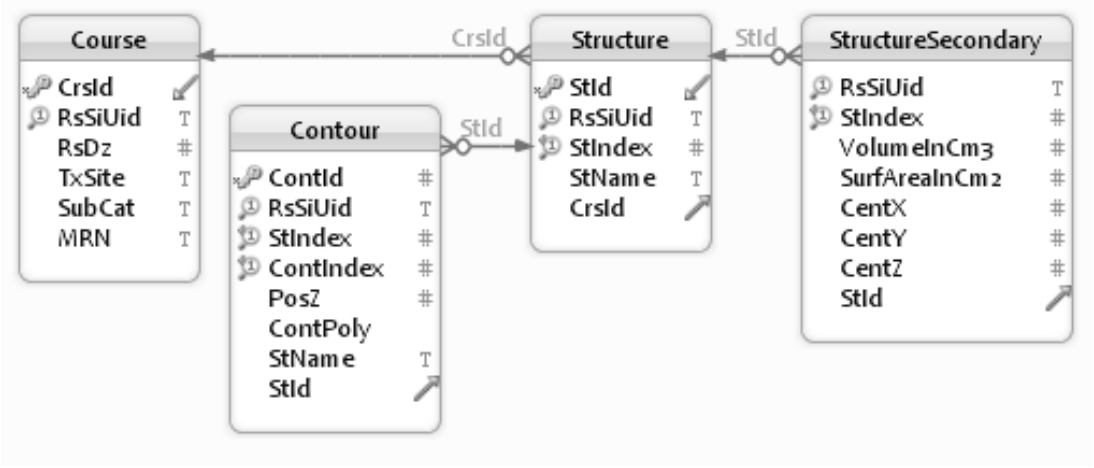

Figure 1. Storage Paradigm for RT Contours

4. Methods

For Approach 1, we will invoke several PostGIS API functions to carry out the computation of two-dimensional geometric measurements as SQL columns, which is used with aggregate functions, and if necessary, nested queries, to calculate the quantities interest: surface area, volume, and geometric centroid. As of PostGIS version 2.0.1, three-dimensional support is only limited to a few functions related to polygonal surfaces and triangular irregular networks, and therefore is not used by this approach. Approach 2 is implemented by directly coding the mathematics of Equations (1) to (7) inside the C routines.

4.1. Surface Area

For computing the total surface area of an anatomical structure represented in DICOM-RT, the algorithm is to find the product of the total path length and the slice thickness of its z-plane, Δz:

$$A_{surface} = \Delta z \sum_{i=0}^{N-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \tag{1}$$

This calculation is simple because the slice thickness denoted in DICOM-RT structure sets is uniform across all contours. Approach 1 requires only a single SQL query, invoking the *st_boundary()* function to return the vertices of the polygon, and the *st_length()* function to yield the contour path length. The surface areas of all contour sets within a treatment course can be obtained by the following query:

```
SELECT cotr."RsSiUid", cotr."StName",
  SUM(crs."RsDz" * st_length(st_boundary(cotr."ContPoly")))
FROM "Contour" AS cotr, "Course" AS crs
WHERE cotr."RsSiUid" = crs."RsSiUid"
AND crs."MRN" = 'XX-XX-XXX'
GROUP BY cotr."RsSiUid", cotr."StIndex", cotr."StName"
ORDER BY cotr."RsSiUid", cotr."StIndex"
```

4.2. Volume

The volume of a DICOM-RT anatomical structure is given by the product of the slice thickness and the sum of the areas of each contour:

$$V = \Delta z \sum_{j=0}^{M-1} A_{contour,j} \tag{2}$$

The area of each contour is given by:

$$A_{contour} = \frac{1}{2} \sum_{i=0}^{N-1} (x_i y_{i+1} - x_{i+1} y_i) \tag{3}$$

Thus the SQL query in Approach 2 for the calculation for (2) can be made with the assistance of the invocation of the *st_area()* function, which produces the contour area in lieu of implementing the mathematics of (3) directly. Thus the volumes of all contour sets within a treatment course can be obtained by one query:

```
SELECT cotr."RsSiUid", cotr."StIndex", cotr."StName",
  SUM(crs."RsDz" * st_area(cotr."ContPoly"))
FROM "Contour" AS cotr, "Course" AS crs
WHERE cotr."RsSiUid" = crs."RsSiUid"
AND crs."MRN" = 'XX-XX-XXX'
GROUP BY cotr."RsSiUid", cotr."StIndex", cotr."StName"
ORDER BY cotr."RsSiUid", cotr."StIndex"
```

4.3. Centroid

To calculate the three-dimensional centroid of a structure, first we need to find the two-dimensional centroid (P_x, P_y) of each contour that belongs to the structure:

$$P_x = \frac{1}{6A_{signed}} \sum_{i=0}^{N-1} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \tag{4}$$

$$P_y = \frac{1}{6A_{signed}} \sum_{i=0}^{N-1} (y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i) \tag{5}$$

where A_{signed} is the signed area defined by:

$$A_{signed} = \sum_{i=0}^{N-1} (x_i y_{i+1} - x_{i+1} y_i) \tag{6}$$

Then the three-dimensional centroid (C_x, C_y, C_z) of the structure would be the weighted average of the centroid of each contour, where the weight is the area of each contour, $A_{contour,j}$:

$$\begin{bmatrix} C_x \\ C_y \\ C_z \end{bmatrix} = \frac{1}{\sum_{j=0}^{M} A_{contour,j}} \begin{bmatrix} \sum_{j=0}^{M} A_{contour,j} P_{x,j} \\ \sum_{j=0}^{M} A_{contour,j} P_{y,j} \\ \sum_{j=0}^{M} A_{contour,j} P_{z,j} \end{bmatrix} \tag{7}$$

While P_x and P_y are given by Equations (4) and (5), P_z is simply the z-coordinate value where the contour is defined, which corresponds to the *PosZ* column of the of the *Contour* table in the database structure diagram in Figure 1. For the SQL query in Approach 1, we can use a subquery to compute the reusable quantities of area contour, $A_{contour,j}$. The PostGIS *st_centroid()* function would handle the calculation of the two-dimensional contour centroid, while the *st_x()* and *st_y()* functions would return the x and y coordinates, respectively. Thus, the calculation of the geometric centroids of all the structure sets for one treatment course can be accomplished in one nested query:

```
SELECT bar."RsSiUid", bar."StIndex", bar."StName",
  SUM(bar.area * bar.x) / SUM(bar.area),
  SUM(bar.area * bar.y) / SUM(bar.area),
  SUM(bar.area * bar.z) / SUM(bar.area)
FROM (
  SELECT cotr."RsSiUid",
    cotr."StIndex", cotr."StName", cotr."ContIndex",
    st_area(cotr."ContPoly") AS area,
    st_x(st_centroid(cotr."ContPoly")) AS x,
    st_y(st_centroid(cotr."ContPoly")) AS y,
    cotr."PosZ" AS z
  FROM "Contour" AS cotr, "Course" AS crs
  WHERE cotr."RsSiUid" = crs."RsSiUid"
  and crs."MRN" = 'XX-XX-XXX'
) AS bar
GROUP BY bar."RsSiUid", bar."StIndex", bar."StName"
ORDER BY bar."RsSiUid", bar."StIndex"
```

5. Results and Discussions

After 4 iterations each across 24 prostate cases, we obtain the following results of average processing time for each approach under with each setting:

Table 1. Performance comparison of settings and approaches

Setting, Approach	**Surface Area**	**Volume**	**Centroid**
Local Retrieval, PostGIS-Based	148 ms	124 ms	224 ms
Local Retrieval, Client-Based	415 ms	405 ms	405 ms
Remote Retrieval, PostGIS-Based	329 ms	304 ms	406 ms
Remote Retrieval, Client-Based	1296 ms	1300 ms	1290 ms

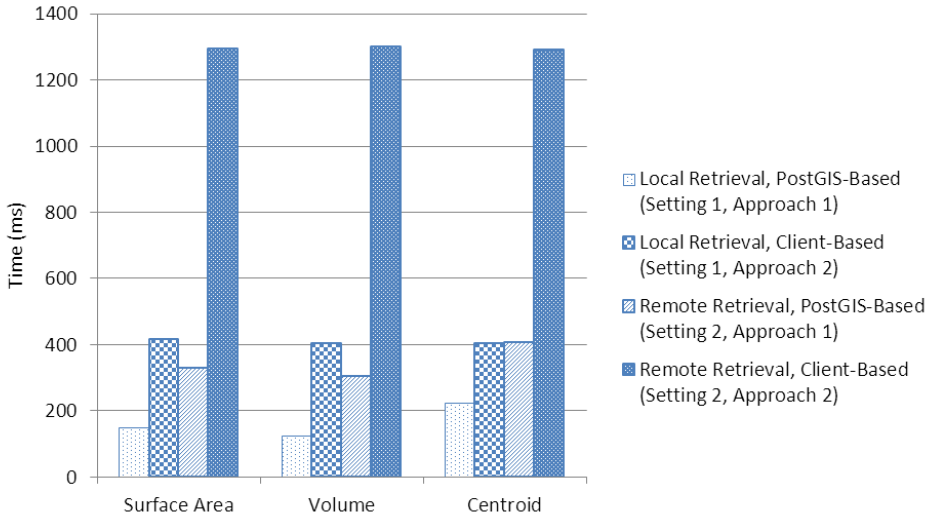

Figure 2. Performance comparison of settings and approaches

The approach of using PostGIS is noticeably faster than the approach of relying client-side computation. The difference is more evident in the setting where the workstation is remotely located from the database server. While the result of Setting 2, Approach 2 is dependent on the hardware processing power of the remote client, as well as the network transfer time, the scenario presented here is realistic. The database server typically has a more powerful processor and is equipped with more RAM than a client workstation. The database server and the remote client could very possibly have a longer distance or a slower connection mechanism than the one presented here, which would produce a sharper contrast in the performance of the two approaches.

6. Conclusions

The proposed paradigm for DICOM-RT structure set archival provides several benefits. First, it eliminates the overhead of retrieving large records from the database. Second, it simplifies the implementation of various numerical and data parsing routines, by relying on a solid application framework that is deployed for production worldwide. Additionally, the centralized, server-side design allows its functionalities to be utilized by various parties.

7. References

[1] Obe R and Hsu L 2011 *PostGIS in Action* (Shelter Island, NY: Manning)
[2] Bourke P 1988 *Calculating The Area And Centroid Of A Polygon*, Retrieved December 12, 2012 from http://paulbourke.net/geometry/polygonmesh/