The Mathematics of 3D Buffering for Simple Geospatial Primitives

Chen Tet Khuan and Alias Abdul Rahman

Institute for Geospatial Science and Technology
&
Department of Geoinformatics
Faculty of Geoinformation Science and Engineering
Universiti Teknologi Malaysia
81310 UTM Skudai, Johor,
Malaysia

{kenchen, alias}@fksg.utm.my

Abstract

This paper describes an effort towards realizing 3D buffering operations for 3D GIS. The buffering is based on spatial data primitives of points, lines, and polygons. All the analytical solutions for the buffering is discussed in detail and form a major discussion of this paper. The mathematics, the geometry, and the algorithms involved will be presented. We have tested the approach by using photogrammetrically captured data sets and a commercial GIS package for visualization purpose. Finally, we present the outlook of the 3D .analytical solutions for 3D GIS

Key words: 3D buffering, geo-spatial primitives, and 3D GIS

1. Introduction

Nowadays, 2D GISs are common and tasks involved in any GIS applications are quite straightforward and can be solved in a very efficient manner (see Abdul-Rahman et al, 2001). 2D spatial problem, such as buffering, overlay and network analysis are well-studied. However, adding an additional dimension to 2D GIS makes a big difference in most of its implementation. Zlatanova 2000 stated that the design, utilization and maintenance of a new 3D GIS comprised a wide spectrum of questions concerning a 3D conceptual model, data collection, spatial analysis, presentation and the newly-common utility, internet access. Although the advances in computer graphics have benefit to the 3D display and visualization, some critical aspects of 3D GIS, i.e. 3D spatial model (see Zlatanova, 2000), together with the semantics information is hardly well defined.

Although the available software in the market such as ArcView 3D Analyst provides 3D display and visualization, it still lack real 3D spatial operation. Kim et.al (1998) has mentioned the 3D spatial operation such as 3D buffering, but that kind of spatial operation is still in the experimental level. This paper attempts to generate 3D buffering of 3D primitives of spatial objects. The primitives are point, lines, surface or polygon, and solid. The output of the 3D buffering is in ArcView 3D Analyst format. However, other aspect of 3D operations such as topology and databasing are out of the scope of this paper.

The paper is organized in the following order: section 2 presents the mathematics of 3D buffering (point, line, and polygon) and forms a major discussion of this paper. We present some results in section 3 and finally conclusion in section 4.

2. The 3D Buffering

In GIS, buffering is an operation to generate a proximity information of a spatial object, e.g. phenomenon along linear features, polygon features, etc. The mathematics or the method for the 2D buffering is quite straight forward, but not the 3D version of the problem. This section describes our 3D buffering in detail (i.e. how to construct them geometrically) based on the geospatial primitives.

2.1 Point Buffering

For the point buffering, we consider a point as the origin of the model, see Fig. 1. The result of the point buffering is a sphere and it is inline with the ESRI's whitepaper (July, 1998), that is the implementation of the PolygonZ to create a solid buffering object. Therefore, joining all related polygon surfaces creates a sphere. Again Fig. 1 shows the construction of circles for 3D point buffering, whereas Fig. 2 shows the method to create a sphere.

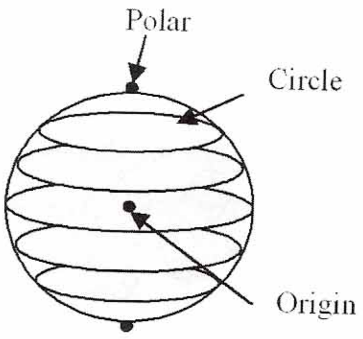

Figure 1: Sphere from a point.

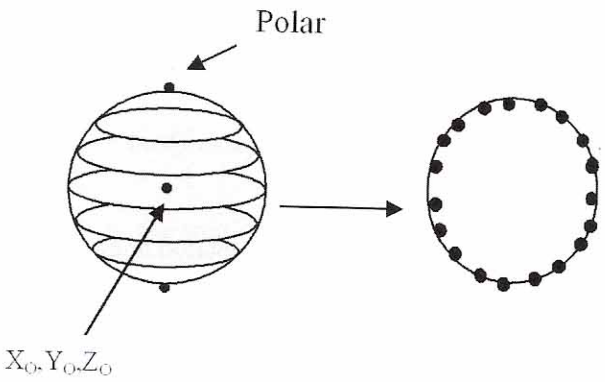

Figure 2: Method to create sphere

Each sphere has 11 rings, see Fig. 3.

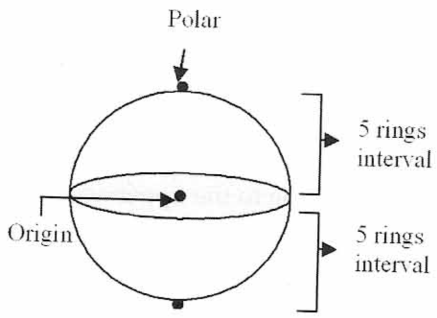

Figure 3: Structure of a sphere

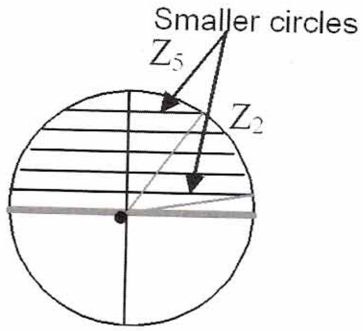

Figure 4: Calculate the Zk

The mathematics of the point buffering follows is as follows:

$$r_k = \sqrt{r^2 - \left(\left(\frac{r}{5}\right)\times(k)\right)^2}$$

$$x_i = r_k \times \cos(\theta_i)$$

$$y_i = r_k \times \sin(\theta_i)$$

Upper ring :

$$z_k = r + \left(\left(\frac{r}{5}\right)\times k\right)$$

Lower ring:

$$z_k \quad = \quad r - \left(\left(\frac{r}{5} \right) \times k \right)$$

where r is the buffer length from origin; k is the number of ring from 1 to 5 for both upper and lower ring; i is the angle from 00 to 3600; rk is the buffering length corresponding to the upper and lower ring k; xi, yi, zk are the coordinates for ring k.

For the polar point,

xp = xo
yp = yo

Upper polar point:
zp(upper) = zo + r

Lower polar point:
zp(lower) = zo - r

where xo, yo, zo are the coordinates for the origin; xp, yp, zp(upper)/zp(lower) are the processed coordinates for each ring.

After creating all the circles and the polar points, all points from a circle need to be joined to the successive circle in order to become a surface for a sphere. The final result of the sphere is Fig. 5.

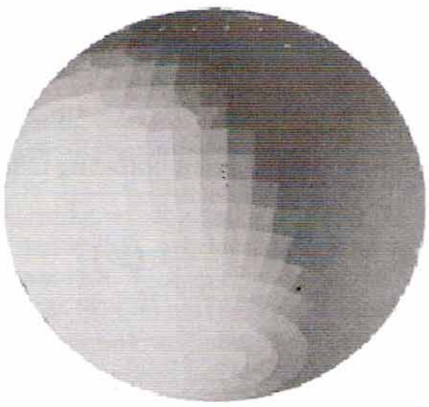

Figure 5: Point buffering output

2.2 Line Buffering

Since a line is a combination of the nodes (points) and the line itself, therefore, output of the 3D line buffering would be the combination of spheres and cylinder. The 3D line buffering approach is shown in Fig. 6.

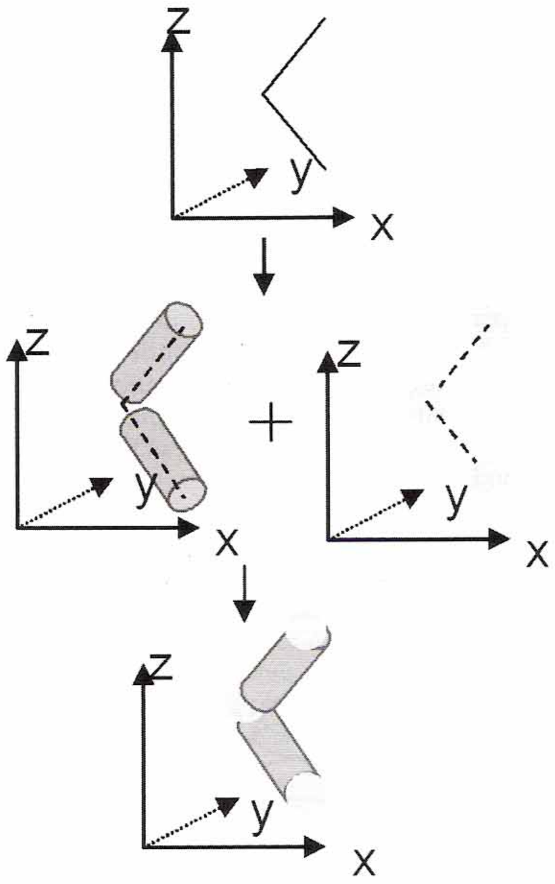

Figure 6: Method to create line buffer

The mathematics of the line buffering is as follows:

$$\theta^{XY} = \text{Tan} -1 \ (\frac{\Delta y}{\Delta x});$$

$$\theta^{XZ} = \text{Tan} -1 \ (\frac{\Delta z}{\Delta x});$$

$$\theta^{YZ} = \text{Tan} -1 \ (\frac{\Delta z}{\Delta y});$$

where θ_{XY}, θ_{XZ}, θ_{YZ} are the rotated angle for XY, XZ, and YZ plane respectively.

After calculating the angle ( ) for each plane, it uses for rotating the circle. Firstly, we rotate the plane circle toward the x-axis.

$$x_{(Z)_i} = r \times \cos(i)$$
$$y_{(Z)_i} = r \times \sin(i) \times \cos(\theta_Z)$$
$$z_{(Z)_i} = r \times \sin(i) \times \sin(\theta_Z)$$

where $z_{(Z)_i}$, $y_{(Z)_i}$, $x_{(Z)_i}$ are rotated coordinates for YZ plane.

Later on, the rotated circle will again rotate for the y-axis. Before we get through this step, the latitude and longitude of the points from the circle need to be calculated.

$$\begin{aligned} B &= \theta\{x_i, z_i\} \\ \text{LongB} &= B + (\theta_{XZ}) \\ \text{LatB} &= \cos\text{-}1\ (Yi / r) \end{aligned}$$

$$x_{(X)_i} = r \times \cos(LongB) \times \sin(LatB)$$
$$y_{(X)_i} = r \times \cos(LatB)$$
$$z_{(X)_i} = r \times \sin(LongB) \times \sin(LatB)$$

where B is the angle for each x_i, z_i; LongB is the longitude of x_i, z_i; LatB is the latitude of ;x_i, z_i

$z_{(X)_i}$, $y_{(X)_i}$, $x_{(X)_i}$ are the coordinates after the rotation of the y-axis.

Finally, it is followed by the rotation of the z-axis.

$$\begin{aligned} A &= \theta\{x_i, y_i\} \\ \text{LongA} &= A + (\theta_{XY}) \\ \text{LatA} &= \cos^{-1}(Z_i / r) \end{aligned}$$

$$x_{(X)_i} = r \times \cos(LongA) \times \sin(LatA)$$
$$y_{(X)_i} = r \times \sin(LongA) \times \sin(LatA)$$
$$z_{(X)_i} = r \times \cos(LatA)$$

Where A is the angle for each x_i, y_i; LongA is thelongitudeof x_i, y_i ;LatAisthelatitudeof x_i, y_i

$z_{(Y)_i}$ $y_{(Y)_i}$ $x_{(Y)_i}$, are the coordinates after the rotation of the y-axis.

After creating the rotated circle, a cylinder is created by joining the two succesive circles. Later on, the point buffer needs to be connected with the cylinder and become a complete 3D line buffering. However, we need to remove the unnecessary internal sphere's surface.

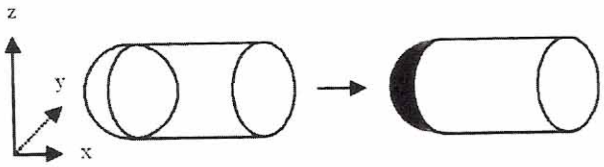

Figure 7: Delete internal segment

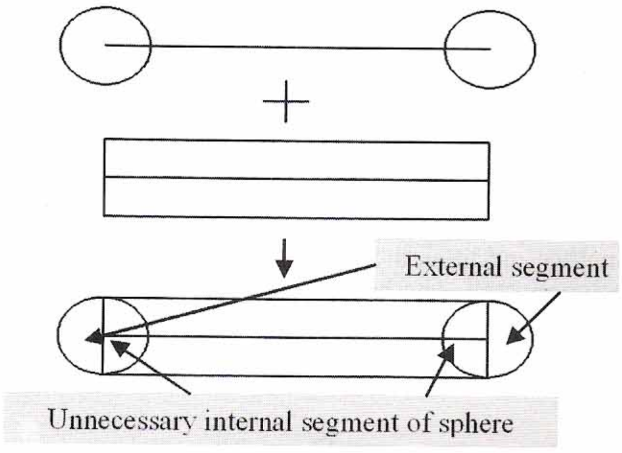

Figure 8: Side view of line buffering

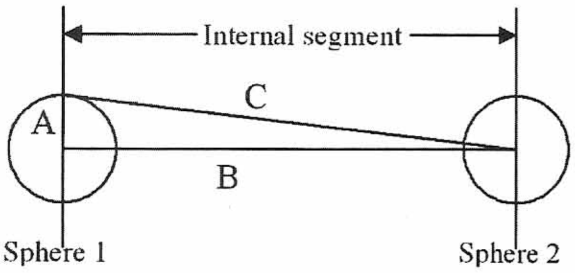

Figure 9: Method to delete the internal segment

Suppose that there are two points, forming a line. After joining the two spheres and a cylinder (see Fig. 8), internal segments need to be deleted in order to avoid "redundancy" for the medium of 3D line buffering. For any data at sphere 1, distance C is used to define internal segment, see Fig. 9. From the center of sphere 2 to any data at sphere 1, any distance that less than C at sphere 1 are considered as internal and will be removed from sphere 1. This process will be repeated in order to remove the internal segment at sphere 2. As a result, the final product of 3D line buffering is shown in Fig. 10.

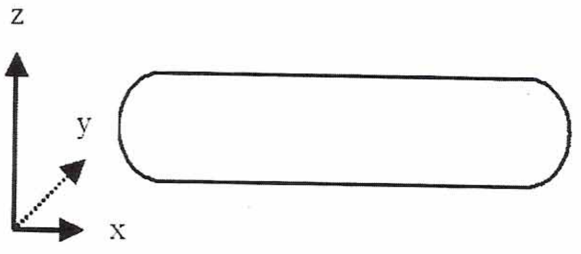

Figure 10: Final result of 3D line buffering

2.3 Polygon Buffering

A polygon is the combination of points, and lines. There are three kinds of geometrical primitives within a single polygon. To generate a 3D polygon buffer, we need to identify all the primitives, which are points, lines, and the polygon surface. As in the foregoing discussion, point-buffering output is a combination sphere, whereas the line-buffering output is the combination of spheres and cylinders. To have the polygon-buffering, we need to generate another additional buffering surface from the polygon itself. The buffering surfaces consist of an upper surface and a lower surface as shown in Fig. 11.

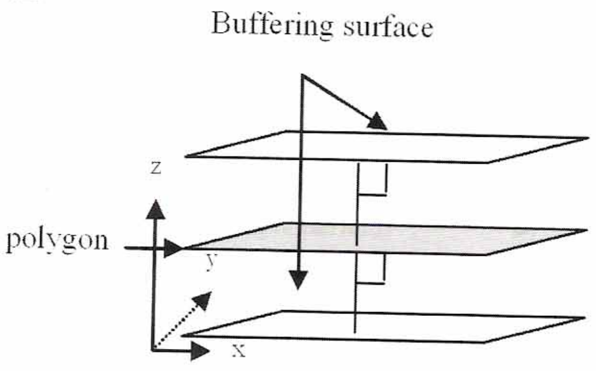

Figure 11: Buffering surface

Both of the buffering surfaces should be perpendicular to the polygon. Therefore vector-cross product is used to calculate the upper and lower surface of the polygon.

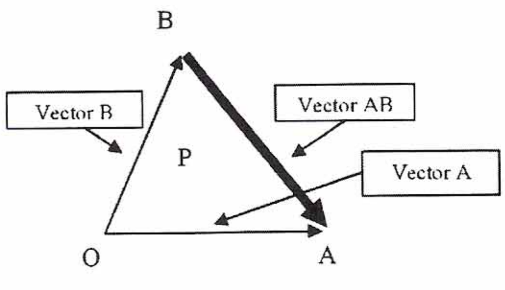

Figure 12: Vector for polygon P

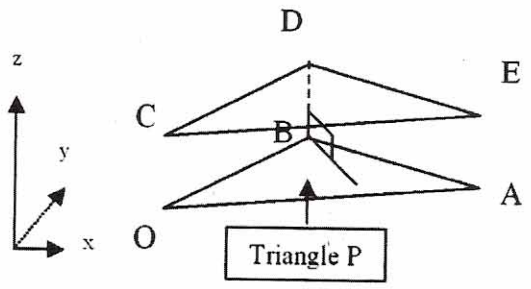

Figure 13: Upper polygon buffer CDE

For example, 3 points form a triangle P (see Fig 12). To calculate upper or lower surface of P, vector-cross product is used to calculate the points (say C,D, and E) that perpendicular to the surface of triangle P (see Fig 13). Some of the mathematics are illustrated below.

n1, n2, and n3 are the parameters for vector AB, whereas the Nx, Ny, and Nz are calculated to generate a constant vector unit for either upper or lower points of buffering surface.

$$\overrightarrow{OA} \times \overrightarrow{OB} = \left[|n_1| \quad |n_2| \quad |n_3| \right]$$

$$|n_1| = \begin{vmatrix} Y_{oa} & Z_{oa} \\ Y_{ob} & Z_{ob} \end{vmatrix}$$

$$|n_2| = \begin{vmatrix} Z_{oa} & X_{oa} \\ Z_{ob} & X_{ob} \end{vmatrix}$$

$$|n_3| = \begin{vmatrix} X_{oa} & Y_{oa} \\ X_{ob} & Y_{ob} \end{vmatrix}$$

$$\overrightarrow{AB} = \sqrt{\left(|n_1|^2 \quad |n_2|^2 \quad |n_3|^2 \right)}$$

$$\hat{AB} = \frac{\left[|n_1| \quad |n_2| \quad |n_3| \right]}{\overrightarrow{AB}}$$

$$= \left(|N_x| \quad |N_y| \quad |N_z| \right)$$

$$x_c = x_o + |N_x| \times r$$

$$y_c = y_o + |N_y| \times r$$

$$z_c = z_o + |N_z| \times r$$

where are the processed coordinates for the surface polygon buffer. r defines as the buffering distant from the center of polygon to the upper or lower polygon.

For the rest of the points, the same equations are implemented to calculate the point D and C. However, for the lower polygon buffer, there is a minor change in the equation. (A x B) is always perpendicular to both $\vec{A}$ and $\vec{B}$, with the orientation determined by the right-hand rule. Therefore,

$$(\vec{OA} \times \vec{OB}) = -(\vec{OA} \times \vec{OB})$$

After the upper and lower polygons buffer are defined, the line (cylinders) and point (sphere) buffering objects need to be combined. However, the internal segment between line buffer and the polygon buffer still remain unchanged. The same approach is used for removing them.

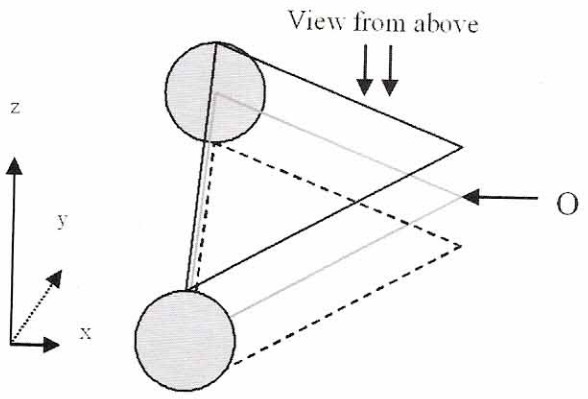

Figure 14: Structure of 3D polygon buffer.

Since the internal segment of sphere was removed, there is no any other intersection between the sphere and the box. To remove all the internal segment of line buffer, we just need to focus on the intersection between the box (polygon buffer) and cylinder (line buffer). First, we need to concentrate on the cylinder and the point O, which is the point opposite to the cylinder.

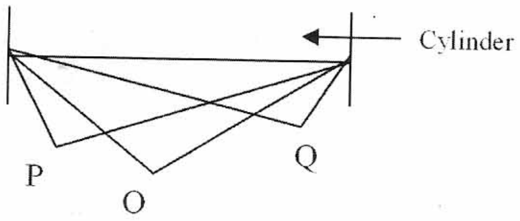

Figure 15: Orthogonal view.

Either the third point that creates a polygon is O, P, or even Q from the Fig. 15, the representation remains the same as in Fig. 16.

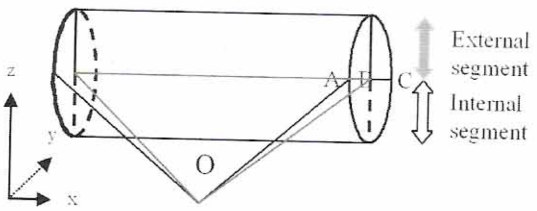

Figure 16: Removing the internal segment.

The cylinder is divided into two segments, which are the internal and external part.

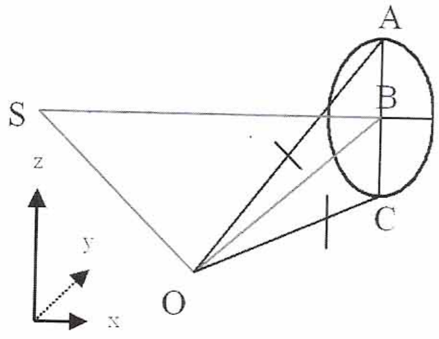

Figure 17: Calculate distance between OA, OB, OC

We need to calculate the distance of OA, and OC. Both are the same because the surface of the circle is perpendicular to the line AC, and point A and C are the upper and lower points of the circle. Therefore, there is a condition that needs to be fulfilled in order to remove the internal segment of the cylinder. That is the distance of either OA or OC is the benchmark of that condition. If any distance from the circle to point O is less than OA or OC, then it should be the internal part of the cylinder and need to be removed. Finally, Fig. 18 shows the result.

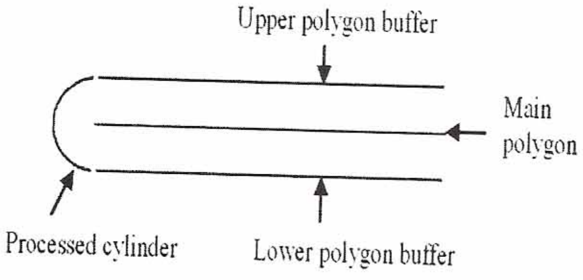

Figure 18: View from above

3. Experiments and Discussion

In this project, we use the C++ language to create a software module called 3D Buffering Tools and test it with the real dataset (UTM campus). The data was captured using digital photogrammetric system (Leica-Helava). Our 3D buffering module / software works with ArcView. The input is in ASCII format and the result is in the shapefile (*.shp). Fig.19, Fig. 20, Fig. 21, Fig.22, and Fig. 23 show the snap shots of the interface and output.

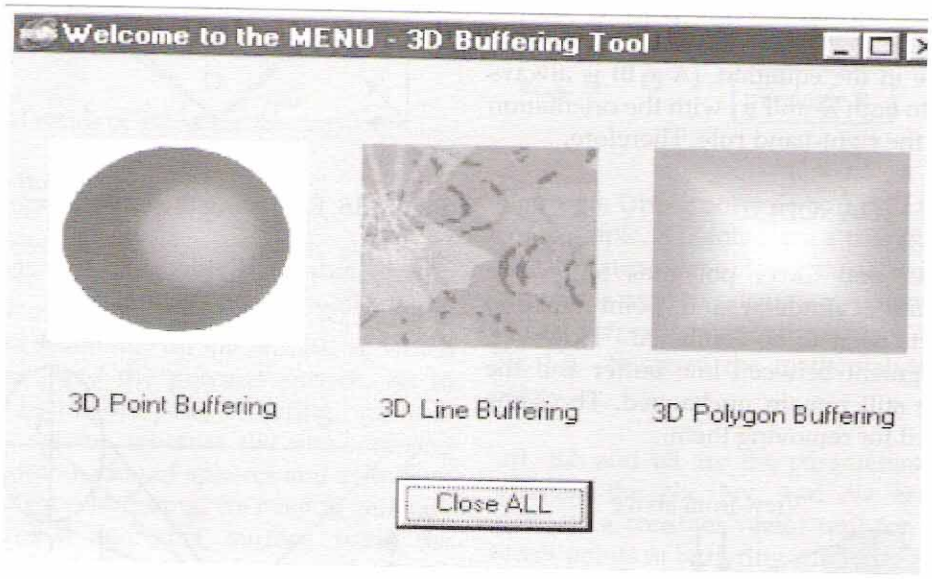

Figure 19: 3D Buffering Tool's menu interface

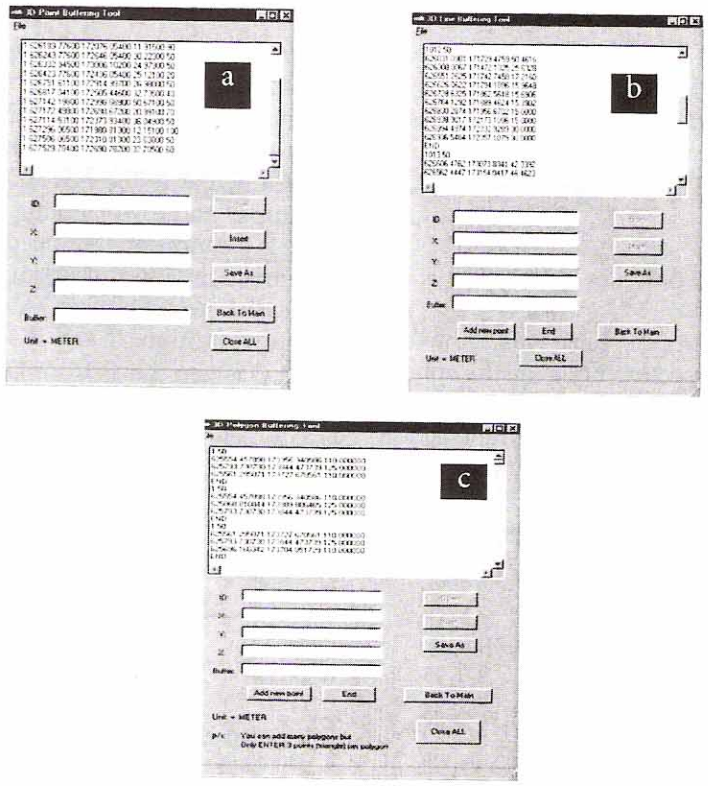

Figure 20: (a) Point, (b) Line & (c) Polygon buffering

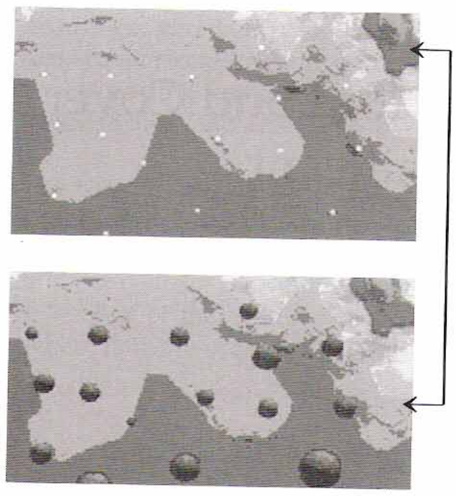

Figure 21: Point buffering output

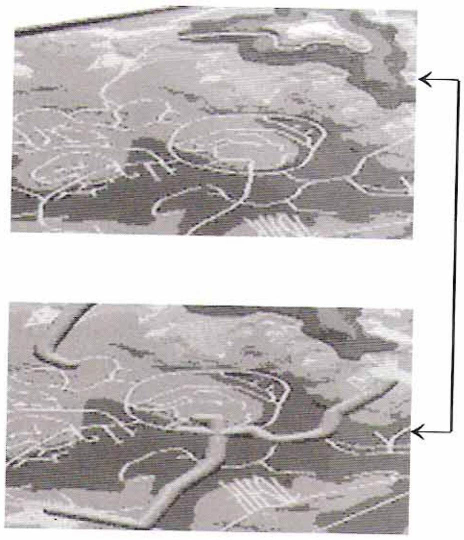

Figure 22: Line buffering output

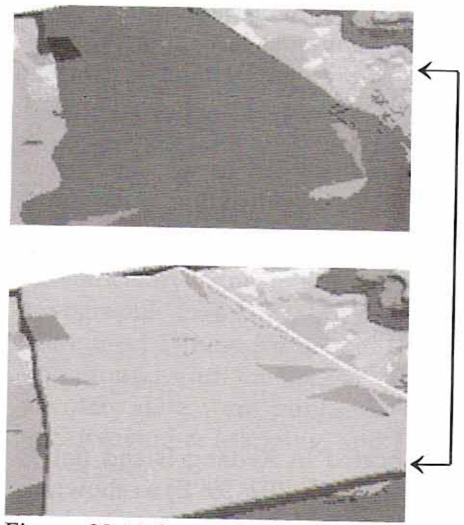

Figure 23: Polygon buffering output

4. Conclusion

The mathematics for creating the 3D buffering tool of geospatial primitives such as point, line, and polygon for 3D GIS have been presented.

We have tested the methods with the real datasets. The software module works separately from the ArcView GIS software package, however, further research needs to be looked into especially in the aspect of topology and the associated 3D analytical operations.

Acknowledgement

The authors of this paper would like to thank Lee KenYoong, Rusman Rusyadi, Surya Afnarius, and Chu Kai Chuan for their valuable help, and Zulkepli Majid for the datasets.

References

Abdelguerfi M., Wynne C., Cooper E., Roy L., (1998). Representation of 3-D Elevation In Terrain Databases Using Hierarchical Triangulated Irregular Networks: A Comparative Analysis. International Journal of Geographic Information Science, Vol. 12(8), pp. 853-873.

Abdul-Rahman, A., Zlatanova, S., Pilouk, M., (2000). 3D GIS Development: Status and Prospects, In: Seminar of Geoinformation 2001, 12-13 Nov, Penang, Malaysia.

ESRI Shapefile Technical Description, (1998). http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

Kim, K.H., Lee K., Lee H.G., and Ha Y.L., (1998). "Virtual 3D GIS's Functionalities Using Java/VRML Environment". http://www.sbg.ac.at/geo/eogeo/authors/kim/kim.html

Zlatanova, S., (2000). 3D GIS For Urban Development. ITC, PhD. Thesis, The Netherlands, 222 p.