GPSLoc: Framework for Predicting Global Positioning System Quality of Service

Hassan A. Karimi[1]; Xiong Liu[2]; Shuo Liu[3]; and Amin Hammad[4]

Abstract: While currently numerous existing engineering applications benefit from the global positioning system (GPS), it is anticipated that operation of many new, emerging applications (e.g., applications related to ubiquitous mobile computing) will rely on the information provided by this technology. Depending on the application requirement, GPS data may be collected and post-processed or collected and processed in real time. In either case, there are questions about availability, quality, and reliability of GPS data in engineering applications. To date, despite available techniques for realizing, and to some extent improving, a certain level of GPS accuracy, there is no integrated, coherent approach or technique that would provide users with solutions that combine GPS availability, quality, and reliability. To that end, we propose quality of service (QoS) assurance for GPS. With GPS QoS, users and applications would be provided with the means for predicting GPS solutions in advance meeting the requirements in a timely and cost-effective manner. We have developed a framework for the proposed GPS QoS called GPSLoc. In this paper, we discuss the requirements, methodologies, models, and algorithms for the GPSLoc framework and the experimentation with one of the GPS QoS parameters (visibility).

DOI: 10.1061/(ASCE)0887-3801(2004)18:3(196)

CE Database subject headings: Global positioning; Data collection; Computer applications; Algorithms.

Introduction

The global positioning system (GPS) has become a dominant positioning technology used in numerous applications. Example applications utilizing GPS are field tasks engineering, fleet and freight management, workforce management, facility and data mapping and modeling, incident/outage positioning, in-car navigation systems, automatic vehicle location (AVL) systems, location-based services, and mobile mapping systems (MMSs). In these and other civil and construction applications, data provided by GPS play a crucial role in the operation and delivery of information to the users. Timely and cost-effective decisions can only be made when there is a high degree of reliability on the information provided by GPS. However, GPS data are subject to uncertainties, and while it may not be possible to eliminate these uncertainties, having knowledge in advance about them helps improve the timeliness, usefulness, and reliability of GPS-based applications. In this regard, there is a need for quality of service (QoS) assurance for GPS. We define QoS of GPS as a set of techniques and strategies that could assure application and users a predictable service from GPS. Despite developments with respect to GPS issues, such as GPS accuracy measurements and improvements, currently there are no unified QoS methodologies and models for GPS (though disparate solutions for specific applications may exist in GPS receivers and software packages).

To better understand the issues of GPS QoS, we define *operation mode* and *requested QoS*. The *operation mode* is a reference to the ways GPS data are collected, which could be either *static* or *dynamic*. In the *static operation mode*, GPS is used to compute position data at one fixed location (or a set of selected locations one at a time) without the real-time processing constraint. In the *dynamic operation mode*, GPS is used to compute position data over a set of points in real time each at a different time; this is also called the *real-time operation mode*. The *requested QoS* is a reference to the required GPS solutions imposed by the user or application and could be either *passive* or *optimal*. In either the *passive* or *optimal requested QoS*, the assumption is that a location and a time are given by the user, or are determined by the application. In the *passive requested QoS*, a GPS solution is sought from GPS QoS. This means that the user requires a GPS solution no matter how good the solution is. In the *optimal requested QoS*, the most optimal solution is sought from GPS QoS. This means that the user requires a GPS solution that is most optimal with respect to availability, accuracy, reliability, etc.; in this case, the user is only interested in an optimal solution (not any solution) and that GPS QoS should only search for such solutions that meet the requirements. We also define the following four parameters in GPS QoS that are applicable to both *passive* and *optimal* modes: *visibility*, *accuracy*, *reliability*, and *flexibility*.

The *visibility* parameter of GPS QoS is a reference to those satellites, out of the available satellites, that are visible or have lines of sight (LOS) at the given location and time. This is needed because there are locations where GPS signals simply are not available due to obstruction of LOS. In other words, a LOS to a satellite is needed in order to obtain a signal representative of the true distance from the satellite to the receiver. Therefore, any object in the path of the signal has the potential to interfere with

[1]Dept. of Information Science and Telecommunications, Univ. of Pittsburgh, Pittsburgh, PA 15260.

[2]Dept. of Information Science and Telecommunications, Univ. of Pittsburgh, Pittsburgh, PA 15260.

[3]Dept. of Information Science and Telecommunications, Univ. of Pittsburgh, Pittsburgh, PA 15260.

[4]Concordia, Institute for Information Systems Engineering, Concordia Univ., Montreal PQ, Canada H3G 1T7.

Note. Discussion open until December 1, 2004. Separate discussions must be submitted for individual papers. To extend the closing date by one month, a written request must be filed with the ASCE Managing Editor. The manuscript for this paper was submitted for review and possible publication on December 13, 2002; approved on September 11, 2003. This paper is part of the ***Journal of Computing in Civil Engineering***, Vol. 18, No. 3, July 1, 2004. ©ASCE, ISSN 0887-3801/2004/3-196–206/$18.00.

the reception of that signal. Objects which can block a GPS signal include terrain heights, tree canopies, and buildings.

The *accuracy* parameter of GPS QoS is a reference to the level of accuracy the GPS receiver is able to compute at the required location and time. There are several external sources which introduce errors into a GPS position. Below are the sources of errors in GPS and the amount of error by each:

Source	Error level
Ionosphere	0–100 m
Troposphere	0–30 m
Measurement noise	0–5 m
Ephemeris data	0–5 m
Clock drift	0–1.5 m
Multipath	0–25 m

We define the *reliability* parameter as the ability of GPS QoS to guarantee a solution that meets the requirements of the user or application for the given location and time. In other words, upon receiving information from the user or application about the required QoS at a given location and time, GPS QoS will search for the set of solutions that meet the requirements.

We define the *flexibility* parameter as the ability of GPS QoS to provide alternative solutions (location, time, or both) in case there is no possible solution for the location and time requested. For example, when it is determined that there is no possible GPS solution for the given location and time that meets the user requirements, GPS QoS will search for a solution at the nearest location (at the same time), nearest time (at the same location), or both (at a different location and a different time) that meets the requirements.

It should be noted that the purpose of GPS QoS is not to mitigate error or improve accuracy, rather its goal is to provide information to predict QoS of GPS. Several benefits are expected from GPS QoS, some of which are:

1. Evaluating the quality of GPS data collected against the QoS required by the user;
2. Predicting the QoS that can be expected at a specific location and time;
3. Planning GPS data collection using GPS QoS maps to avoid undesirable solutions and maximize productivity.

Currently there are no coherent methodologies and techniques that provide users with predictive QoS of GPS. There exist off-the-shelf software packages (e.g., *Pathfinder* 1999) capable of providing the user with the satellites that produce the best solution for a given location and time. However, such solutions suffer from two shortcomings. One shortcoming is that they do not consider the actual physical environment of the user's position, that is, such three-dimensional (3D) data as terrain heights and buildings are not taken into account. Detailed and accurate 3D data (which contain terrain heights and 3D objects) of the geographic area where the user is located would help determine reliable GPS solutions. A common example of the need for 3D data on and near the location of the user for predicting QoS is when a GPS receiver searches for the satellites with good geometry. Without checking for potential obstacles using 3D data, the GPS receiver may include satellites that actually do not have LOS with the receiver. Another shortcoming is that existing solutions address mostly GPS accuracy-related issues focusing on individual cases or applications and are not coherently integrated with other issues (in addition to the accuracy issue) to provide a comprehensive QoS of GPS. We propose a framework called GPSLoc that includes

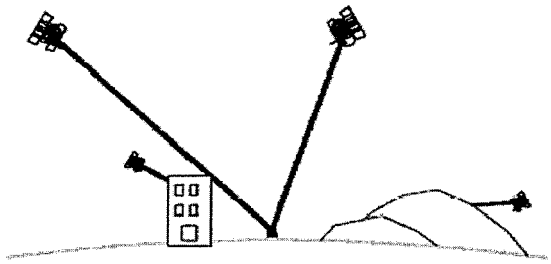


Fig. 1. Global positioning system signal blockage

models and algorithms to predict GPS QoS for applications using GPS. In GPSLoc, detailed 3D geometrical data on and near where the user is located are used and all possible GPS-related issues are taken into account. As is shown in Fig. 1, the LOS between each available satellite and the receiver may be obstructed by terrain heights, such as mountains, or by 3D objects, such as buildings.

In this paper, we focus on the *visibility* parameter for which we have developed solutions in GPSLoc. Our reason for focusing on the *visibility* parameter is that its result impacts all the other parameters. That is, by realizing visible satellites, GPSLoc can proceed to determine GPS QoS with respect to the other parameters, while without knowledge about visible satellites all subsequent computations and analyses are worthless. The two major components of GPSLoc for predicting GPS QoS are a 3D database and a set of algorithms for LOS intersection (LOSI). The 3D database component contains detailed data on terrain heights and other obstacles which are used in the set of algorithms to compute the LOS between each satellite and the receiver at a given location and time.

This paper's contributions are an introduction to the concept of GPS QoS for engineering and other applications that use GPS, development of a new methodology for predicting GPS QoS, and development of models and algorithms for GPS QoS. The structure of the paper is as follows. First, a representative engineering application where GPS QoS can be used is described. Second, the 3D database models and algorithms used in GPSLoc are described. Third, the algorithms for computing LOS used in GPSLoc are described. Fourth, the experimentation results using GPSLoc are discussed. In the last section, conclusions and future research are summarized.

Example Scenario

To better understand the need for GPS QoS and the benefits that users will gain from it, a sample application scenario using GPSLoc is described. GPSLoc can be of great value to utility infrastructure systems. Typically in utility infrastructure systems, data (including GPS data) of a high quality are desired, for example to identify utility lines or update existing maps based on maintenance performed. For example, a utility maintenance field crew that has to visit several sites during a day operation can use GPSLoc in more than one way. In the *planning mode*, GPSLoc can help the crew gain knowledge in advance about the QoS of GPS they will be able to obtain for each site. GPSLoc will analyze the planned locations and times of the visits and will provide the GPS solutions that meet the requirements. In doing so, GPSLoc will check the *visibility*, *accuracy*, and *reliability* parameters. If GPSLoc realizes that there are no solutions at all or that available solutions do not meet the requirements, it can suggest alternatives by using the *flexibility* parameter. The crew can use

this information provided by GPSLoc to plan visiting each site in an order that takes into account the priority of the maintenance needs as well as the QoS of GPS they require. Another way GPSLoc can assist the maintenance crew in a *real-time mode* is through the use of an AVL system (which may also have communication links with the office and other field crews) available in the vehicle where the real-time location of the vehicle is computed (and may be transmitted to others at the office or at other maintenance sites). GPSLoc for the AVL system will assist the crew in navigation and routing activities optimizing their performance while allowing the office managers to maximize the productivity of the overall maintenance operation by using the real-time locations of all the crews in the field. An interesting observation in this application is that the requested QoS of GPS for the *planning mode* (e.g., an accuracy range within a few centimeters, no real-time processing constraint, and flexible with respect to time of data collection) is different from the requested QoS of GPS for the *real-time mode* (e.g., an accuracy range within a few meters, real-time processing constraint, and fixed with respect to location and time).

Three-Dimensional Database Models and Algorithms

Terrain heights (e.g., mountains) and 3D objects (e.g., buildings) are the major obstacles for the GPS signal. There are two reasons why terrain heights are needed. First, places such as cliffs or hills are potential obstacles. Second, heights of 3D objects must be calculated relative to elevation of their base. Therefore, the 3D database in GPSLoc must include data on all types of obstacles including terrain heights and 3D objects and it must cover at least the area where GPS data are collected. The database must be based on one or more 3D models along with appropriate data structures and algorithms for 3D data manipulation. Currently there exist separate data models for terrain representation, used mostly in geospatial information system (GIS) software packages, and data models for 3D objects (e.g., buildings), used mostly in Computer-Aided Design (CAD) software packages.

Miller and La Flamme (1958) introduced the Digital Terrain Model (DTM) as ''a statistical representation of the continuous surface of the ground by a large number of selected points with known *X*, *Y*, and *Z* coordinates in an arbitrary coordinate field.'' DTM is a generic term for digital representation of terrain information. There are two widely used terrain models: the digital elevation model (DEM) and the triangulated irregular network (TIN). DEM is a grid (or raster) model where the terrain's elevation data are sampled at a regularly spaced interval and stored as an array (Lo and Yeung 2002). Each raster cell has an elevation value and all values make a matrix of elevations. TIN is a vector model (Peucker et al. 1978) based on a set of irregularly spaced points to represent the terrain. One advantage of TIN over DEM is the possibility of adapting the irregularly spaced sample points to the terrain's features. For instance, in TIN more points in rough areas and fewer points in smooth areas can be used to better represent the terrain's features, while in DEM the same number of points is the only way to represent both rough and smooth areas. Another advantage of TIN is that since it is a vector model, it is possible to obtain highly accurate data points. TIN also results in less storage than DEM since the number of points in TIN is adjustable. Because of its advantages, TIN has been adopted by several GIS and automated mapping and contouring software packages. Given that the accuracy in representing 3D objects is an important requirement and that the data model should make it possible for developing computational geometry algorithms for computing LOS, TIN is more suitable for GPSLoc than DEM.

3D databases have been discussed in the field of 3D GISs where geometric and semantic information has been incorporated into one model for spatial analysis (Maguire et al. 1991; Aronoff 1995). There are also 3D models such as 3D FDS (Formal Data Structure), TEN (Tetrahedral Network), and the cell tuple model (Zlatanova 2000) where 3D objects (e.g., buildings) are treated independent from the terrain, that is, they are located on a plane with no elevation information. These models can handle spatial relationships among objects existing in the database using SQL-type queries. In computing the intersection of an arbitrary line with a 3D object, both geometric and topological information are needed. In a simplified 3D database model, the height of a 3D object can be added to the *Z* value of the terrain. Conceivably such a data model may be obtained either by integrating an existing vector model for representing the terrain and a model for representing 3D objects, or by developing a new model that takes into account the primitives of both the terrain and 3D objects in one data structure. The main issue in the integrative approach is to develop an algorithm to link the two different models, each as a separate schema, together. For example, by using a TIN to represent terrain data and a TEN to represent 3D objects, there is a need for an algorithm that links the two models together. The main issue in the second approach, developing a new model, is to use a single data structure for both the terrain and 3D objects. For example, if the TIN model is considered for both types of data, there must be a method of incorporating 3D objects into it.

The first approach requires two data structures and two sets of retrieval procedures. This approach is inefficient in GPSLoc, which requires frequent interactions with the database. The second approach requires development of a new algorithm to combine terrain heights and 3D objects into a single model, but since it involves only one data structure and one set of retrieval procedures, it is more efficient for GPSLoc. Because of the advantages it offers, a new single database model, i.e., the second approach, is adopted in GPSLoc. The single model is an extension of the TIN model, thus we call it eXtended TIN (XTIN), by adding points that represent 3D objects to an existing TIN representing terrain heights. The result of XTIN is a 3D model made up of intraconnected triangles representing both the terrain and 3D objects. Since 3D objects may have regular shapes, such as a column or a cylinder, or irregular shapes, such as a football stadium, XTIN must support methods of including both shape types (regular and irregular). Therefore, the requirement of XTIN for GPSLoc is to include both regularly and irregularly shaped objects and to preserve the shapes of objects accurately. In the following, the construction method of XTIN in GPSLoc is discussed.

Extended Triangulated Irregular Network

The 3D data model in GPSLoc must represent geometric and topological information for both terrain heights and 3D objects. XTIN is a new data model which can be used for GPSLoc, and other applications requiring 3D databases. In GPSLoc, before XTIN is constructed, the TIN model for the area of interest is first generated. The algorithm for generating TIN in GPSLoc is based on the algorithm by Garland and Heckbert (1995). In this algorithm, first two triangles are generated using the four corner points of the DEM data set. Then all the remaining points in the DEM are used to find the best point, which is defined as the point

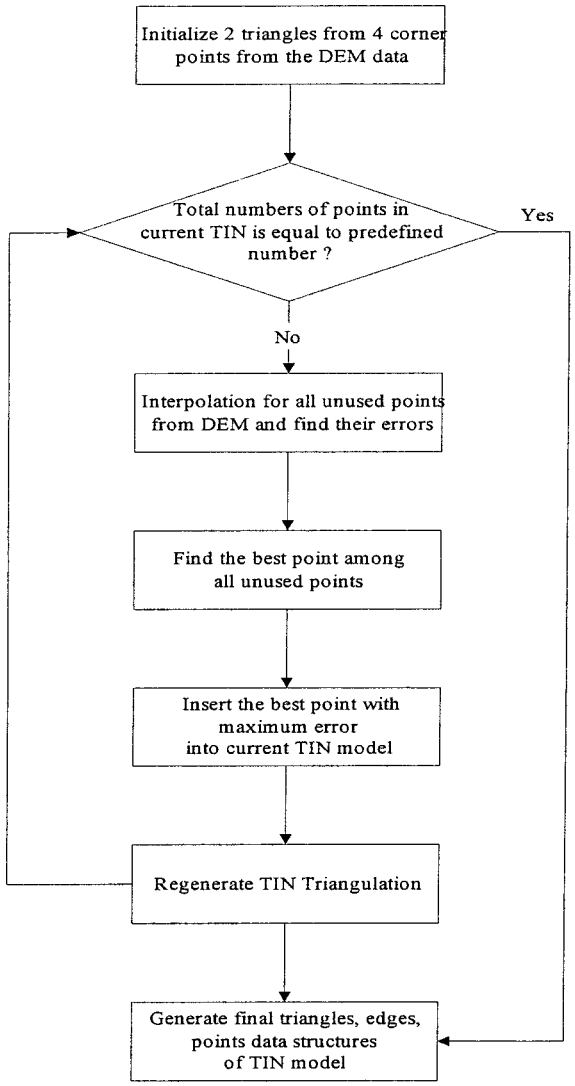


Fig. 2. Triangulated irregular network (TIN) generating algorithm

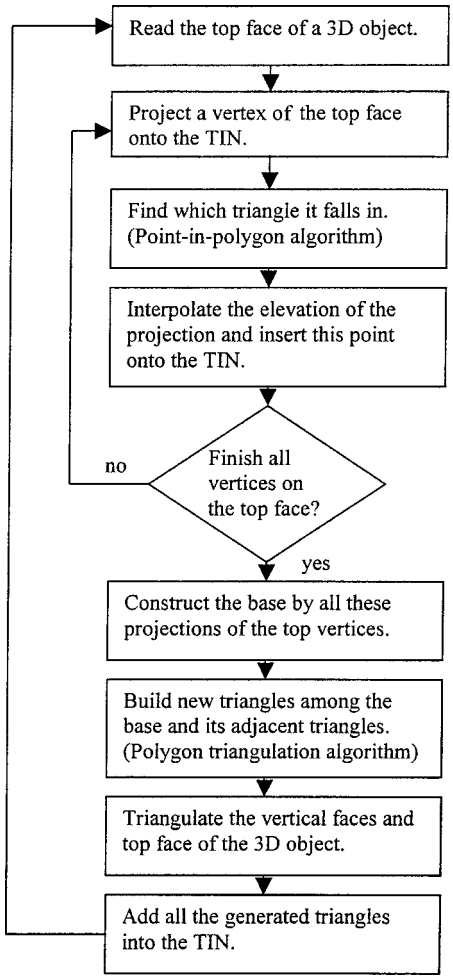


Fig. 3. Extended triangulated irregular network (XTIN) algorithm

with the largest interpolation error, to insert into the current TIN. The interpolation errors are calculated by taking the differences between the interpolated heights (using the *X*,*Y* coordinates of each DEM point in the corresponding location in the current TIN to determine the elevation value, i.e., *Z*) and the actual heights (the *Z* values in the DEM). The point with the largest interpolation error is chosen as the next point to be inserted to minimize the overall error in the TIN. One of the following two criteria can be used to terminate the triangulation process: the number of total selected points or the largest interpolation error of the TIN. In GPSLoc, the former criterion is chosen because it allows an easier way of controlling the total number of points and triangles in the final TIN model. The steps of this algorithm are shown in Fig. 2.

XTIN is built by incorporating the triangles from 3D objects into TIN. In doing so, it is expected that an object will intersect with one or more triangles in TIN. The important part in merging object triangles into TIN is how to update TIN (after the object is inserted) since this directly impacts the structure of the resultant triangle set and the performance of queries on XTIN. Discussed below are the three methods for this along with their strengths and weaknesses.

The first method is to put the two sets of triangles together without making any changes. This method is simple and fast to implement. The topology integrity in the resultant triangle set, however, is not maintained. In this method, a common edge between two triangles, one from TIN and one from object triangles, that intersect may not exist, which is a property for two neighboring triangles and important in traversing the triangle network and indexing. Furthermore, removing the triangles in TIN that are covered by objects could improve performance.

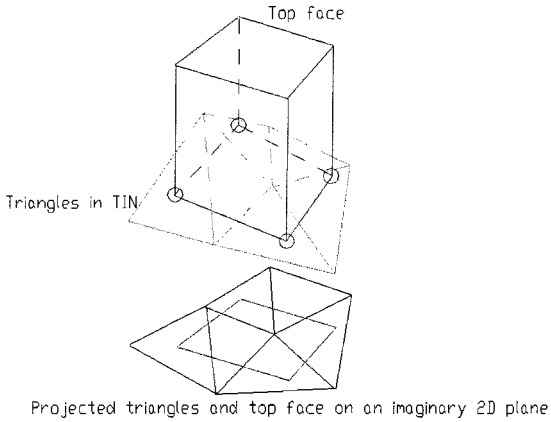


Fig. 4. Projecting top face of a three-dimensional object and triangulated irregular network (TIN) triangles

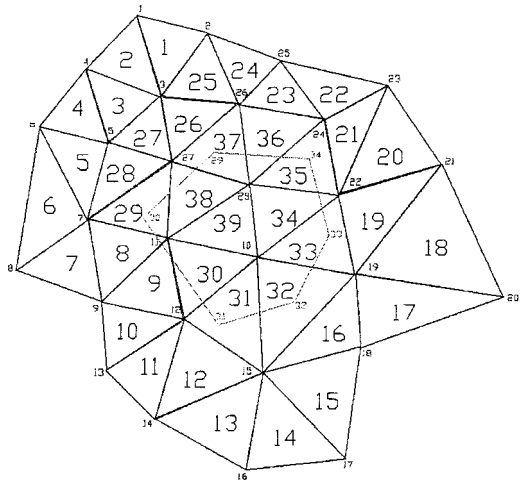


Fig. 5. Triangulated irregular network with a top face of a three-dimensional object

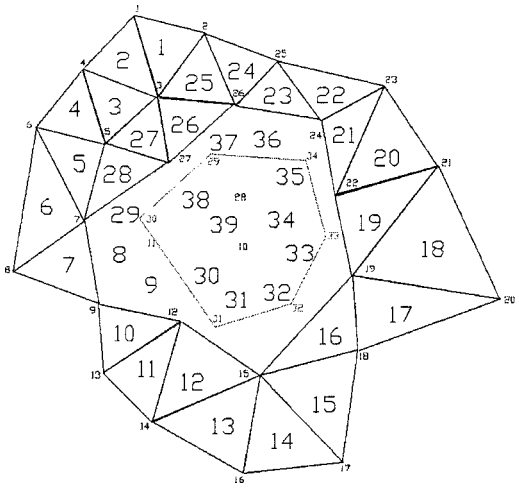


Fig. 6. Triangulated irregular network after some triangles are removed

The second method is to update TIN by inserting into it all the vertices on the base of an object as new points in TIN. This method seamlessly joins both triangle sets, from TIN and objects, and maintains the Delaunay triangulation property (Kreveld 1997) in TIN. But one major problem with this method is how to retain the shape of the 3D object's base while inserting new vertices into TIN, which is a necessary condition for the seamless join. There is no guarantee that any arbitrary connection between points will be preserved after they are inserted into TIN using Delaunay triangulation. Even though an algorithm may be devised to overcome this problem, the entire TIN may have to be recomputed each time a new point is added. This would make the computation very expensive, especially when working on a large area with many 3D objects.

The third method is to only update triangles in TIN around an inserting object. The base of a 3D object is formed by projecting its top face onto the TIN. All the edges of the adjacent triangles in TIN that intersect, touch, or are contained by the base are removed. New triangles are built in the area between the base and the edges left from the adjacent triangles. This method guarantees a seamless join of TIN and object triangles since the shape of the 3D object's base is maintained. All the unnecessary triangles in TIN covered by the base are removed to avoid redundant computation in query execution. The Delaunay triangulation property may be no longer valid in the newly generated triangles around the base if a polygon triangulation algorithm (e.g., O'Rourke 1994b) is used. The time performance of this method is better than the time performance of the second method since it only requires the triangles around objects and not the entire TIN. Considering the amount of triangles generated and updated in this method and the gain in computing time, this method is preferred over the other two methods and was chosen in our work. Fig. 3 shows the steps of the XTIN algorithm in GPSLoc, which is based on this method. Some of the details of the algorithm are described below.

In order to simplify the computation, a generalization on representation of 3D objects is adopted. Given the case of irregular object shapes, a 3D object is considered as a solid entity composed of its top horizontal face and all the vertical faces formed by projecting the top face onto the ground. This generalization saves time in constructing and triangulating 3D objects and is valid in most situations which simplifies constructing XTIN by using only the top face of the 3D objects.

After obtaining top faces, XTIN will be constructed by incorporating 3D objects into TIN. This is accomplished by projecting the top faces onto the triangles of the TIN and connecting the vertices of the top faces to the projection points in the TIN. In doing so, the projection points are identified first, and then the triangles in which these points fall are determined and are used to interpolate the elevation of the projection points. The approach taken in this work to construct XTIN first projects the top faces and the TIN triangles onto a 2D plane and then uses a point-in-polygon algorithm to find the triangles that contain the projection points. A horizontal plane passing through the lowest elevation point in the TIN was chosen as the 2D plane. A projection point on this 2D plane should have the same horizontal coordinates (i.e., X and Y) as its original vertex and the same vertical coordinate (i.e., Z) as the 2D plane. This projection process is illustrated in Fig. 4, where the top face of a 3D object (a cube) and the TIN triangles are projected onto an imaginary 2D plane in the bottom. The four circles illustrate the corners in the base of the cube, which is the projection of the top face onto the TIN. After all the points are on the 2D plane, the point-in-polygon algorithm by

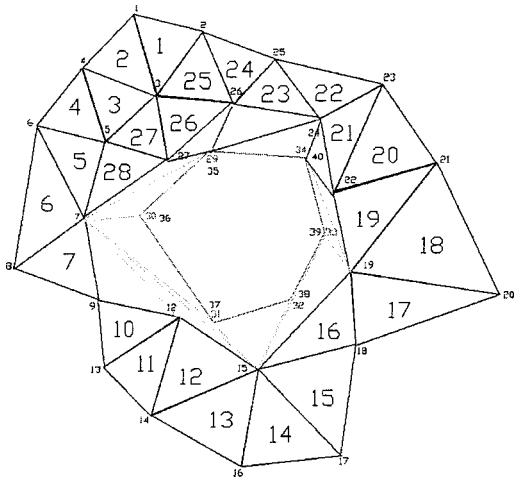


Fig. 7. Triangulated irregular network after polygon triangulation

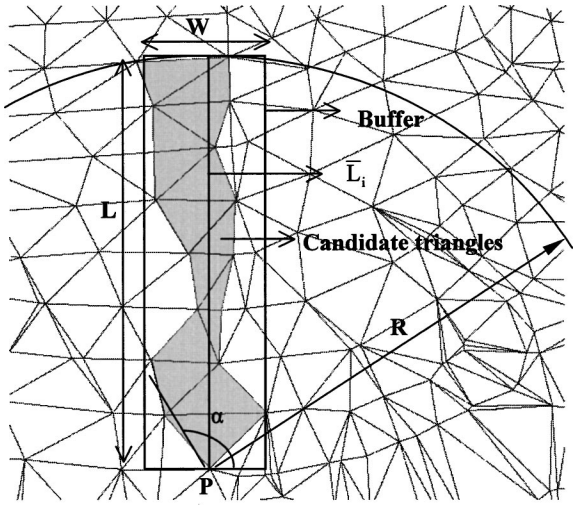


Fig. 8. Projected extended triangulated irregular network

O'Rourke (1994a) is used to find the triangles each projection point falls in. This algorithm draws a ray r from a point q in an arbitrary direction parallel to one of the axes and counts the number of intersections of r with the polygon p. The point q is outside of p if the number of intersections is even, otherwise q is in p. After finding which triangle a vertex is projected on, the vertical coordinate (height) of the projection on that triangle can be computed by introducing the coordinates of the projection to the equation of triangle plane in point normal form (Hill 2001). The normal of a plane can be computed as the cross product of two vectors which are composed by three points in that plane. Suppose the normal is $(x0,y0,z0)$, thus the plane can be represented as a dot product including its normal and a point on this plane, e.g., the projection $(x1,y1,z1)$, as follows:

$$(x_0,y_0,z_0)\cdot[(x_1,y_1,z_1)-(x_0,y_0,z_0)]=0 \qquad (1)$$

The height of the projection (z_1) then can be solved from this equation.

When the elevation points of all base vertices are computed, a 3D object can be constructed. Since the projection points are added into triangles in TIN, TIN has to be updated accordingly.

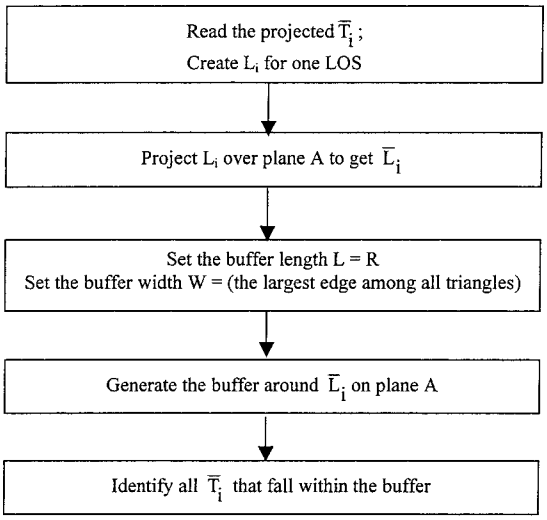


Fig. 9. Filter algorithm

As discussed before, in this work the method that updates a portion of TIN is used. This procedure is illustrated in Figs. 5, 6, and 7. Fig. 5 shows a top face of a 3D object and selected TIN triangles in 2D. First, all the triangles that intersect, touch, or are contained by the base are retrieved from the TIN. If a triangle is contained by the base (no touch) or its two vertices are within the base, then it is removed from the TIN. If only one vertex is within the base, then the edge on this triangle opposite to that vertex is saved into an edge set and the triangle is removed. Fig. 6 shows the result after the triangles are removed. After processing all the retrieved triangles, the edge set forms a polygon which contains the base. With the exception that the base is fully contained by a triangle (no touch on edges), the triangle becomes the polygon. The area to be triangulated is a ring shape between the polygon and the base. The ring can be divided into two polygons by connecting two pairs of vertices from the base and the polygon. The polygon triangulation algorithm by O'Rourke (1994b) is used in GPSLoc. In each iteration, this algorithm attempts to draw a diagonal in the polygon that does not intersect with any edge in the polygon. If one diagonal is found, the algorithm cuts the polygon into two pieces along the diagonal and continues on each part until it reaches a triangle. The time complexity of this algorithm is $O(N^3)$. Fig. 7 shows the new triangles generated by the polygon triangulation algorithm.

The vertical faces of a 3D object can be built by connecting the vertices in the top face and the corresponding projections on the existing TIN in that order. The vertical faces are all quadrilateral, thus either pair of diagonal vertices of the quadrilateral can be used to triangulate the face. The top face can be triangulated using the same polygon triangulation algorithm described above. The new triangles are added into the existing TIN and the TIN becomes the XTIN. In the current version of GPSLoc, no sorting or indexing techniques are used for TIN or XTIN.

Line of Sight Intersection

LOSI is a set of algorithms in GPSLoc designed to predict satellite visibility. The basic requirements of LOSI include a user's position and time and the XTIN of the area surrounding the user's position. Due to the large number of triangles in XTIN, finding the triangles that obstruct LOS involves an extensive database retrieval process. Therefore, special sorting and indexing techniques are used in GPSLoc to improve the performance of LOSI.

The objective of the *visibility* parameter in GPSLoc is to determine which satellites, out of those available, have LOS with the user's location at a given time. Once the visibility issue is resolved, that is, once the specific satellites visible to the receiver are determined, GPSLoc can proceed with the other three GPS QoS parameters. LOSI uses the satellites' and the receiver's positions, the time the GPS data are needed, and the XTIN of the geographic extent surrounding the receiver's position. TIN-based terrain databases typically occupy several gigabytes of storage (Guedes et al. 1997), which impacts data retrieval and computation. Since the XTIN triangles closely surrounding the user's location may obstruct LOS, the performance of LOSI can be improved by using only those triangles that are close to the receiver. In LOSI, an algorithm called *Range_Query* is used to filter out the most likely triangles from the XTIN after a user's location is given. The selection of these triangles (candidate triangles) within a range is important since it helps manipulate a very large database (Brinkhoff 1993) while satisfying the performance requirement of GPSLoc. The value of the range may vary from the *static*

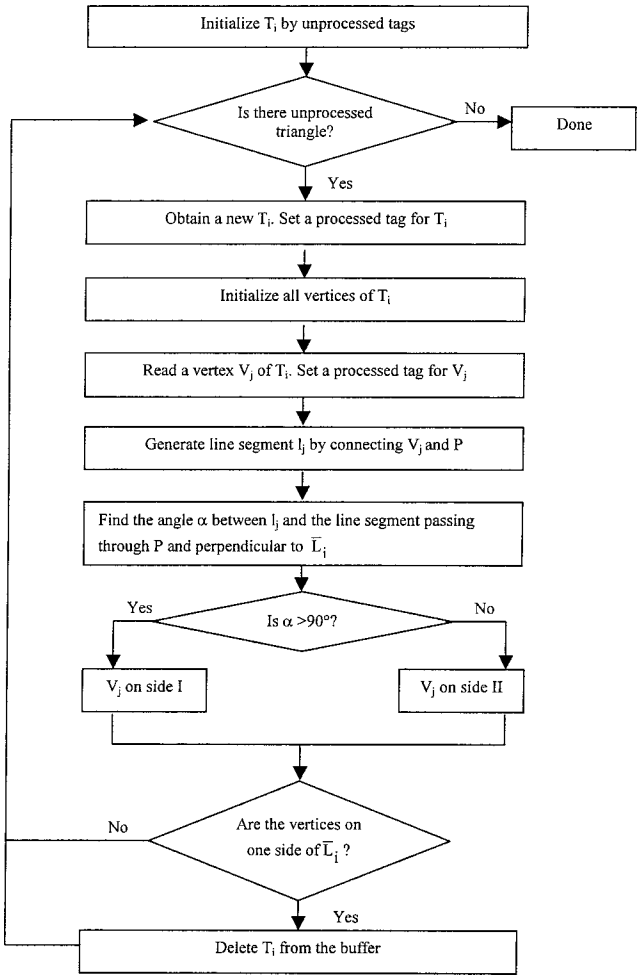


Fig. 10. Candidate algorithm

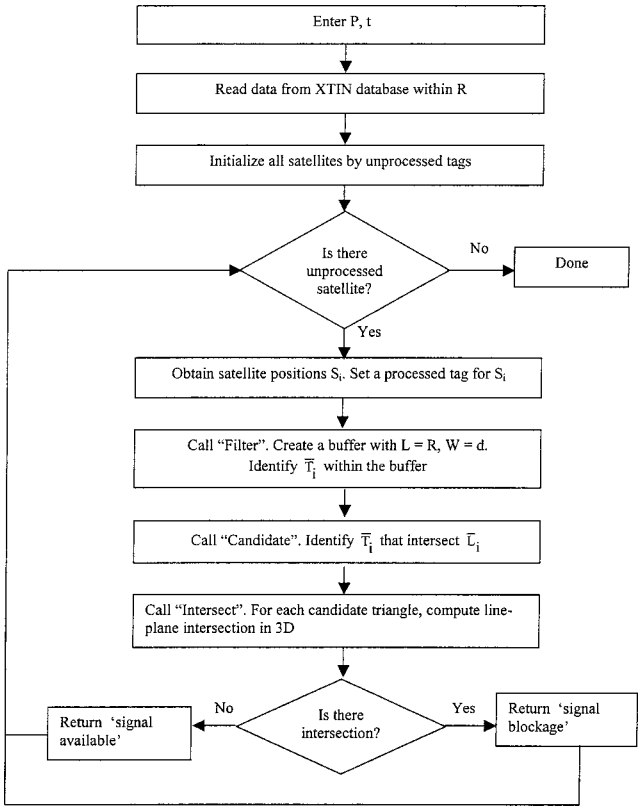


Fig. 12. Line of sight intersection algorithms

mode to the *dynamic mode* as it impacts the subsequent computations; in the *static mode* a large range may be selected, while in the *dynamic mode* only a small range may be feasible.

Spatial indexing is another way to improve the performance of retrieving data (triangles) from the database. The quadtree spatial indexing (Samet 1990a,b) is a widely used technique in GISs and

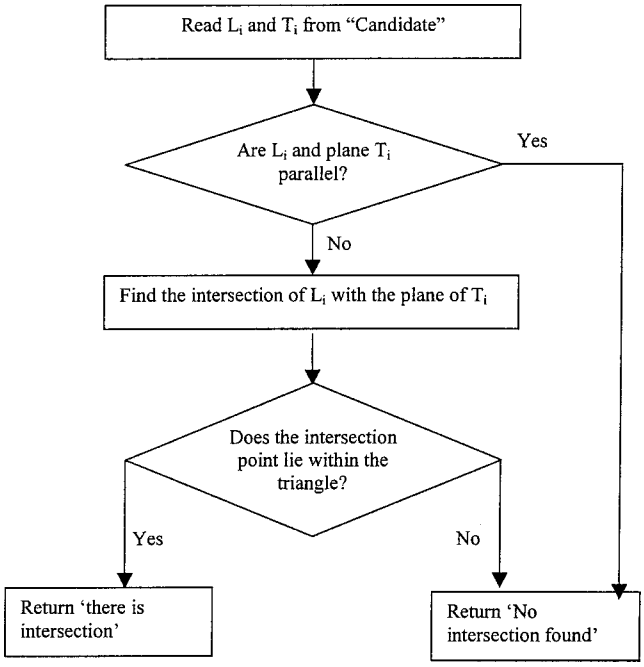


Fig. 11. Intersect algorithm

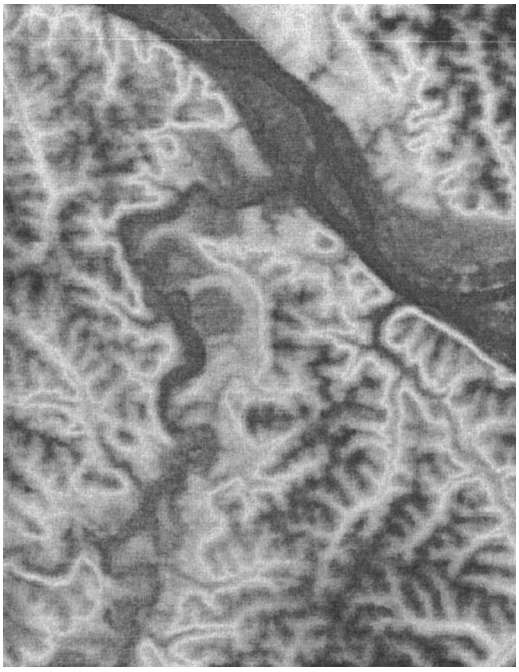


Fig. 13. Digital elevation model image used in GPSLoc prototype

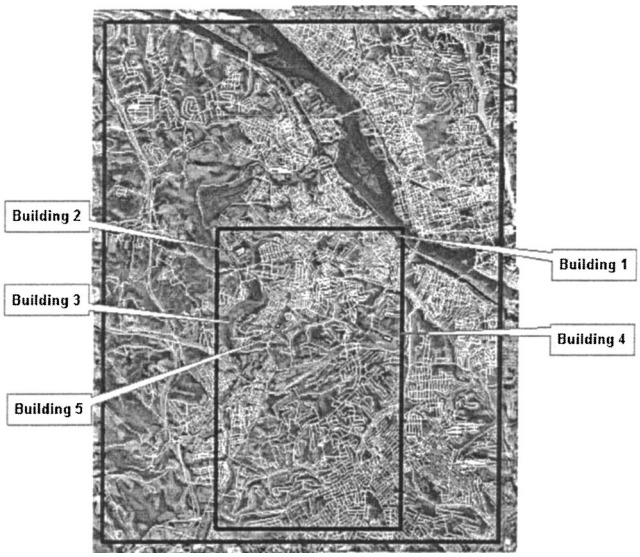


Fig. 14. Test area

spatial database systems and is used to index the XTIN in GPSLoc. To index the XTIN in GPSLoc using the quadtree technique, the entire area of interest is initially divided into a set of four equal quads (or cells), and each quad is subdivided into four subquads, and the process continues for several levels where they are organized into a quadtree hierarchy. Since the triangles are projected onto a plane, each cell in the plane would contain a set of projected triangles. Triangles may overlap multiple cells; these are the triangles whose vertices are contained within more than one cell. Although this results in redundancy, it guarantees that LOSI will not miss any triangle that may obstruct LOS. In computing the projection of a vertex, first by traversing the quadtree, the cell in which the vertex is located is found. Then all the triangles that fall within that cell are tested to check if the vertex is projected onto it; the point-in-polygon algorithm (O'Rourke

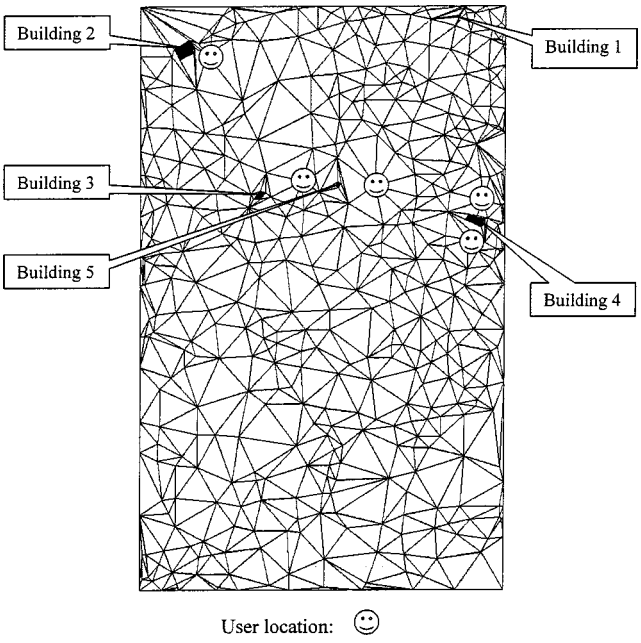


Fig. 15. Extended triangulated irregular network of test area

Table 1. User's location

ID	East (m)	North (m)	Elevation (m)
1	579,360.0	4,475,490.0	280.0
2	580,350.0	4,475,430.0	302.0
3	581,610.0	4,474,680.0	323.0
4	581,820.0	4,475,220.0	326.0
5	578,070.0	4,477,230.0	232.0

1994a) discussed earlier can be used for this. This approach reduces the range of searching for triangles into a small area and improves the overall performance of GPSLoc.

The *Range_Query* algorithm results in a list of interconnected triangles which contain much fewer triangles than the original XTIN. Suppose a GPS receiver at position P and a satellite at position S_i; the LOS is the line connecting P and S_i. A satellite is visible to the receiver if the line segment that joins them does not intercept any terrain features, such as mountains, or 3D objects, such as buildings (Kreveld 1997). To determine visibility, the line segment $\overline{PS_i}$ and all triangles in the cell are computed for intersection. If $\overline{PS_i}$ does not intersect any of the triangles, then there is a LOS between the GPS receiver and the satellite. If $\overline{PS_i}$ and at least one triangle intersect, then there is no LOS between the GPS receiver and the satellite. This approach is not only simple to implement, it also supports a broad range of operations.

LOSI uses two algorithms called *Filter* and *Candidate* to further select candidate triangles from those produced by *Range_Query*. After candidate triangles are identified, the *Intersect* algorithm is used to compute LOS. These algorithms are invoked in order in LOSI to compute visibility. The following notations are used in the LOSI algorithms:

- User's position: P,
- User's time: t,
- Satellite position: S_i ($i=1,2,3,\ldots$),
- Range: R,
- LOS connecting P and S_i (3D): L_i,
- Projected L_i (2D): $\overline{L_i}$,
- Buffer length: $L=R$,
- Buffer width: W,
- XTIN triangle (3D): T_i, and
- Projected XTIN triangle (2D): $\overline{T_i}$.

Because 2D intersections are computationally less intensive than 3D intersections, the selection of candidate triangles in GPSLoc is carried out on the same plane onto which the XTIN is projected. Point data can be projected simply by neglecting the elevation data (i.e., the z value). Fig. 8 depicts an example of the projected XTIN used to select triangle candidates.

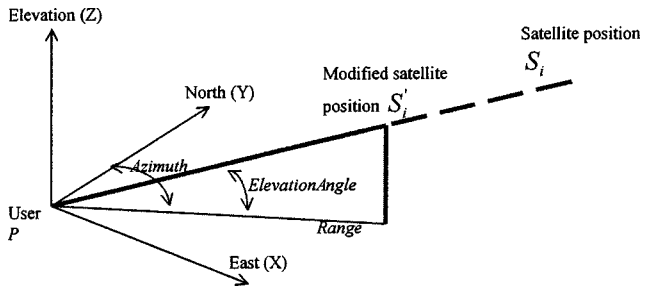


Fig. 16. User's location, satellites' locations, and line of sight

Table 2. Available satellites to user's location (ID=1)

Satellite	Elevation angle (deg)	Azimuth (deg)	Range (m)	*SatX* (m)	*SatY* (m)	*SatZ* (m)
GPS BII-02 (PRN 02)	6.0	96.2	5,000	584,330.75	4,474,950.00	805.52
GPS BIIA-25 (PRN 03)	18.5	52.3	5,000	583,316.12	4,478,547.64	1,952.98
GPS BIIA-20 (PRN 07)	3.7	232.2	5,000	575,409.22	4,472,425.46	603.34
GPS BIIA-28 (PRN 08)	63.5	300.3	5,000	575,043.02	4,478,012.64	10,308.45
GPS BIIR-03 (PRN 11)	39.7	143.1	5,000	582,362.10	4,471,491.58	4,431.08
GPS BIIR-02 (PRN 13)	12.2	204.7	5,000	577,270.66	4,470,947.46	1,361.04
GPS BIIA-14 (PRN 26)	1.8	332.9	5,000	577,082.28	4,479,941.06	437.13
GPS BIIA-15 (PRN 27)	75.8	209.8	5,000	576,875.13	4,471,151.17	20,039.81
GPS BIIR-05 (PRN 28)	31.2	290.2	5,000	574,667.53	4,477,216.49	3,308.11
GPS BIIA-17 (PRN 29)	9.1	322.2	5,000	576,295.46	4,479,440.78	1,080.87

The steps of the *Filter* algorithm are shown in Fig. 9. This algorithm is used to filter out those triangles that are candidate triangles by creating a buffer around the projected LOS (L_i in Fig. 8). The length of the buffer, *L*, is the length of the range selected and the width of the buffer, *W*, is selected as the length of the largest edge among all triangles in the XTIN. The choice of *W* is important because a large value of *W* may result in a low performance and a small value of *W* may eliminate some candidate triangles.

The steps of the *Candidate* algorithm are shown in Fig. 10. This algorithm is used to further refine the list of possible candidate triangles. The premise of the algorithm is that a triangle T_i whose projection T_i does not intersect the projected LOS, L_i, does not intersect the LOS, L_i. To determine whether T_i intersects L_i, a line connecting a vertex of T_i and the user's location can be drawn and the angle α between this line and the buffer line passing through the user's position can be calculated (see Fig. 8). If all three such angles for T_i (one for each vertex) are greater than or smaller than 90°, then T_i is on one side of L_i and does not intersect L_i. Those triangles are ruled out to intersect in 3D and are therefore discarded. The triangles left in the buffer are the candidate triangles, which will be the input to the *Intersect* algorithm.

The steps of the *Intersect* algorithm (Bourke 1997) are shown in Fig. 11. This algorithm is used to compute intersections between the line segment, L_i, and the candidate triangles. To summarize, LOSI is a collection of the previous algorithms to compute satellite visibility (Fig. 12). LOSI first reads the user's position and time. Then it selects a proper range and calls the *Project* algorithm to project the XTIN onto a 2D plane. It then calls the *Filter* and *Candidate* algorithms to select the candidate triangles. Finally, the *Intersect* algorithm is called to compute LOS. For a given location and time, LOSI determines which satellites are visible to the user's position. Once the visible satellites are identified, GPSLoc can proceed with determining further QoS issues such as finding the best solution by using the geometric dilution of precision (DOP) of the visible satellites. A higher DOP indicates poor satellite geometry and an inferior measurement configuration.

Experimentation

In order to test the methodologies and algorithms described in this paper, we have developed a GPSLoc prototype to simulate the *visibility* parameter. All the algorithms in the prototype were coded in the *Java* programming language. The TIN model of a portion of Pittsburgh City was built using the DEM of the area which was obtained from the Pennsylvania Spatial Data Access website ⟨http://www.pasda.psu.edu⟩. The DEM was a 7.5-min raster image in the Universal Transverse Mercator (UTM) projection and the NAD'27 datum with 30-m resolution (i.e., the data spacing is 30 m by 30 m). Fig. 13 depicts the DEM image used in the GPSLoc prototype.

Fig. 14 shows the test area for which the TIN model was built. For spatial orientation, the DEM, the street network, and an aerial photo of the area are overlaid. The small window in the figure represents the study area in our experimentation, which contains 268 rows and 166 columns of data points. Because the space between the data points is 30 m, the easting length and the northing length are 4,950 and 8,010 m, respectively, and the area covered is 39,649,500 square meters.

Once the TIN model of the test area was built, the XTIN model was constructed using the XTIN algorithm with five buildings (see Fig. 14). The locations of the buildings were spread out in the area. Each building consists of a top face and vertical faces formed by projecting the top face onto a horizontal plane. Fig. 15 shows the top view of the XTIN constructed for the test area.

In the experimentation, five user locations were used (see Fig. 15). Table 1 shows the UTM coordinates of these five locations. Given these user locations and a time, the satellite visibility to the user is computed. The location of a satellite may be in a different UTM zone than the user's, in which case a large portion of the LOS connecting the user location and the satellite is outside the test area. The portion of the LOS whose projection lies in the test area was taken into account. Fig. 16 shows the relative position of the satellite S_i to the user, where *Azimuth* is the angle between the projected LOS and the northing, *Elevation angle* is the angle between the LOS and the projected LOS, and the cutoff point on the LOS is called modified satellite position S_i'. The line segment connecting the user and S_i' represents the portion of the LOS and its projection was used as a range to retrieve XTIN data from the database. The following equations are used to determine the UTM coordinates of S_i':

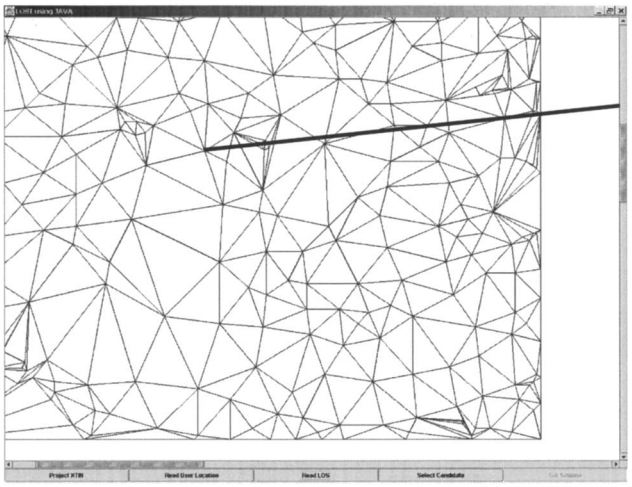
$$SatX = UserX + R\sin(Azimuth) \tag{2}$$

$$SatY = UserY + R\cos(Azimuth) \tag{3}$$

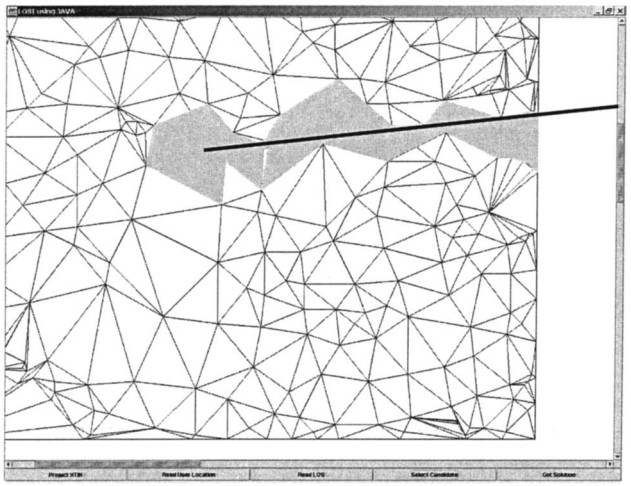
$$SatZ = UserZ + R\tan(Elevation\ \ angle) \tag{4}$$

where R=range; $SatX$=easting coordinate (X); $SatY$=northing coordinate (Y); and $SatZ$ (Z)=elevation value.

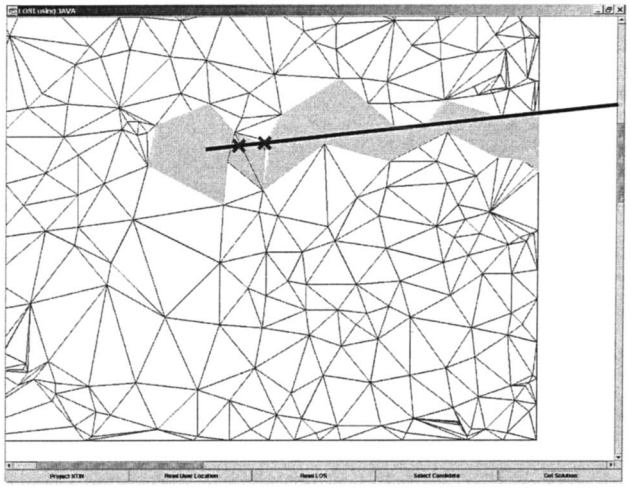
For testing, the time at 18:54 GMT on November 19, 2002



(a)



(b)



(c)

Fig. 17. (a) Expected line of sight; (b) selection of candidate triangles; and (c) computed line of sight

Table 3. Line of Sight Intersection Results

Satellite	Signal blockage
(a) Location 1	
GPS BII-02 (PRN 02)	Yes
GPS BIIA-25 (PRN 03)	No
GPS BIIA-20 (PRN 07)	Yes
GPS BIIA-28 (PRN 08)	No
GPS BIIR-03 (PRN 11)	No
GPS BIIR-02 (PRN 13)	No
GPS BIIA-14 (PRN 26)	No
GPS BIIA-15 (PRN 27)	No
GPS BIIR-05 (PRN 28)	No
GPS BIIA-17 (PRN 29)	No
(b) Location 2	
GPS BII-02 (PRN 02)	No
GPS BIIA-25 (PRN 03)	No
GPS BIIA-20 (PRN 07)	Yes
GPS BIIA-28 (PRN 08)	No
GPS BIIR-03 (PRN 11)	No
GPS BIIR-02 (PRN 13)	No
GPS BIIA-14 (PRN 26)	Yes
GPS BIIA-15 (PRN 27)	No
GPS BIIR-05 (PRN 28)	No
GPS BIIA-17 (PRN 29)	No
(c) Location 3	
GPS BII-02 (PRN 02)	Yes
GPS BIIA-25 (PRN 03)	No
GPS BIIA-20 (PRN 07)	No
GPS BIIA-28 (PRN 08)	No
GPS BIIR-03 (PRN 11)	Yes
GPS BIIR-02 (PRN 13)	No
GPS BIIA-14 (PRN 26)	No
GPS BIIA-15 (PRN 27)	No
GPS BIIR-05 (PRN 28)	No
GPS BIIA-17 (PRN 29)	No
(d) Location 4	
GPS BII-02 (PRN 02)	No
GPS BIIA-25 (PRN 03)	No
GPS BIIA-20 (PRN 07)	No
GPS BIIA-28 (PRN 08)	No
GPS BIIR-03 (PRN 11)	No
GPS BIIR-02 (PRN 13)	No
GPS BIIA-14 (PRN 26)	Yes
GPS BIIR-05 (PRN 28)	No
GPS BIIA-17 (PRN 29)	No
(e) Location 5	
GPS BII-02 (PRN 02)	No
GPS BIIA-25 (PRN 03)	No
GPS BIIA-20 (PRN 07)	Yes
GPS BIIA-28 (PRN 08)	No
GPS BIIR-03 (PRN 11)	No
GPS BIIR-02 (PRN 13)	No
GPS BIIA-14 (PRN 26)	Yes
GPS BIIA-15 (PRN 27)	No
GPS BIIR-05 (PRN 28)	No
GPS BIIA-17 (PRN 29)	No

was chosen and a list of satellites' locations for each of the five user's locations from http://sirius.chinalake.navy.mil/satpred was obtained. With a range of 5,000 m, the resulting modified satellites' locations using Eqs. (2)–(4) for the user's location with ID=1 (579,360.0,4,475,490.0,280.0) are listed in Table 2.

After the XTIN data, the user's locations, and the modified satellites' positions were determined, they were used in the GPSLoc prototype to test satellite visibility for each user location using the LOSI algorithms. First, each LOS between one user location and one available satellite was projected onto the plane on which the XTIN was constructed. Then a range was generated to select triangle candidates that have the potential to block the LOS using the *Filter* and *Candidate* algorithms. Finally, each of the triangle candidates was tested using the *Intersect* algorithm to determine whether it intersects the projected LOS; if they intersect, there is no LOS, otherwise there is a LOS. Fig. 17 shows one example of the process of computing LOS, where the LOS from Satellite GPS BII-02 (PRN 02) is shown. The results of LOSI for the user's location (ID=1) are provided in Table 3(a). There was a total of 10 satellites that were available at the user' location and the given time. LOSI determined that of these available satellites, two satellites—GPS BII-02 (PRN 02) and GPS BIIA-20 (PRN 07)—were obstructed from view [see Table 3(a)]. The experimentation results for the user's location with ID=2–5 (see Table 1) are provided in Tables 3(b–e). As shown in these tables, there is at least one satellite that was obstructed for each user's location.

Conclusions and Future Work

In this paper, GPSLoc was proposed as a framework that provides predictive GPS QoS to users and applications. It is argued that GPSLoc plays a key role in the operation and success of many existing and emerging engineering applications. The 3D database component of GPSLoc requires highly accurate data, an optimized model integrating both terrain heights and 3D objects, and optimized storage and manipulation techniques. The LOSI component of GPSLoc requires a set of optimized algorithms for effective and accurate computation of GPS visibility. It is obvious, as discussed and demonstrated in the paper, that visibility is a precondition for the other GPS QoS parameters.

While several issues and requirements for GPS QoS within the context of the GPSLoc framework were discussed in this paper, there are many open questions and issues that need to be researched. We envision that future research in GPS QoS and GPSLoc should include development of methodologies, models, and algorithms for the other three GPS QoS parameters, namely accuracy, reliability, and flexibility; development of metrics to measure GPS QoS; and development of strategies for effective deployment of GPSLoc in different engineering applications in order to meet the specific requirements (e.g., storage and real-time processing) of each application.

References

Aronoff, S. (1995). *Geographic information systems: A management perspective*, WDL Publications, Ottawa.

Bourke, P. (1997). ''Determining whether a line segment intersects a 3 vertex facet.'' ⟨http://astronomy.swin.edu.au/~pbourke/geometry/linefacet/⟩.

Brinkhoff, T., Horn, H., Kriegel, H., and Schneider, R. (1993). ''A storage and access architecture for efficient query processing in spatial database systems.'' *3rd Int. Symp. on Large Spatial Databases*, 357–376.

Garland, M., and Heckbert, P. S. (1995). ''Fast polygonal approximation of terrains and height fields.'' *Technical Rep. CMU-CS-95-181*, Computer Science Dept., Carnegie Mellon Univ., Pittsburgh.

Guedes, L. C. C., Gattass, M., and Carvalho, P. C. P. (1997). ''Real-time rendering of photo-textured terrain height fields.'' *Proc. of Brazilian Computer Graphic Symp.*, Sibgrapi, 18–25.

Hill, F. S., Jr. (2001). *Computer graphics using OpenGL*, Prentice Hall, Upper Saddle River, N.J., 178.

van Kreveld, Marc. (1997). ''Algorithms for triangulated terrains.'' *Conf. on Current Trends in Theory and Practice of Informatics*, 19–36.

Lo, C. P., and Yeung, Albert, K. W. (2002). *Concepts and techniques of geographic information systems*, Prentice Hall, Englewood Cliffs, N.J., 307–311.

Maguire, D. J., Goodchild, M. F., and Rhind, D. W. (1991). *Geographic information system: Principles and applications*, Longman Scientific & Technical, London.

Miller, C., and La Flamme, R. A. (1958). ''The digital terrain model—Theory and applications.'' *Photogramm. Eng.*, 24(3), 433–442.

O'Rourke, Joseph (1994a). *Computational geometry in C*, Cambridge University Press, Cambridge, U.K., 233–236.

O'Rourke, Joseph (1994b). *Computational geometry in C*, Cambridge University Press, Cambridge, U.K., 41.

Pathfinder. (1999). *GPS Pathfinder Office Documentation*, Trimble Navigation Limited.

Peucker, T. K., Fowler, R. J., Little, J. J., and Mark, D. M. (1978). ''The triangulated irregular network.'' *Proc. of American Society of Photogrammetry: Digital Terrain Models (DTM) Symp.*, St. Louis, 516–540.

Samet, H. (1990a). *Applications of spatial data structures: Computer graphics, image processing and geographical information systems*, Addison-Wesley, Reading, Mass.

Samet, H. (1990b). *The design and analysis of spatial data structures*, Addison-Wesley, Reading, Mass.

Zlatanova, S. (2000). *3D GIS for urban development*, ITC dissertation, Number 69, 83–91.