REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	10 August 1998	Final

4. TITLE AND SUBTITLE

Site Specific Propagation Prediction Software For Communication Channel Modeling

5. FUNDING NUMBERS

DAAG55-98-C-0019

6. AUTHOR(S)

George Liang

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Site Ware Technologies, Inc.
439 10th Street
Brooklyn, NY 11215

19981222 043

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

ARO 37901.1-EL-SBI

11. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy or decision, unless so designated by other documentation.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved For Public Release; Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

Report developed under SBIR contract.
Site Ware Technologies proposes to develop a computer package that will be able to predict the propagation characteristics of radio signals, such as: path loss, time delay profile, angle of arrival distribution, spatial correlation and depolarization. The program will give site specific predictions in a wide range of outdoor physical environments for frequencies in the range from about 100MHz to 10 GHz. The software tool will utilize a novel technique known as the Vertical Plane Launch (VPL) method for predicting the propagation paths in a cluttered building environment. The VPL method has previously demonstrated the capability of accounting for reflections from a vertical wall and diffractions from the edge of buildings. The objectives is to investigate and demonstrate enhancements to the existing VPL kernel, including the addition of terrain, foliage effects and diffuse scattering from surfaces and objects in the vicinity of the mobile, as well as its integration with an user interface and visualization software.

14. SUBJECT TERMS

SBIR Report, VPL, Propagation Prediction

DTIC QUALITY INSPECTED 3

15. NUMBER OF PAGES 14

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	

Site Ware Technologies, Inc.

Final Report

to

U.S. Army Research Office

on

Site Specific Propagation Prediction Software Tool For Communication Channel Modeling

Contract No. DAAG55-98-C-0019

George Liang

August 1, 1998

1 Introduction

The technical objective for Phase I was to demonstrate the feasibility of extending the VPL kernel in order to develop a full function site specific propagation software tool. The VPL kernel which was developed prior to this SBIR effort and represents a substantial improvement in the state of the art deterministic propagation models. The VPL method can determine the full 3D propagation paths between a source and receiver point which can include multiple specular reflected and edge diffracted interactions. Although the VPL method can accurately characterize the multi-path environment of propagating around buildings it is deficient in its ability to account for other types of scatterers in the physical environment. Notably, the VPL method was not able to account for the effects of the terrain on the propagation path.

In addition to the electromagnetic problem there were a number of computational limitations, primarily associated with the very long execution time of a single simulation. In order to develop a truly commercial propagation prediction software tool that might be accepted by industry it was imperative that the total execution time be dramatically reduced. Therefore a number of different algorithms were studied and implemented to achieve the reduction in the runtime.

Finally, it is necessary to develop a graphical user interface as the front end to the computational engines. The ability of the user to visualize, generate and edit the input databases and view the results of the simulation in a convenient method will enhance the usefulness of the software tool. A portion of the tasks in this Phase I effort has been focused on the development of the algorithms and techniques to display and manipulate the input and output data on screen.

The following objectives were outlined in the Phase I proposal and designated the major technical areas that were investigated during the Phase I work period.

Objective 1: The implementation of the a terrain reflection and blockage model that interacts with an elevation database to find the effects of the terrain along a ray path.

Objective 2: Investigate techniques and algorithms for modeling rough surface scattering from the ground. This work will lay the groundwork for the implementation of a model that can account for the multipath due to scattering from terrain not along a ray path.

Objective 3: The implementation of several algorithmic improvements to the VPL kernel to reduce the computation time. The increased efficiency of the ray tracing kernel will enable the addition of several new features as well as increase the utility of the software tool.

Objective 4: Investigate and develop a set of processes that will enable the user to extract the needed information from USGS databases to produce a propagation features database.

Objective 5: Combine the ray tracing propagation kernel with GUI operating on a Microsoft Windows NT platform to facilitate the input and output tasks of the end user.

Objective 6: Additional functionalities will be added to the GUI front end and the visualization software to enhance the capabilities of the propagation tool to extract and display the relevant and necessary data produced by the ray tracing kernel.

The completion of the Phase I goals will lay the framework for the development of a commercially viable software package in Phase II.

2 Terrain Reflection and Blockage

The effect of terrain on the propagation path must be considered when the variations in the ground elevation are significant. For the case when the terrain has very small variations, it is sufficient to model the terrain as completely flat. In this simplest case only one ray pair consisting of the ray which does not hit the ground and the ray that undergoes one reflection from the ground as it travels from the transmitter to the receiver. In the other case, when the terrain variation is large and the angle of the ground reflected path is significantly different from the flat ground case or the propagation path may be obscured by the ground thereby producing diffraction effects. In this case it is necessary to obtain a more accurate picture of terrain beneath the ray by extracting the terrain profile along the ray path from a digital elevation database and determining its effect on the ray path.

2.1 Extracting the Terrain Profile from the Data Base

An efficient method to find the terrain profile has been implemented and integrated with the VPL kernel. This method employs a table lookup of the four nearest elevation points to a point along the ray path. Therefore instead of doing a sequential search of the terrain database which is very time consuming for large databases, it is possible to quickly extract the terrain elevation profile for points along the ray path. When the four closest points are found the elevation for the point on the ray path is determined using a bilinear interpolation scheme.

2.2 Terrain Correction Factor

Once the terrain profile has been found it must be examined to determine its effect on the propagation path. The terrain correction factor is calculated base on one of the follow three conditions:

1. The terrain completely obstructs the ray path between the source and the receiver, the source and the first horizontal edge or the last horizontal edge and the receiver.

2. The approximated sloping plane determined by the last four terrain points is sloping upward as one moves towards the receiver.

3. The approximate slope plane of the last four terrain points is sloping downward as one moves towards the receiver.

For the first condition the loss factor is approximated as a single knife edge diffraction at the crest of the obstructing terrain. The loss factor for the second condition is found by allowing for a single reflection from the sloping plane that is approximated from the terrain points. For the third condition a single reflection from the sloping plane is found only if the slope of the plane does not exceed the height of the source when it is extended back. Finally if none of the three conditions exists then the the loss factor is approximated for the highest terrain point that partially interferes with the first Fresnel zone but does not complete obstruct the ray path.

3 Terrain Scattering

Rough surface scattering from the terrain is a difficult phenomena to account for since the energy received at a point can come from all directions and its magnitude is dependent on the source of illumination at the surface. Therefore, in order to determine the diffuse scattering from the terrain it is necessary to subdivided the ground into small patches to find the illumination on the patch from the source and then calculate the re-radiated energy of the patch at the observation point. Figure 1 shows an example of the method used to model diffuse scattering from the rough ground. Initially the ground must be subdivided into appropriate size patches. The average size of each patch is determined by the average surface roughness and the simulation frequency. Each patch is represented by a point at its center which determines if the patch can be illuminated by the source. If the patch is in the line of sight of the source the magnitude and direction of the illumination source is calculated and stored.

The scattered energy at the observation point is then found by determining all the illu-

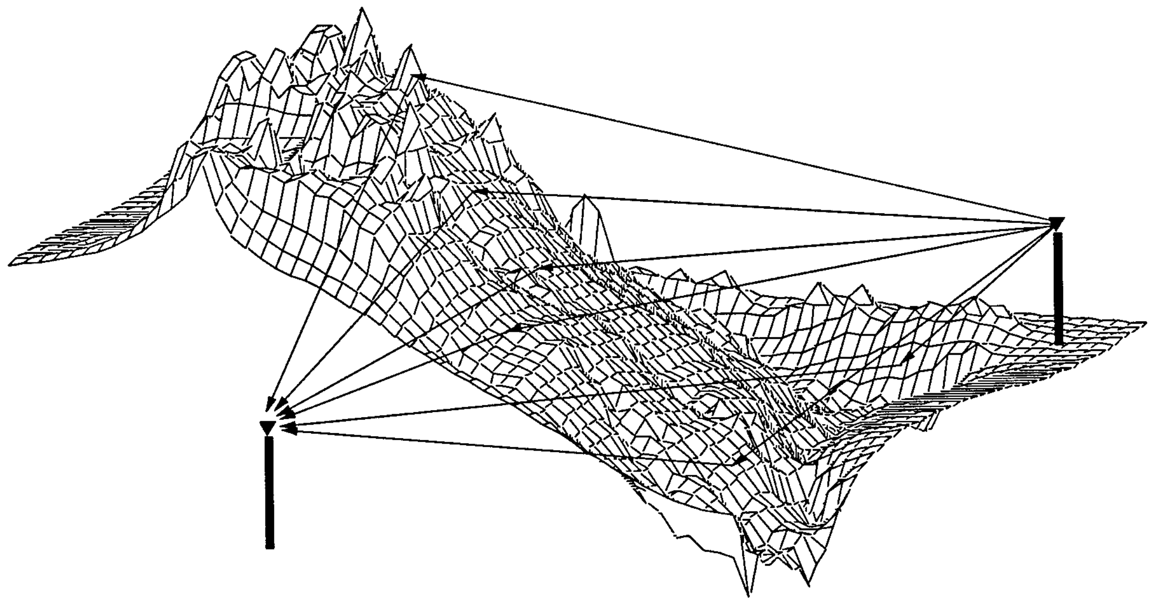

Figure 1: Rough surface scattering from the ground.

minated patches that is in the line of sight of the observation point. Therefore if the terrain patch is in the LOS of both the source and observation points the diffuse energy is calculate with the Rayleigh rough surface formulation.

4 Improve Computational Speed

In order to reduce the computation time of a simulation it is necessary to identify the algorithms within the program where the largest portion of the time was spend. Of the two functionalities that consumes the largest amount of time that causes the bottlenecks, one is algorithmic and the other is theoretic. The algorithmic bottleneck is associated with the technique used to determine whether a ray will intersect with a particular wall. The theoretic bottleneck is due to the need to perform a nearly complete ray trace at each diffracting vertical edge (building corner).

The algorithmic bottleneck can be corrected by redesigning and implementing a more efficient technique to find the intersection of a ray with the walls of the buildings in the simulation area. This technique employs a lookup table of the building walls to test only those walls that lie near the ray path instead of testing all the walls against each ray. The theoretic bottleneck is a much more difficult problem to solve since it is not possible to simply eliminate or approximate the ray trace from a diffracting edge. Instead, it is possible to take advantage of the fact that the diffracting corners and the receivers remain stationary between simulations and thus conducting a single ray trace for these diffracted rays and storing the results for future simulations with different transmitter locations.

4.1 Subdivision of the Horizontal Space

One of the most intensive operation in the ray trace is the one that finds all the building walls intersected by the 2D ray representing the vertical plane. Approximately 70% of the total computation time is spent executing the function used to determine all the intersections between a ray and all possible walls that fall along its path. This operation is extensive because it is necessary to check every wall in the building database with each ray generated by the ray trace. In order to reduce the overall computation time it is essential to limit the number of walls that need to be tested for a particular ray. One method of accomplishing this is to provide some foreknowledge of the subset of walls that lie along the path of a ray launched in an arbitrary direction.

To accomplish this we subdivide the prediction area into smaller areas and identify in a table all the subdivided areas that a given wall lies in. During the ray trace, only those walls lying in the areas the ray passes through are checked for intersections. Figure 2 shows an example of the subdivision of the prediction area in the horizontal plane. The process begins with finding the minimum and maximum (x, y) coordinates of all the walls in the overall prediction area. The area is then subdivided into the largest integer number of grid squares that are closest to the specified grid square size, which is currently set to be 50 meters. Each

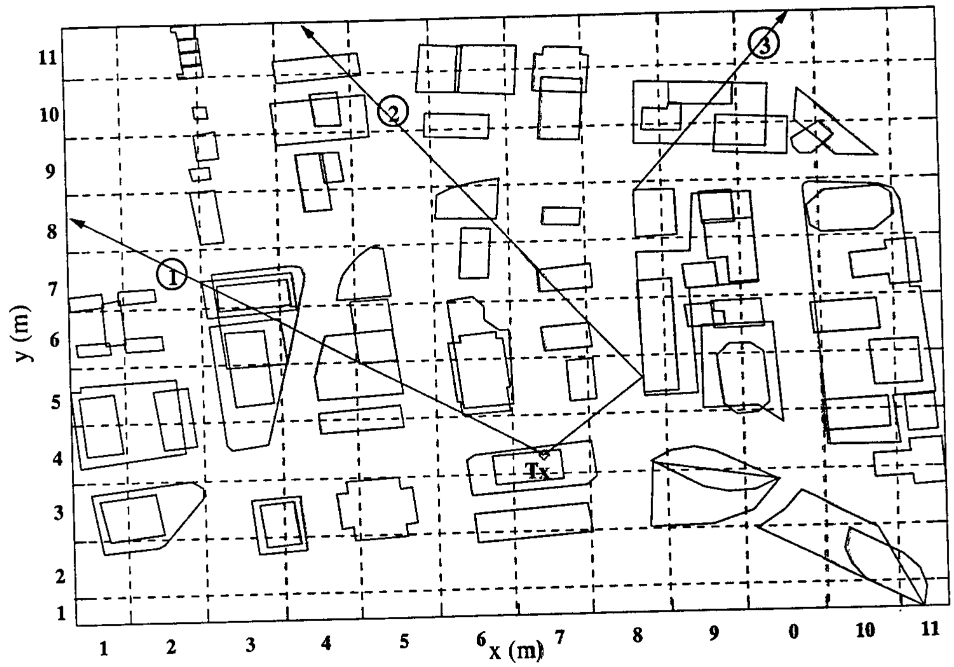

Figure 2: Subdividing the horizontal plane into grid squares.

wall is checked in turn to find the grid square(s) that the end points are located in and the grid square(s), if any, that the wall spans across. A table is then generated, listing for each grid area the walls that lie in or passing through it. The subdivision of the prediction area and the assignment of wall to the gridded horizontal area is done prior to the start of the ray trace as a preprocessing step.

A ray can be launched from the transmitter (1), or building corner (3) or generated at a point of reflection (2) as shown in Figure 2. Each ray, regardless of its source has a unique direction and point of origin. Using the unique parameters of the ray it is possible to determine the grid squares that a ray will travel through from the origin to the boundary of the prediction area. Then using the grid coordinates as reference coordinates in the table of walls it is necessary to test only those walls that are assigned to the grid squares that the ray travels through. With a list of walls that the ray can potentially intersect, the algorithm which compares the angles of the end points and the intersection point with the origin of the ray is used to determine if a actual intersection can occur. This method prevents the large number of frivolous calculations with walls that are located nowhere near the ray path.

A runtime profile of the computation time indicates that there is an improvement of 57% in the overall run time of the program when using the gridding over the previous method of testing against all the walls. In particular, the percent of time spent in the function used to find the intersections between a ray and the walls was reduced from consuming 70% of the total computation time to just 25%. With the gridding in place, finding intersections is no longer the most computationally intensive operation.

4.2 Preprocessing the Ray Trace From the Diffracting Vertical Edges

The major computational effort when utilizing ray tracing for propagation prediction is from the diffraction at the edges of buildings since a source ray will generate a whole family of diffracted rays. It is necessary to treat the edge as a secondary source and trace each of these diffracted rays in order to account for the effect of diffracting around a building. The VPL technique has already reduced the computational burden of diffractions by treating horizontal and vertical edges differently. Diffraction at horizontal edges are approximated as a diffracting source with only two diffracted rays, one in the incident and the other in the specular reflected direction of the horizontal plane. If the angular separation between adjacent rays from a source is $\frac{1}{4}^{\circ}$ then the VPL technique will effectively reduce the number of ray traces of a 90° corner from 1080 to 2.

On the other hand, diffraction from a vertical edge of a building is currently treated as a secondary source. This means that in the ray trace the vertical edge is consider equivalent to a transmitter with rays launched at equal intervals from the entire exterior angle of the corner. Therefore at a 90° corner there is approximately 1080 rays generated (270° exterior angle $\div \frac{1}{4}^{\circ}$ separation between adjacent rays) which will undergo the same types of interactions as the rays from the actual transmitter. As an example, an area that contains 80 building structures it is possible to have over 400 corners. Normally for a simulation run that considers up to one vertical edge diffraction, up to 300 corners will be illuminated by at least one ray from the transmitter. In order to account for diffraction around the buildings it is necessary to perform the equivalent of almost 300 addition ray traces from a transmitter. This is the primary reason for the extended amount of time needed to finished the calculations. Additionally, if double diffraction at the vertical edges is desired, a further 400 ray traces will need to be done since all the corners will be illuminated by a first order diffracting corner. As an example, if the ray trace from a transmitter took one minute to complete, then to consider single diffraction from a vertical edges would add approximately 5 hours to the overall runtime.

It seems quite evident that the greatest savings in computation time would come from reducing the amount of time spent doing the ray traces from the corners or by eliminating them all together. One method that can be employed to eliminate the need to perform the ray traces from the corner for each simulation run is to do the ray traces from the corners and then store the results for later use, or to preprocess. Although this technique does not completely circumvent the need to do the ray tracing from the corner it does significantly reduce the runtime when the preprocessed data exist. Preprocessing is a viable technique because the position of the buildings (and thus the corners) and of the receivers usually remain unchanged during successive simulations. Therefore, rays between a corner and receiver pair will remain unchanged even when the position of the transmitter is changed.

The mechanics of the preprocessing is identical to a ray trace from the actual transmitter except that the rays are traced only for the exterior angles of the corner and is repeated n times for the n number of corners in the database. Information about the rays that illuminates the receivers and other corners are store in a formatted file. If this preprocessed data file exists at the start of a simulation the ray trace is done only for the transmitter.

Subsequent to the ray trace from the transmitter the preprocessed data is read into the data structure from the file and the power received from every combination of a single and double diffracted ray from a vertical edge to a receiver is determined from the preprocessed data. The greatest advantage of using preprocessing is that it reduces all subsequent simulation using the preprocessed data to just a single ray trace from the transmitter which is usually in the order of a few minutes for a small database to less than 1 hour for very large and complex databases. Another unexpected advantage is that only a single set of ray traces needs to be performed at each corner. This means that even if double vertical edge diffraction needs to be considered it is only necessary to trace from a vertical edge once.

5 Graphical User Interface Development

The development of the graphical user interface has provided the user with a better interface with the VPL kernel and computational engine. The GUI has also enhance the ability of the user to visualize and manipulate the input data as well as view the results of the simulation more effectively. The GUI and visualization software is a critical part in the development of a fully functional propagation prediction tool and its future commercial success. During this Phase I project the goal was to developed an user interface that would allow a demonstration of the capabilities of viewing and editing the building, receiver and terrain data, allow the user to enter the runtime parameters and execute a simulation and finally to view the results of the simulation.

The current implementation reads the same input data as the ones used for a simulation and present a graphical visualization of the data on screen. The building, receiver and terrain features can be viewed separately or combined together to show a complete picture of the physical environment. The scene can be viewed in 2D from a number of different positions such as top and side views and in 3D perspective view. The scene can also be controlled by an orientation tool that allows the user to rotate and zoom in on the scene. The scene can also be traversed by clicking and drag the mouse directly on the scene. Individual objects within the scene such as a building or receiver point can also be manipulated. A building can be moved and rotated by a simple click and drag of the mouse cursor on the building. In addition of changing the location of a building it is also possible to change the shape and height of the building by clicking and dragging a vertex of the building. A receiver point can similarly be moved to any location in the scene. An entirely new building or receiver point can be added to the scene by selecting the appropriate check box on the orientation tool and then clicking a position in the scene. The new building will initially appear as a simply square cube, its position and shape can then be adjusted by the same procedures as for an existing building in the scene.

The VPL kernel was initially developed as a separate stand-alone module that prompted the user to enter the parameters used in the simulation. When considering the integration of the VPL kernel into the GUI it was decided to maintain a high degree of separation between the development of the GUI and the VPL kernel. This separation would allow for two distinctly different development teams to work on each part of the project with minimal interference between both groups. In order to integrate the VPL kernel with the GUI it was necessary to redesign the interface between the user and the computational engine. The parameters would now have to be passed by the GUI to the VPL kernel since the GUI is the only means by which the user can interact with the program. A parameter page along with a data file page was implemented into the GUI with all the fields in both pages being encapsulated into a single data structure and passed to the VPL kernel when it is executed as a separate thread within the application.

The results of the simulation includes all the rays that have illuminated a receiver within a predetermine range of received power. All rays for each receiver is currently written to an output file where it is retrieved for additional processing before being displayed. The results of the simulation contains a wealth of raw information that can be processed and includes:

- Ray path length (and thus propagation time).
- Angle of departure from source and arrival at the receiver.
- Power received (for a 1W source).

These characteristics can be view directly to obtain some basic insight to the propagation environment or it can be processed further to extract a sophisticated picture of the multipath characteristics of the channel. Consequently, there is a large variety of different outputs that can be generated from the raw simulation data.

An example of one type of output is a line plot of the power received versus location or receiver number. This generic type of 2D line plot can be used to plot a variety of different results. In addition to merely plotting the data the software is capable of allowing the user to manipulate the characteristics of the plot including:

- Adding addition plot that are overlayed on top of the existing plot.
- Adding and removing the grid that overlays the graph.
- Changing the x and y tick marks spacings
- Changing the minimum and maximum value of the coordinates thereby expanding or compressing the view of the data on the plot.
- Adding labels to the axis and title of the plot.

6 Conclusions

The technical goals of this Phase I SBIR project was to demonstrated the feasibility of enhancing the capability of the VPL kernel by including terrain effects on the propagation path and diffuse scattering from the ground. In addition to enhancing the electromagnetic features of the VPL kernel it was also necessary to demonstrate a significant improvement of the overall computational time. Ultimately the commercial success of a propagation prediction tool is dependent of its ability to provide accurate simulation results in a reasonable length of time.

Another aspect of developing a site specific propagation software application is in the user's ability to interface with the input and output data. Therefore it is necessary to provide the user with the means to view, edit and generate the input data, execute a simulation base on the input data and to produce and view the results in a useful manner. The graphical user interface that was developed during Phase I clearly shows the convenience and ease in which the user can interact with the propagation prediction application.