3D-Visualization of Power System Data Using Triangulation and Subdivision Techniques

Abhishek Venkatesh
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332
venky@gatech.edu

George Cokkinides, A. P. Sakis Meliopoulos
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332
Cokkinides@comcast.net, sakis.m@gatech.edu

Abstract

3-D surface visualizations of various power system operating quantities has always been challenging in terms of correctly capturing the changes of an arbitrary geographical shape power system. Triangulation methods offer promise for meaningful 3-D surface visualizations of such systems. In this paper we propose a scheme for such visualizations based on subdivision of triangle meshes. Input consists of various power system quantities such as voltage magnitude, voltage phase, reactive power flow, real power flow, electric current, etc. The data may be available from simulations or from real time streaming data from a model that is two-dimensional (geographic). We first perform a Delaunay Triangulation on the set of 2D sites and generate a triangle mesh. This triangle mesh is used to represent a coarse 3D surface. The height of this surface at a site is equal to the power system quantity at that site. This surface is refined using the butterfly subdivision scheme with an additional constraint that the heights of the interpolated vertices lie within the bounds of the original vertices from which they were interpolated. After each level of subdivision, we perform a modified Laplacian smoothing to compensate for the discontinuity introduced due to this bounding. The method is suitable for effective visualization of large geographic data. Example visualizations and performance indices are provided in the paper.

Keywords: Delaunay triangulation, corner table, butterfly subdivision, laplacian smoothing, 3D-visualization, interpolating, surface.

1. Introduction

Efficient visualization techniques can help operators analyze large amount of data on the Computer. In this paper we present a visualization scheme based on triangulation and subdivision techniques for presenting large sets of geographically dispersed power system data to the user.

Given a set of 2D sites where readings are available for the power system, the goal is to display a smooth interpolating 3D surface through these data points with high fidelity. We start by triangulating the 2D sites using the popular Delaunay triangulation scheme. Many efficient algorithms [2] [3] [4][5] have been proposed for the Delaunay Triangulation. We choose the randomized insertion based incremental algorithm which is known for its simplicity, efficiency and robustness [3] [5]. The 2D triangle mesh thus generated is used to represent a 3D surface in which the height at a site is equal to the power system reading at that site. The subdivision of this initial control mesh leads to a series of refinements which introduces new vertices and triangle faces into the mesh. The positions of the newly introduced vertices are calculated using affine combination of neighboring vertices. The analysis of the subdivided surface is not straightforward but it provides a powerful and efficient tool for generation of surfaces for large data sets.

There are many popular subdivision schemes available for triangle meshes [6] [7] [11]. Of these, the butterfly subdivision scheme proposed by Dyn, Gregory, and Levin achieves C1 continuity and interpolates the initial control mesh both of which are desirable properties for our visualization purpose. Unfortunately the height of the interpolated vertices introduced using the butterfly subdivision scheme may not lie within the range of its neighbors. This can be particularly misleading in visualization of power system data. As an example, if we want to locate regions where the voltage reading is above a threshold, the original butterfly subdivision scheme may produce points which are above the threshold even if all the original vertices from which it was interpolated lie below the threshold. Thus we use a modified butterfly subdivision scheme which bounds the height of the interpolated vertices within the range of the neighboring control vertices of the original mesh. This bounding might lead to discontinuities in the surface. To compensate this discontinuity we introduce a smoothing step after each level of the butterfly subdivision which is described later in the paper.
We use the corner table representation [1] of triangle meshes as the underlying data structures for our algorithm. Since we are dealing with large data sets, we need to be efficient in terms of both storage and speed. The corner table provides a compact representation of the triangle mesh in terms of storage and provides easy access to neighboring vertices which simplifies the implementation of the various steps of our proposed visualization scheme consisting of triangulation, subdivision, bounding and smoothing.

The paper is organized as follows: Section 2 discusses the Corner table operations. Section 3 presents the incremental Delaunay Triangulation. Section 4 presents a refinement of the Bounded Butterfly subdivision scheme accompanied by modified Laplacian smoothing. Section 5 presents examples of 3D surface displays using these techniques. In section 6 we present example results for electric power systems. Finally section 7 provides summary and future directions for efficient 3D surface visualizations. In addition, Appendix A provides example screenshots of actual power system data (LongBay substation, USVirginIslands).

2. Corner Table

The Corner Table [1] provides a compact data structure to represent triangle meshes: A triangle mesh is defined by specifying the set G(x,y,z location) of its vertices and a set T of its triangles stored as V-table. The V-table stores vertex indices of G in sets of three with each such set representing a triangle in the set T. The association of a vertex with a triangle is denoted as a corner. The three corners of a triangle are stored in counter clockwise order. Given this ordering and a corner we can get the previous or next corner. Every triangle has three neighbors. The corner table captures this information in the ‘O’ (opposite) table. For every corner ‘c’ the corner ‘o’ of the neighboring triangle opposite to corner ‘c’ is stored. We use the following notation for the corner table implementation:

\[ G[ ]: \text{Stores vertex coordinates.} \]
\[ V[ ]:(V-table):\text{Stores triplets of indices forming a triangle. The corresponding vertex coordinates of the triangle can be accessed by using these three indices into G.} \]
\[ O[ ]:(O-table):\text{Stores the opposite corner for every corner corresponding to V.} \]

3. Delaunay Triangulation

Given a set of planar points P, a Delaunay triangulation is a triangulation such that none of the triangle’s circumscribing circle contains any point from P([4]). Some nice properties of Delaunay triangulation are:

- Compared to any other triangulation of the points, the smallest angle in the Delaunay triangulation is at least as large as the smallest angle in any other. Thus it maximizes the minimum angle of all the angles of the triangles in the triangulation. This is desirable because we get nice big fat looking triangles instead of skinny looking triangles. However, the Delaunay triangulation does not necessarily minimize the maximum angle.
- The triangulation is unique if no 4 points lie on the same circumcircle and no three points are on the same line. If 4 points lie on the same circle there can be multiple triangulations depending upon number of such occurrences in the point set P.
- It has at most 3n-6 edges and at most 2n-5 triangles.
- Efficient Algorithms exist to compute the Delaunay triangulation.
- The boundary of the Delaunay triangulation also defines the convex hull of the point set P.

The incremental insertion algorithm of Delaunay triangulation in 2D is very popular due to its simplicity and stability. This algorithm [5][12] repeatedly adds one vertex at

**Figure 1: Corner Table Operations**

Using the V and O tables, given a corner, c, we can access (See figure 1):

- c.t: its triangle.
- c.v:its vertex.
- c.p: previous corners in c.t.
- c.n: next corner in c.t.
- c.o: opposite corner.
- c.l: left neighboring corner (c.p.o)
- c.r: right neighboring corner (c.n.o)
a time and re-triangulates the affected triangles as shown in Figure 2.

The algorithm is as follows:

1) Start with a triangle large enough to contain all the input points. Since this is the only triangle it satisfies the Delaunay property.

2) A new vertex ‘P’ is added to the existing Delaunay triangulation as follows:
   i) Find the triangle ‘t’ which contains the new vertex P. Let the three points of the triangle be A,B,C. This can be done by starting from an arbitrary triangle and moving in the direction of P (see Figure 3)
   ii) Delete the triangle ABC and create three new triangles which have the vertices as ABP, BCP and CAP (see Figure 4a)
   iii) The edges of the old triangle ABC are inspected to verify that they still satisfy the empty circumcircle condition. If the condition is satisfied the edge remains intact. If the condition is violated then the edge is flipped as shown in Figure 4b. Each flipping may result in two more edges becoming candidates for flipping. In worst case all edges have to be flipped. But usually if the vertices are inserted in a random order only a few edges get flipped.

The main steps of the above triangulation scheme are point location, insertion and edge flipping. Figures 3 and 4 show the corners table operations involved in these steps.

The algorithm is as follows:

4. Refining

After performing the triangulation we have a coarse surface represented as a triangle mesh which is used as the control mesh for the refinement step. The refinement step takes this control mesh and subdivides it, introducing new vertices and faces. To generate a smooth surface the position of these new vertices are altered based on the nearby vertices.

We are dealing with visualization of large data sets and need to process the surface in real time. The butterfly subdivision
has some desirable properties for this as it is interpolating, local, simple to implement and generates a reasonably smooth surface with C1 continuity for regular settings (vertices with valence 6)[7]. Unfortunately the surface generated may have overshoots (figure 6) or undershoots which can be misleading for the visualization of the power system data. Thus we add an additional constraint to the butterfly subdivision that the height of the newly introduced vertices is contained within the bounds of the original vertices from which they were interpolated. This leads to plateaus like patches in the surface along with loss of continuity around the border of these plateaus. To compensate for this discontinuity around these plateaus and around vertices not having valence 6 we perform a modified version of the laplacian smoothing after each level of subdivision. Thus one level of refinement includes one level of butterfly subdivision followed by the modified laplacian smoothing operation. In the following subsections we describe the original butterfly subdivision scheme with our bounding constraint and the modified laplacian smoothing scheme.

4.1 Bounded Butterfly Subdivision

The butterfly subdivision (see Figure 5) scheme [6] recursively transforms each triangle face of the triangle mesh into a patch consisting of four triangular faces interpolating the original triangle mesh points. Each edge of the original triangle mesh is split into two by introducing a new vertex on the edge. The position of this newly introduced vertex is adjusted using the butterfly rule (see Figure 5) and its height is bounded within the range of the height of the vertices participating in the butterfly rule (see Figure 6).

4.2 Smoothing

The Laplace mesh smoothing algorithm is simple but effective tool: The position of a vertex is replaced by average of the position of its neighbors. Unfortunately the simple laplacian smoothing shrinks the original mesh which is undesirable. Reference [8] proposes a method that reduces this shrinkage using the Laplace+HC Mesh Smoothing Algorithm. The idea here is to move the vertices somewhat back towards their former positions. The direction and amount for this backward movement is calculated using the original and previous vertex position as well as from the mean displacement vector in the neighborhood. Reference [9] generalizes this approach by means of the tuck(s) operation. tuck(s) is an operation in which a vertex is moved by a fraction ‘s’ towards the average of its neighbors. A sequence of tuck(s) operations for a value of s in [0,2/3] eliminates sharp features which is the same as the laplacian smoothing. The sequence {tuck(s);tuck(-t);} is applied to reduce the shrinking effect which is same as the idea proposed in reference [8]. The tuck(s) operation reduces the details and shrinks the mesh and tuck(-t) exaggerates the details of the mesh. Since the tuck(-t) operation is done on the smoother mesh it does not reproduce the original sharp features of the mesh. By varying the parameters ‘s’ and ‘t’ we can have better control over the smoothing operation for our purpose.

Our goal is to generate a smooth interpolating surface. So the tuck operations are limited to the vertices introduced by the subdivision step so that the surface is interpolating. The value of ‘t’ is chosen greater than ‘s’ for the tuck smoothing operation to reduce the effect of shrinking due to the smoothing operation. In our implementation we choose s=0.6 and t=0.8. Since the original vertices are not modified the smoothing operation would actually decrease the smoothness around the original vertices as shown in Figure 7. To maintain smoothness around the original vertices we use a modified tuck operation as follows: The displacement vector for a vertex in tuck operation is computed as before, but in addition to that the displacement vector of the original vertex of the control mesh is added to the displacement vector of its nearby vertices. This smoothing scheme reduces the discontinuities around the plateau regions and vertices not having valence 6 (see Figure 7)
5. Displaying the Surface

The smooth 3D surface rendered trivially (see Figure 9c) might not be easy to comprehend though it is pleasing to the eye. We also show the refined surface as an Iso-Surface that represents points of constant value within a volume of space. The value maybe voltage magnitude, voltage phase, reactive power flow, real power flow, electric current, etc. Generating the Iso-Surface on the CPU is computationally costly because it involves computing the intersection of the IsoCurves with each of the triangle in the 3D mesh. Since the number of triangles we have to process after refinement is huge, this approach will not be efficient for displaying the surface in real time. We take advantage of the texture mapping and interpolation provided by OpenGL to do the job for us. Given the number of desired IsoCurves, we generate a corresponding OpenGL Texture which is mapped to the 3D triangle mesh surface to render the iso-surface. The texture map will have as many entries as there are isoSurfaces. The color at texture coordinates (0,0) is the lowest IsoSurface value and the color at texture coordinates (1,0) is the highest IsoSurface value. For example in Figure 9, nine IsoSurface levels with 9 different colors in the texture map is used to texture map the 3D surface. The texture coordinate for a vertex is computed using its height (=VertexHeight/MaxVertexHeight). When the triangles are rendered OpenGL interpolates these texture colors automatically. Example of the rendered IsoSurface is shown in Figures 9d and 9e.

6. Results

We built an application for the above visualization scheme in Visual C++ and OpenGL. We present here some sample screenshots and timing data we obtained using our application.

Figure 8 shows the result of applying our visualization scheme on a frustum. It is important to note that the original butterfly subdivision scheme generates an interpolating surface but overshoots the original height of the frustum. Applying the bounded butterfly subdivision produces an interpolating surface that does not overshoot but does not produce a smooth surface near the vertices of the original control mesh. Applying the bounded butterfly subdivision along with the proposed modified tuck based laplacian smoothing scheme produces an interpolating surface which reduces the discontinuities present in the bounded butterfly subdivision.

In Figure 9 the control mesh 9a) is refined using the bounded butterfly subdivision along with the modified smoothing scheme to generate 9c). The corresponding Iso-surface with nine levels is shown in 9d) and 9e).

Table 1 presents the execution time in seconds for varying number of sites with three levels of refinement steps. The timing was measured on an Intel(R) Core(TM)2 Duo CPU T5470 @ 1.60GHz (2 CPUs), with 1024MB RAM and GeForce 8400M GS graphics card running Windows Vista operating system. Finally, Appendix A provides additional examples of visualizations.

<table>
<thead>
<tr>
<th>Number of points(sites)</th>
<th>Total Delaunay Triangles Generated</th>
<th>Delaunay Triangulation Time (sec)</th>
<th>Time for 3 levels of refinement (sec) (subdivision and smoothing)</th>
<th>Triangle Count after 3 levels of subdivision</th>
<th>Render texture mapped Iso-Surface</th>
</tr>
</thead>
<tbody>
<tr>
<td>512</td>
<td>1006</td>
<td>0.005</td>
<td>0.328</td>
<td>64384</td>
<td>15</td>
</tr>
<tr>
<td>1024</td>
<td>2026</td>
<td>0.016</td>
<td>0.702</td>
<td>129664</td>
<td>47</td>
</tr>
<tr>
<td>2048</td>
<td>4069</td>
<td>0.046</td>
<td>1.482</td>
<td>260416</td>
<td>109</td>
</tr>
<tr>
<td>4096</td>
<td>8163</td>
<td>0.109</td>
<td>3.088</td>
<td>522432</td>
<td>250</td>
</tr>
<tr>
<td>8192</td>
<td>16354</td>
<td>0.235</td>
<td>6.552</td>
<td>1046656</td>
<td>562</td>
</tr>
</tbody>
</table>

Table 1: Execution time in seconds for varying number of sites for 3 levels of refinement.
7. **Conclusion and Future Work**

In this paper we presented a visualization scheme for various power system quantities such as voltage magnitude, voltage phase, reactive power flow, real power flow, electric current, etc. The sites where readings are available are triangulated using the incremental Delaunay Triangulation algorithm to generate a control mesh. The standard butterfly subdivision scheme generates an interpolating surface but overshoots the original control mesh which might be misleading for visualization of power system data. Using the bounded butterfly subdivision scheme along with the modified tuck based laplacian smoothing we are able to generate a smooth surface which lies within the bounds of the original control mesh. The proposed scheme is simple and can be used to identify specific regions with interesting readings.

We are looking at ways to improve the efficiency by utilizing the parallel computing power of modern day graphics processor. Since the presented scheme operates on local area it is easily parallelizable. An animation of the 3D surface can be generated for continuous readings received from the sites to view quantities and observe them over a period of time. The animation may also be used to diagnose the readings and study the behavior around readings of special interest.

8. **Acknowledgments**

The work reported in this paper has been partially supported by the PSERC project S-25, “Effective Control Center Visualizations”. This support is gratefully acknowledged.
References


[2] Steven Fortune. Voronoi Diagrams and Delaunay Triangulations. Computing in Euclidean Geometry (Ding-Zhu Du and Frank Hwang, editors), Lecture Notes Series on Computing, volume 1,


Appendix A

This Appendix provides example screenshots of the above visualization scheme on actual voltage readings obtained from the Long bay substation of the US Virgin islands power system. The purpose of the screenshots is to illustrate the intermediate steps for obtaining the final surface 3-D visualizations with excellent visualization of sharp transitions of voltage magnitude. In all visualizations, the geographic location of the power network is shown as a set of 2D curves which is visible from any perspective view. The sites where the reading is available are highlighted by small disks on these curve lines (see figure c). The user can click on these disks to visualize the part of the power system network he is viewing.

a) Power line ground network represented as a set of 2D curves

b) 2D-Delaunay Triangulation of sites on the network where reading is available.

c) Side view of Texture Mapped Iso-Surface mounted on top of the ground network. The sites where reading is available are shown as blue disks on this curve,
d) Texture Mapped Iso-Surface of triangulated Mesh.

e) Smooth Refined Surface

f) Texture Mapped Refined Iso-Surface

g) Wireframe view of Texture Mapped Refined Iso-Surface.