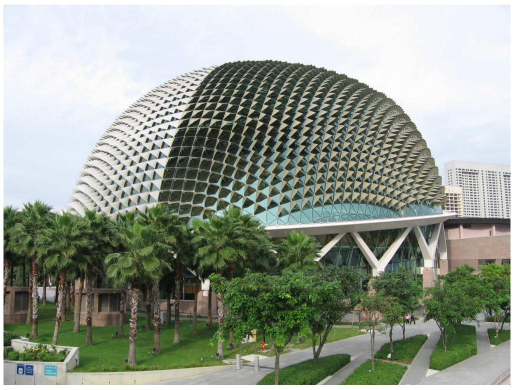
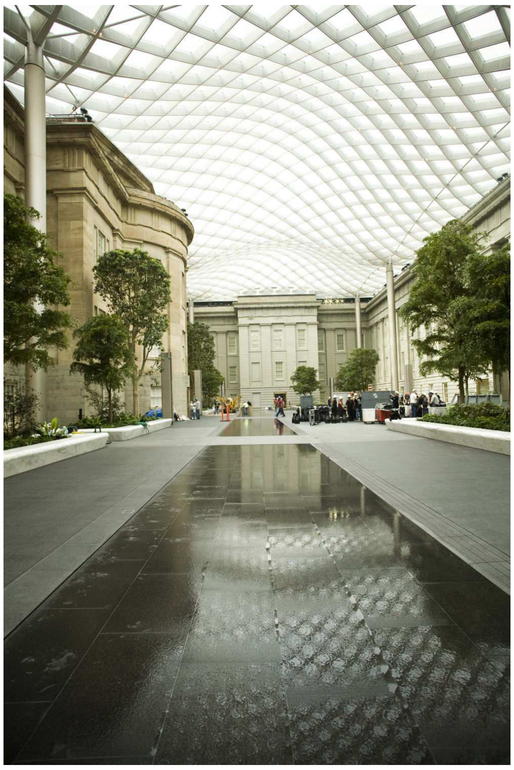
Teaching Mathematics and Statistics in Sciences - IPA HU-SRB/0901/221/088
Interesting Mathematical Problems in Sciences and Everyday Life - 2011

Building a conference center or triangulating a surface

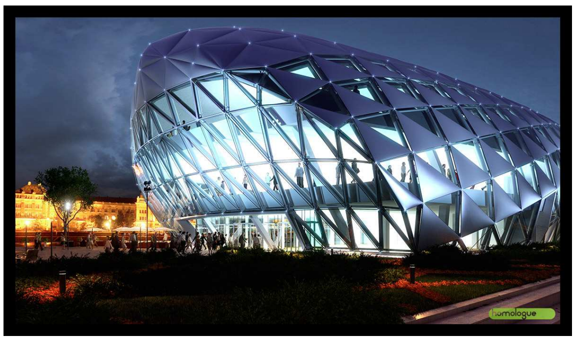
Géza Makay
Bolyai Institute, University of Szeged, Hungary
Aradi vértanúk tere 1, Szeged, Hungary, 6720
makayg@math.u-szeged.hu
http://www.math.u-szeged.hu/˜makay/

The Alukonstruct Ltd, located in Szeged, Hungary, is building surfaces from steel support beams and glass triangles. To construct such surfaces is a challenge to the engineers, they must take into account among other things the surface's dead load, the weight of the snow, the wind pressure, and even smaller earthquakes. There is a large literature on triangulating regular surfaces like sphere, cone or cylinder, and there are many computer programs to construct such surfaces. Lately the artists and engineers experiment with uncommon surfaces. It is a very hard problem even to triangulate such surfaces, not to mention the engineering work after that. In the paper we present a method to construct appropriate triangulation for such surfaces.





Some examples of buildings abroad



The CET in Budapest, Hungary

1. Introduction

The Alukonstruct Ltd [1] presented a mathematical and computer programming problem to the Bolyai Institute (Department of Mathematics) [2] at the University of Szeged, Hungary. The problem is to triangulate non-regular surfaces (see video: alagut.avi) so that by postprocessing this triangulation further in engineering softwares written for this purpose the surface can be constructed using steel support beams and glass triangles. The proper triangulations should have the following properties:

- Inside the surface all vertices must be of degree 5 or 6, i.e. only 5 or 6 triangles can connect to a vertex, or in other words, only 5 or 6 support beams can meet in a vertex. We exclude other degrees, because smaller degrees make the surface weak, while using larger degrees means smaller angles in the glass triangles, and that makes the glass fragile.
- There is a lower and upper limit on the length of the support beams. Shorter support beams make the construction lengthy and troublesome, longer support beams mean larger glass triangles, and it makes the handling of those triangles hard and the glass fragile.
- There is a lower and upper limit on the angles of the triangles. Smaller angles make the glass fragile, larger angles make the surface weak.
- There are fixed curves on the surface (for example: the edge of the surface, or the line of a support buttress), the triangulation must keep these curves because they are fundamental parts of the construction.
- There are fixed points on the surface (the corners on the entrances or the top point of a pier), the triangulation must keep these points because they are fundamental parts of the construction.

Naturally, confirming to these restrictions does not mean that the surface can be built in the practice, but surfaces not fulfilling these properties can hardly be constructed.

Obviously, the program to be written must be user-friendly, practicable and of course productive. To this end the program must be able to do the followings:

- Import a commonly used format (DXF=Drawing Interchange Format [3]) from engineering softwares and export the results in the same format. To help the work in the engineering software further, the program must compute normal vectors for the surface in the vertices and the middle points of the edges.
- Show the imported and result triangulations in perspective view as a surface, as wireframe, hiding or unhiding the invisible faces, edges and vertices. The surface can be illuminated using diffuse light or spot lamps, the colors can be set by the user. The program should be able to show the problem zones of the surface in false colors. The surface can be moved, turned, magnified, and the perspective can be changed, so that the user can look at the surface from any angle and viewpoint.

- After setting the parameters for the triangulations and starting the generation, the program must create a statistics about the generated triangulations, so that an experienced engineer will be able to choose the appropriate triangulations for further processing and eventual building.
- The triangulation must be editable by the user: he can add or delete vertices, edges and triangles while the program keeps the structure intact. The user can also make the program move the vertices on the surface automatically so that the limits on the length of the support beams and the angles fit better to the specified parameters.

2. The fundamental algorithm

There are only a few methods in the literature about triangulating non-regular surfaces, and even those focus on only one or two of the above conditions (see, for example the Delaunay triangulation [4]). We did not find a paper concerning the degree of the vertices inside the surface. So, a new algorithm must be created that does take all conditions into account. Let us consider the problem from this point of view and work out the basic methods for solving this problem.

The surface is in the 3 dimensional space, so we will use 3 dimensional points for the vertices, two such points determine an edge, and 3 points connected pairwise determine a triangle.

We receive the surface in triangulated form, it would be very hard to handle it in any other format. This means that the original surface (constructed for example from a function's graph or transformations) sustained a modification (the triangulation) and it became worse, not original. Our first question is: what was the original surface, what surface should we triangulate? Naturally, we work with triangulation, so we must refine the given triangulation in some way to approximate the original surface better. We can do this using the following trick: we can break up all triangles into 4 by connecting the midpoints of its edges. Then we move the newly introduced vertices so that the plane angle of all triangles and their adjacent triangles will be about the same: this ensures that the surface will appear to be smoother. During this moving we must keep the fixed curves on the surface: we move the points on the fixed curves in the plane defined locally by the curve, while we move all other vertices orthogonal to the surface. This method is called the refinement of the surface. By executing this refinement several times we can get a fine enough triangulation of the surface, which will approximate the original surface well. Another advantage of using this refinement is that the would-be triangulation will use vertices from this refined triangulation. This ensures that the vertices of the new triangulation are on the surface, and moving them around does not distort the surface provided by the new triangulation (see videos: ikozaeder.avi, tokfinom.avi).

Now the problem can be formulated as the following: which vertices of the refined triangulation should be in the new triangulation, and which ones should be connected to get an appropriate triangulation? We tried several methods, most of the were based on distributing the vertices evenly on the surface (by maximizing the minimum distance between the points, for example), then fitting the faces on these points so that they satisfy the conditions (for example, by taking the convex hull of the points). These methods did not work: they did not agree with the condition on the degree of the vertices. The finally working method was to create the triangulation by adding triangles one at a time and keeping the degrees to be 5 or 6 only. During this construction we exercise extra caution not to step over fixed edges and points. Also, the program automatically moves the new vertices around so that the limits on the length of the beams and the angles match better to the conditions.

3. The mathematical basics and theorems used

It was a surprise, that no higher mathematics is needed to solve the problem: all theorems are covered in high schools or understandable by high school students.

Since the condition on the degrees of vertices seems to be the most restricting one among the conditions, let us look at it a little closer. We will user Euler's Polyhedron Theorem [5]. It states that a polyhedron with v vertices, e edges and f faces must satisfy the equality $v + f = e + 2$. We can deduce from this a (for me very surprising) new

Theorem 1 *If a polyhedron contains only triangles and all of its vertices are of degree 5 or 6, then it has exactly 12 vertices of degree 5.*

Proof. Let us denote by v_5 and v_6 the number of vertices of degree 5 and 6, respectively. Then we have $v = v_5 + v_6$. Let us compute the number of edges by two different methods. If we sum up the degrees of all vertices, we count the number of edges twice, since all edges have two ends, so we have $5v_5 + 6v_6 = 2e$. Also, all faces have 3 edges, so 3 times the number of faces is also the number of edges twice, since an edge is computed twice on the faces joined by the edge: $3f = 2e$. We put Euler's Polyhedron Theorem beside these equalities and we have a liner system of equations:

$$5v_5 + 6v_6 = 2e \tag{1}$$
$$3f = 2e \tag{2}$$
$$v_5 + v_6 + f = e + 2 \tag{3}$$

Computing $6 \cdot (3) - (1) - 2 \cdot (2)$ we get the desired result: $v_5 = 12$. ■

Although it is not relevant in solving our problem, how many vertices of degree 6 are possible, but it is an interesting question for a mathematician. It turns out that this number can be 0 or at least 2, only 1 vertex of degree 6 is not possible.

The icosahedron [6] is a good example for this theorem. It has 20 regular triangles and 12 vertices of degree 5. If we apply the refinement explained above to this polyhedron, all new vertices will be of degree 6, and by doing several refinement we will get an approximate of the sphere. This is the easiest way to triangulate a sphere.

This theorem shows why the condition on the degrees is the most limiting. Naturally, we do not work with complete polyhedrons, our surfaces have holes in them, but it modifies the statement only slightly: we can have at most 12 vertices of degree 5 inside the surface. Also, a little drawing and experimenting shows that the surface to be triangulated must be part of the surface of a (more or less) convex polyhedron.

In the 3 dimensional Cartesian coordinate system we use the following basic notations and relations:

- The length of a vector (x, y, z) is $\|(x, y, z)\| = \sqrt{x^2 + y^2 + z^2}$. The distance of two points is

  $$d((x_1, y_1, z_1), (x_2, y_2, z_2)) = \|(x_1 - x_2, y_1 - y_2, z_1 - z_2)\| = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

- The scalar product of two vectors [7] is

  $$(x_1, y_1, z_1) \cdot (x_2, y_2, z_2) = x_1x_2 + y_1y_2 + z_1z_2 = \|(x_1, y_1, z_1)\| \cdot \|(x_2, y_2, z_2)\| \cdot \cos\alpha,$$

  where α is the angle of the two vectors;

- The vectorial product of two vectors [8] is

  $$(x_1, y_1, z_1) \times (x_2, y_2, z_2) = (y_1z_2 - y_2z_1, z_1x_2 - z_2x_1, x_1y_2 - x_2y_1),$$

  which is a vector orthogonal to the vectors (x_1, y_1, z_1) and (x_2, y_2, z_2), its direction is given by the right-hand-rule, and its length is

  $$\|(x_1, y_1, z_1)\| \cdot \|(x_2, y_2, z_2)\| \cdot \sin\alpha,$$

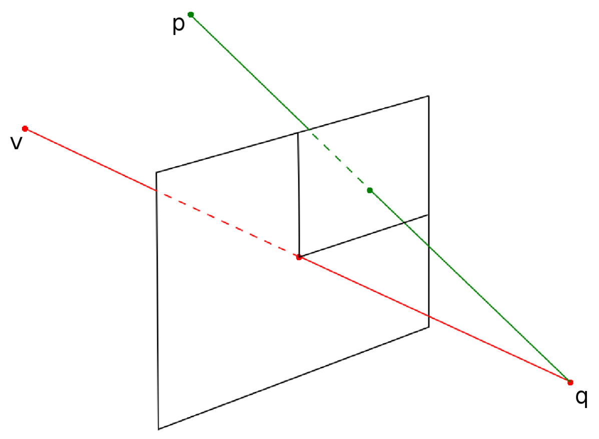
  where α is the angle of the two vectors;

Using the scalar product and the length of the vectors, we can compute length of the edges and the angles of the triangles. We use these to distribute the vertices on the refined surface so that the edges' lengths and angles match better to the conditions. To this end we compute a weighted sum (the goal function) of the deviation of the length from the expected length of the edges meeting in a vertex and the deviation of the angles from 60° of the triangles having that vertex as a point. We try to decrease this sum by moving that vertex to one of its neighbors on the refined surface and we apply this method to all vertices of the triangulation. We repeat this procedure until we cannot move any vertex without increasing the goal function, so we found a minimal value for the goal function for the actual triangulation. Since we tried to decrease the deviances from the best values in the weighted sum, this make our triangulation ”better”, it fits better to the conditions.

If we compute the vectorial product of two edges of a triangle, and divide that vector with its length (we normalize it), we will get a normal vector for that face. If we compute the sum of the normal vectors of two adjoining triangles and normalize that vector, we arrive at the normal vector of the connecting edge. To get a normal vector for the surface in a vertex, we normalize the sum of the normal vectors of the triangles having that vertex as a point. We determine these normal vectors so that they point outwards from the surface: we fix a direction to one triangle and by setting the adjoining triangle’s normal vectors appropriately, the normal vectors will all point to the same direction (inwards or outwards). Then either the user changes the directions of all normal vectors by ”hand” or the program (assuming that the surface is more of a roof than a floor) sets it automatically.

How can one compute the effect of a spot lamp to the surface? If a spot lamp’s light is orthogonal to the surface, then its effect is maximal. The larger the angle of the surface’s normal vector and the vector pointing from the center of gravity of the triangle to the spot lamp, the smaller lighting the triangle will receive from the lamp; if this angle is larger than $90°$, then the spot lamp does not illuminate the outside part of the triangle. This angle can be computed easily using the scalar product. Using this angle the program computes a weighted mean of two colors: one is the (usually) darker diffuse color, and the other is the fully illuminated face’s color. One can even simulate colored spotlights: the diffuse color is the color of the surface, the fully illuminated color is the effect of the spotlight’s color on the surface. We can use the computed color to display the perspective projection of the triangle.

And now to one of the most interesting parts of the program: how can one compute the perspective projection of a triangle, and how can we handle the hiding of invisible faces and edges? The second part of the question is easier, we answer that first. We look at the surface from a point in the 3 dimensional space. We can compute the distance of the center of gravity of the triangle from this point, and order the faces by descending order of this distance. We draw the triangle with the largest distance first and proceed to the triangles with smaller distances: in this way the closer triangles naturally cover the further triangles and edges. We draw the edges and vertices together with their appropriate triangles, so their coverage will be fine, too.



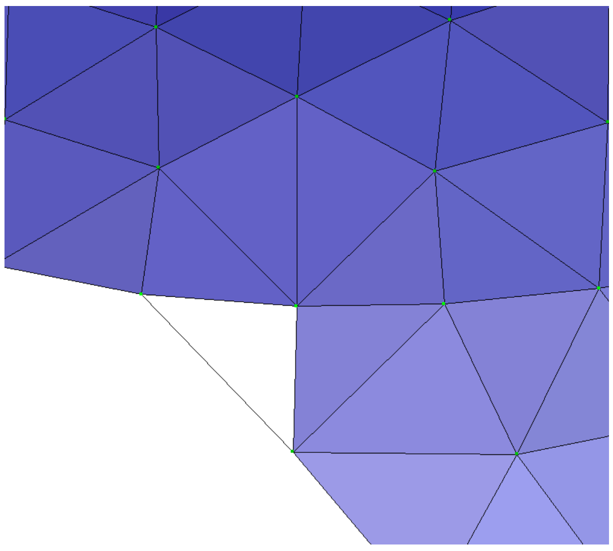
To compute the perspective projection of a triangle, one must compute perspective projections of vertices as they make up the triangle [9]. Let us denote the point where we look at the surface from by q, and the point where we look at by v. This gives the viewing direction $\vec{v}$ which is the normalized vector of $v - q$. We must ensure that the viewing direction is not vertical, then we can determine the right direction ($\vec{r}$) from the fact that a person usually looks at things with keeping his head upright: $\vec{r}$ satisfies the conditions that it is perpendicular to $\vec{v}$, its third component is 0, and of course it is a unit vector. These conditions give a simple linear system for the first and second component of the $\vec{r}$, which can be solved easily. Then the upward direction ($\vec{u}$) is the vectorial product of $\vec{v}$ and $\vec{r}$ with the extra condition that its third component is positive ($\vec{u}$ points upwards). One can imagine the perspective projection as a projection to a plane perpendicular to $\vec{v}$, this plane is of distance 1 from the viewing point q, its vertical axis is $\vec{u}$ and the horizontal axis is $\vec{r}$. The projected point is the intersection of this plane with the line connecting q and the point to be projected and we can express the projected point as a linear combination of $\vec{r}$ and $\vec{u}$. So, we first check that the point is in front of us: $(p - q) \cdot \vec{v} > 0$, then its perspective projection is

$$((p - q) \cdot \vec{r}, (p - q) \cdot \vec{u})/((p - q) \cdot \vec{v}).$$

This formula projects the viewed point v to the two dimensional point $(0, 0)$, hence we need to translate and magnify the resulting 2 dimensional points so that $(0, 0)$ goes to the center of the screen and the obtained image is large enough to view. One can change the perspective by moving the viewing point q closer to the viewed point v. To rotate the image we change the viewing point q, to move the image we change the viewed point v, but we keep their distance constant as it would change the perspective too. Now we can show the user any triangulation from any point in any perspective.

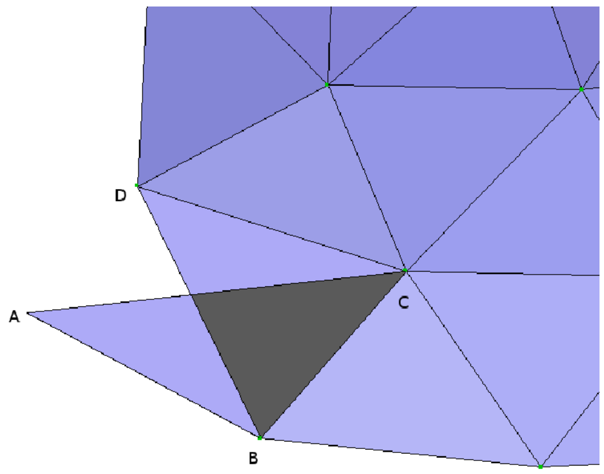
Now we arrived at the heart of the problem: how can we triangulate a surface given by a fine enough triangulation so that it confirms to the given conditions? We now know that we add one triangle at a time so that we pay attention to the condition on the degrees and we try to keep the edges' lengths and the angles in the given intervals by moving the vertices around. So let us make a triangle on the surface and we covered part of the surface by a triangle. This triangle has 3 external edges, edges where the surface still continues. We construct a new triangle so that the new triangle has one of this external edges as an edge of its own, and we continue this procedure as long as we can find external edges in our triangulation. If there are no external edges, then we are finished with the triangulation of the surface. But how can we add a new triangle to the existing triangulation?

We must take several things into account. A vertex inside the surface can only be of degree 5 or 6. Hence, if a vertex already has 6 edges connected, then we cannot add more edges to the vertex, so we must connect the other ends of the existing edges in the correct order (!). In this way the vertex becomes a point inside the triangulation, and we do not need to work with it any more (except for moving it around).



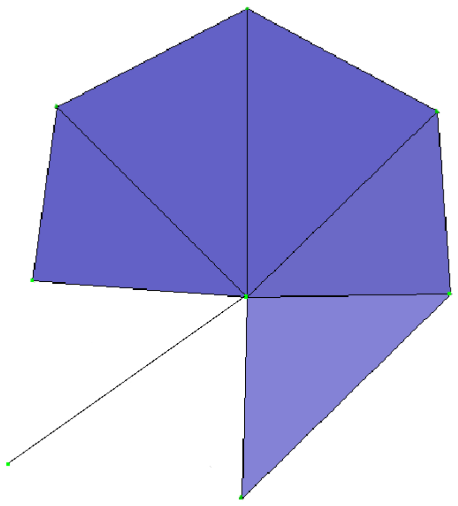
The drawn edge must be added to make the vertex have degree 6

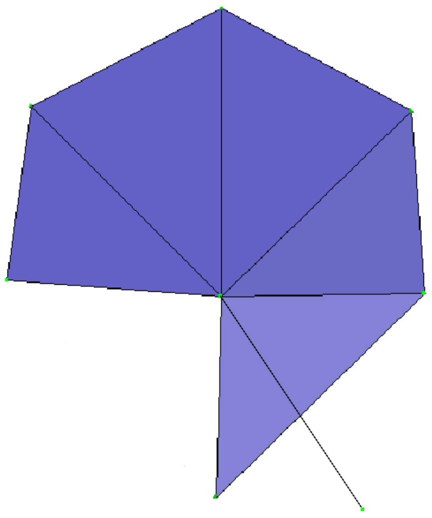
In all other cases we find an external edge of the triangulation, and construct a triangle on it. But we must make sure, that the overlapping shown on the figure below does not occur.



The triangles ABC and DBC are overlapping

This means, that the line between the ends of the external edge and a new vertex (which together with the external edge creates the triangle) must not cross the area covered by the actual triangulation. Since the surface is bending, it is not clear, what we mean by crossing the triangulation, but we clarify it now. Let us construct a plane tangent to the surface at one of the vertices of the external edge (i.e. the plane is perpendicular to the normal vector of the surface at that vertex), and project the neighboring triangles perpendicularly to this plane. If the line segment connecting the original vertex and the new vertex intersect any of the projected triangles, then that new vertex is not appropriate for adding it to the triangulation (see the figure below).


Correctly placed new vertex


Incorrectly placed new vertex

To compute the perpendicular projection we need to use the scalar product again. This ends the brief description of how the triangulation can be achieved (see examples of triangulation: lepke.avi, tok.avi).

Naturally, the actual algorithm contains much more than we could show here. We did not explain how the fixed edges and points are kept in the new triangulation (the edges and points are attracted by the fixed edges and points), how the user can correct of the triangulation by hand (this is just handling the data structures in the program), how we move the newly added vertices in the refinement process, etc. But these are not really mathematical problems, but rather a programming issue.

References

[1] Alukonstrukt Ltd: http://www.alukonstrukt.hu/

[2] Bolyai Institute: http://www.math.u-szeged.hu/

[3] DXF: http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=12272454&linkID=10809853

[4] Delaunay triangulation: http://en.wikipedia.org/wiki/Delaunay_triangulation

[5] Euler's Polyhedron Theorem: http://en.wikipedia.org/wiki/Euler_characteristic

[6] Icosahedron: http://en.wikipedia.org/wiki/Icosahedron

[7] Scalar product: http://en.wikipedia.org/wiki/Dot_product

[8] Vectorial product: http://en.wikipedia.org/wiki/Cross_product

[9] Perspective projection: http://en.wikipedia.org/wiki/Perspective_projection#Perspective_projection

[10] Videos, VRML files: http://www.math.u-szeged.hu/˜makay/avis/