Journal for Geometry and Graphics
Volume 14 (2010), No. 2, 135–145.

Force Directed Near-Orthogonal Grid Generation on Surfaces

Franz Gruber, Guenter Wallner, Georg Glaeser

Department of Geometry, University of Applied Arts Vienna
Oskar Kokoschka Platz 2, A-1010 Wien, Austria
email: franz.gruber@uni-ak.ac.at

Abstract. Single curved surfaces can always easily be covered by meshes that result into an equilateral and orthogonal grid when the surface is developed. On double curved surfaces, however, we can never find a mesh consisting of 'squares on the surface'. Nevertheless, there is a need for 'orthogonal and locally almost equilateral meshes' on such surfaces in several fields, e.g., in architecture (fair and easy to build, increased rigidity) and computer graphics (undistorted mapping of textures, good tessellation for rendering purposes and also for aesthetical reasons). We present an iterative force-directed algorithm that is capable of optimizing given grids with rectangular topology and yields the task in an optimal way. It allows to cover arbitrary parametric double-curved surfaces with grids that are almost orthogonal and, optionally, locally have almost constant grid size in both directions.

Key Words: orthogonal grids, grid generation, force directed algorithm

MSC 2010: 53-04, 53A05, 68U05

1. Introduction

Among the immense variety of curved surfaces, the single curved ones can be developed into the plane. In the plane, we can attach an equidistant orthogonal mesh to the surface and reverse the developing transformation.

When the surface is double-curved this process of mesh generation will no longer work. In fact, it is theoretically impossible to ever find two orthogonal sets of parameter lines with global constant arc lengths from grid point to grid point on both curve sets. Therefore the motivation of this work is to find grids that fulfill the following two geometric conditions simultaneously:

(1) All lines intersect orthogonally and

(2) locally have almost constant grid size in both directions.

It is well known that condition (1) by itself is fulfilled with the principal curvature lines on double curved surfaces (see [12]). Such lines can be found by solving differential equations that work with the first and second derivations of the parameterized surface equation.

We should emphasize that in this paper we are concerned with grids on surfaces that are topologically equivalent to a rectangular grid, in the following simply called quadrangular grids. These grids need not necessarily be topologically equivalent to the principal lines of curvature.

We present an iterative force-directed algorithm that optimizes quadrangular grids with regard to condition (1) and additionally (2). It does not depend on higher analytical theory and is therefore simple to implement and leads to useful results with small expense. Input of the algorithm is an arbitrary, in general non-orthogonal, quadrangular grid on a parametric surface, e.g., by means of different parameterizations. In comparison to analytical methods this gives creative freedom for the designing process of the mesh. On the other hand it avoids the usual problems (e.g., numerical instability) in solving partial differential equations at the cost of numerical imprecision. However, we intend our algorithm as a tool for artists and architects, for which numerical accuracy is usually secondary over aesthetic expression and applicability.

In the remainder of this paper vectors are written in bold face and the superscript * is used to denote unit vectors. The paper proceeds as follows: Section 2 reviews related work in the area of grid generation and force directed algorithms. In Section 3 we outline our proposed algorithm, whereas in Section 4 results and possible applications are presented. The paper is concluded in Section 5.

2. Related work

Since grid generation is a necessity for the computational simulation of physical field phenomena and processes, an extensive body of literature is available. Computational Fluid Dynamics (CFD), which involves the calculation of nonlinear partial differential equations (PDE), which in generally are not solvable analytically, is one of the major domains driving the research in this area (cf. [15]). Orthogonal grids — in particular — are preferable, since the numerical accuracy is highest in such grids. Additionally, as stated by V. Akcelik et al. [1], an aspect ratio close to one is important for isotropic problems to reduce errors in derivatives of the approximate solution. Conformal mappings are frequently used to obtain orthogonal grids in two dimensions (see, e.g., [2]). Conformal mappings, however, are not suited well for parameterizations of arbitrary surfaces ([2]) and are usually restricted to have equal scale factors (see, e.g., [1] which also includes a comprehensive list of references on this topic).

In a related research area Y. Liu et al. [9] introduced the concept of conical meshes[1] for architectural freeform design.

Force-based or force-directed algorithms are usually associated with the drawing of graphs in an aesthetically pleasing way. Their purpose is to place the nodes in a way such that edges are more or less of equal length and/or to minimize the number of edge crossings. They were first introduced to the graph drawing community by P. Eades [4] in 1984 who in turn based his work on a VLSI technique originally described by N. Quinn and M. Breuer [14] for layouting of circuit paths. Since then the algorithms have been optimized (e.g., [3, 6]) and

[1]Conical meshes are quadrilateral meshes with planar faces, which possess a natural offsetting operation and provide a support structure orthogonal to the mesh.

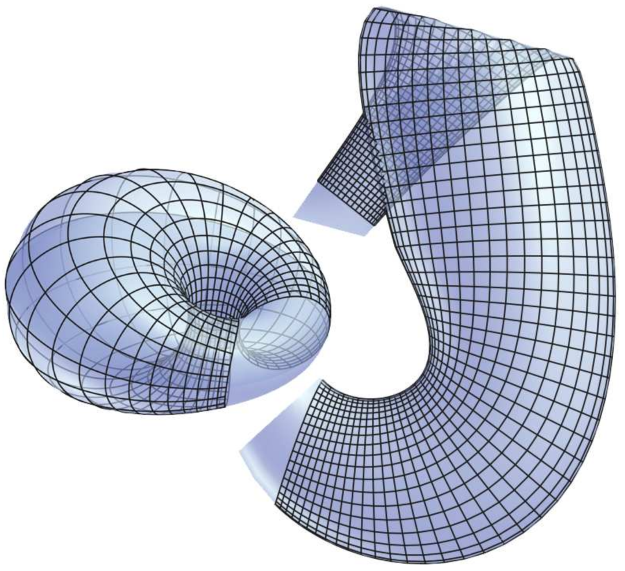


Figure 1: These examples use grids which are closed 'in one direction'. While orthogonality can be fulfilled reasonably well, the grid size varies considerably.

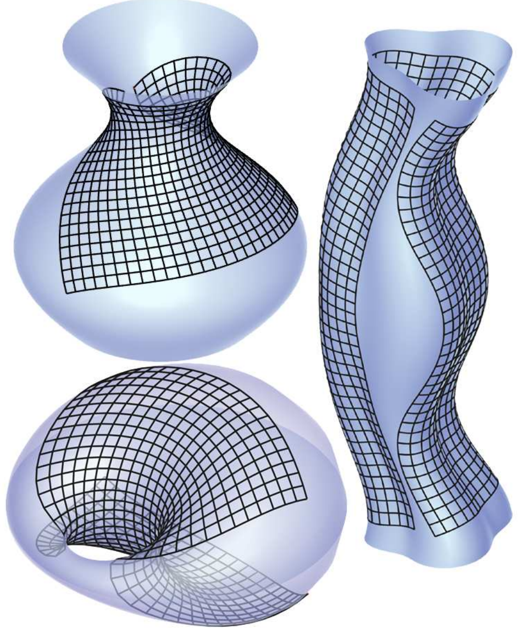


Figure 2: If the grid does not need to follow border rules, the degree of freedom allows the algorithm to produce meshes that almost precisely fulfill both orthogonality and aspect-ratio-preservation.

applied to a wide variety of graph related problems (e.g., removing of node overlapping [8] or the drawing of clustered graphs [5, 16]).

Over the years force based algorithms have been adopted to various other problems as well. X. Provot [13] introduced a mass and spring system for the simulation of cloth by modeling it as regular rectangular grid of $m \times n$ virtual masses which are connected by massless springs of natural length non equal to zero. F. Gruber and G. Glaeser [7] used a force directed approach for the construction of a minimal surface from given boundary curves. Similar to the work presented here they control the desired (ideal) length of edges by replacing them with imaginary springs between vertices. Moreover, mass-spring systems are used to overcome the computational expense of finite elements approaches for the simulation of deformable bodies (e.g., [10, 11]) at the cost of physical accuracy.

3. Algorithm

Let us assume that the basic surface is given by a differentiable mapping $x : \mathbb{R}^2 \to \mathbb{R}^3$ and a rectangular grid in parameter space is mapped onto this surface, yielding the initial grid. In order to obtain orthogonality and local equilaterality our force-directed algorithm uses simple forces to push the vertices in directions that lead to a more orthogonal and equilateral grid than in the iteration before.

In the following discussion we will denote a vertex of the grid as p. Edges e connect two vertices p_1 and p_2. The grid can either be closed in one direction (u or v, see Fig. 1), in both directions (Fig. 6) or, the grid can be open in both directions (Fig. 2). Interior vertices refer

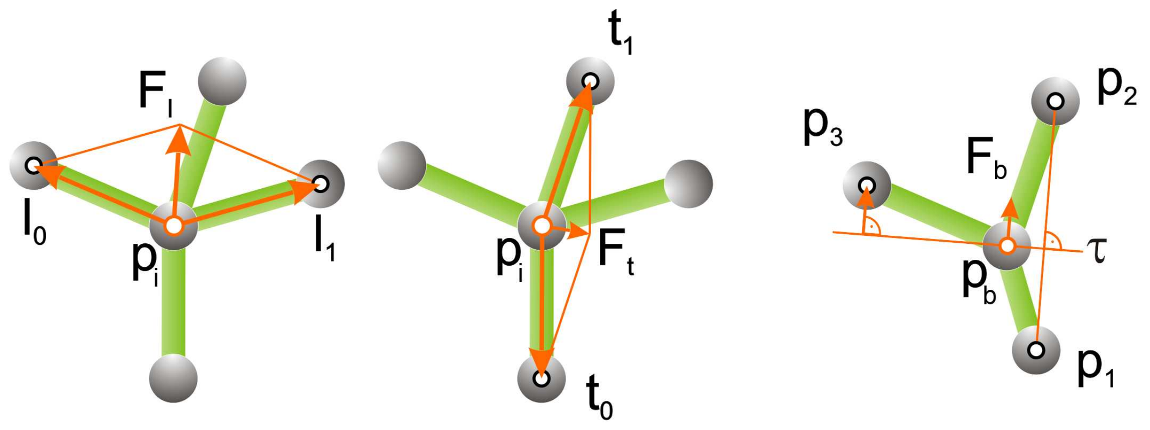


Figure 3: Calculation of the disposition once for a vertex p which lies on the boundary (right side) and once if the vertex is an interior point (left side). In that case the disposition $\mathbf{F}_i$ is the weighted sum of $\mathbf{F}_l$ and $\mathbf{F}_t$.

to vertices which have four neighbors and boundary vertices have three neighbors. In case of open meshes vertices at the corners of the grid are omitted by the algorithm since no forces are exerted on them and will therefore be excluded from the discussion.

At the beginning of each iteration the disposition p_{disp} of each vertex p is set to zero to sum up the disposing forces for the next iteration.

In a first step, we idealize the orthogonality of the projected incident edges on the tangential plane of each vertex (Fig. 3, Step 1 of Listing 1). In case of interior points p_i this is accomplished by adding small forces to the disposition, which should obtain straight angles between incident edges in longitudinal and transversal direction. As shown in Fig. 3 the disposition of p_i is the sum of the angle bisector of $\angle(l_0, p_i, l_1)$ and $\angle(t_0, p_i, t_1)$, weighted by a small factor d_i:

$$\begin{aligned} \mathbf{F}_i &= d_i \cdot (\mathbf{F}_l + \mathbf{F}_t) = \\ &= d_i \cdot ((\mathbf{l_0} - \mathbf{p_i})^* + (\mathbf{l_1} - \mathbf{p_i})^* + (\mathbf{t_0} - \mathbf{p_i})^* + (\mathbf{t_1} - \mathbf{p_i})^*) \end{aligned} \tag{1}$$

To avoid folding at the boundary of the grid, each boundary point is slightly displaced in the direction of 'the center' of its neighboring points. To be more specific, let us assume a boundary point p_b with its two neighboring boundary points p_1 and p_2 and interior point p_3, as depicted in Fig. 3. Furthermore, τ is the plane passing through point p_b with normal $(\mathbf{p}_2 - \mathbf{p}_1)^*$. The resulting disposition of p_b is then given by the normal of τ weighted with the signed distance between τ and p_3 and a small constant d_b, mathematically

$$\mathbf{F}_b = d_b \cdot \mathit{SignedDist}(\mathbf{p}_3, \tau) \cdot (\mathbf{p}_2 - \mathbf{p}_1)^* \tag{2}$$

Afterward, in a second step, we idealize local equilaterality for every edge by calculating the deviation $\vec{\delta}$ of its actual length from a predefined ideal length and add $\vec{\delta}$ multiplied by a small factor d_e to the disposition of the incident vertices of the edge (see Step 2 of Listing 1 and, e.g., [3, 4, 6]).

As a side note, we should point out that if the control of edge lengths is completely omitted the successional process of orthogonalization becomes instable. In the current implementation we use a constant ideal length for all edges of the grid which automatically considers the curvature of the surface. In other words, high curvature results in locally small grid size and

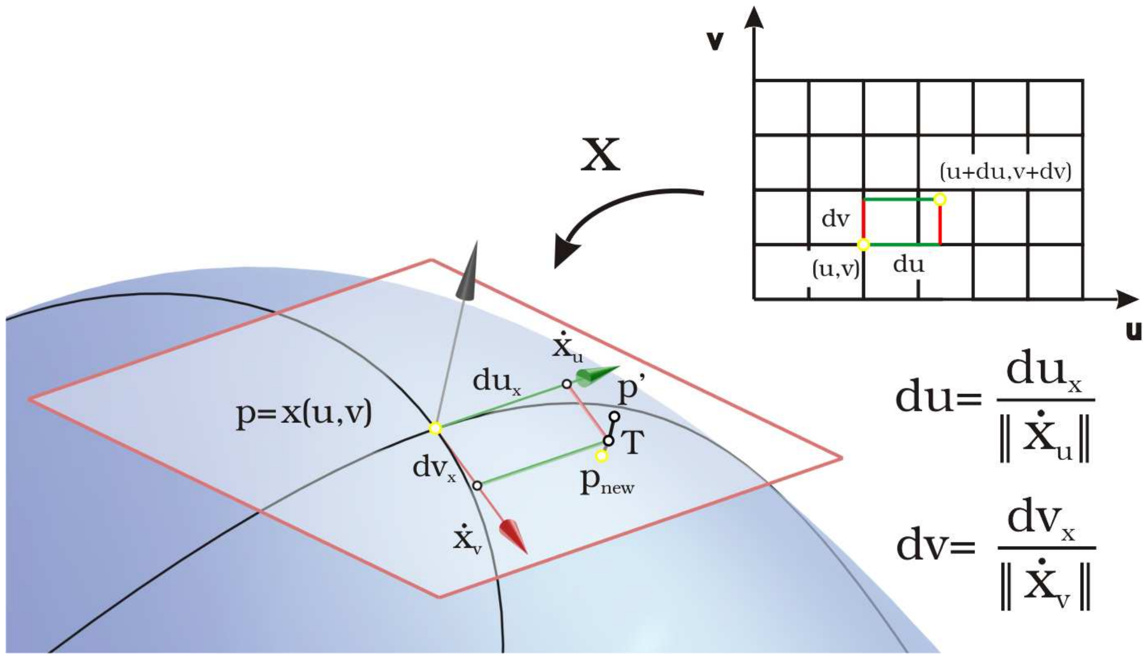


Figure 4: Linear approximation of the normal projection of a point P on the tangential plane of $x(u, v)$ onto the parametric surface S.

low curvature yields larger grid size. However, we currently experiment with ideal lengths in analytical relation to the local curvature of the surface.

In a final step we add p_{disp} to the actual position of p. For the following explanation we denote $p' = p + p_{disp}$. Since in general p' does not lie on the surface anymore, p' has to be projected back onto the parametric surface S (see Fig. 4 and Step 3 of Listing 1). Furthermore, let us denote $p = x(u, v)$ and $p_{new} = x(u+du, v+dv)$, where x is the parametric representation of S. To find a linear approximation of p_{new}, we calculate at first the normal projection T of p' onto the tangential plane of point p. The step size of u and v result in

$$du = \frac{du_x}{\|\dot{\mathbf{x}}_u\|} \qquad dv = \frac{dv_x}{\|\dot{\mathbf{x}}_v\|} \tag{3}$$

where du_x and dv_x are the coordinates of point T in respect to the basis $\{\dot{\mathbf{x}}_u, \dot{\mathbf{x}}_v\}$. Secondly, to improve the numerical accuracy of the linear approximation, p' is projected onto the tangential plane of p_{new}. Although the process can be repeated multiple times to increase to accuracy we found out that one repetition is sufficient for a stable progress. The pseudocode given in Listing 1 summarizes the steps explained above.

```
function IterationStep() {
  for(each vertex p)
   p.disp = 0;

  // step 1: idealize orthogonality
  for(each vertex p) {
    if (p == inner point} {
      p.disp += $\mathbf{F}_i^{res}$;
    }
    else if (p == boundary point) {
      p.disp += $\mathbf{F}_b$;
    }
  }

  // step 2: idealize edge length
  for(each edge e) {
    ideal = FindIdealLength(e);
    $\Delta l$ = ideal - e.actualLength;
    e.p1.disp -=  $d_e \cdot \Delta l \cdot$ e.direction;
    e.p2.disp +=  $d_e \cdot \Delta l \cdot$ e.direction;
  }

  // step 3: disposition and back-projection to surface
  for(each vertex p) {
    p.pos += p.disp;
    p.pos = S.NormalProjection(p.pos);
  }
}
```

Listing 1: Pseudocode for one iteration of the algorithm which calculates the disposition of each vertex p.

This code is repeated for each iteration until the solution converges to specified thresholds in regard to orthogonality and equilaterality (as defined in Section 4). Since convergence to these thresholds can not be guaranteed in all cases the algorithm terminates also if the changes in disposition are negligeable from one iteration to the next.

The user can change the behavior of the algorithm during runtime by altering the ideal edge length, d_i, d_b and d_e. These parameters should take on values between 1% and 5% of the average edge length of the grid. Otherwise the disposition during one iteration may exceed a critical amount which leads to instability of the process. Modifying d_e changes the ideal length preserving force, whereas higher values for d_i and d_b enforce the orthogonality of the mesh. It can be observed that orthogonality and equilaterality typically show *inverse behavior* which means increasing one of them decreases the other one. The intermediate results are displayed so that the user has immediate feedback.

4. Results

To assess the quality of our algorithm with respect to orthogonality the maximum deviation from orthogonality (MDO) and the average deviation from orthogonality (ADO) is measured. Following V. Akcelik et al. [1] the former is given by

$$MDO = \max_{i,j}\left(|90^\circ - \theta_{i,j}|\right) \tag{4}$$

and the latter is calculated from

$$ADO = \frac{\sum_{i=2}^{n_x-1}\sum_{j=2}^{n_y-1}|90^\circ - \theta_{i,j}|}{(n_x-2)\cdot(n_y-2)} \tag{5}$$

where the angle $\theta_{i,j}$ is the maximum angle between longitudinal and transversal edges at grid position (i,j). These values are not calculated directly on the discretized mesh since the discretization by itself introduces some error. Instead, grid interpolating spline curves are used.

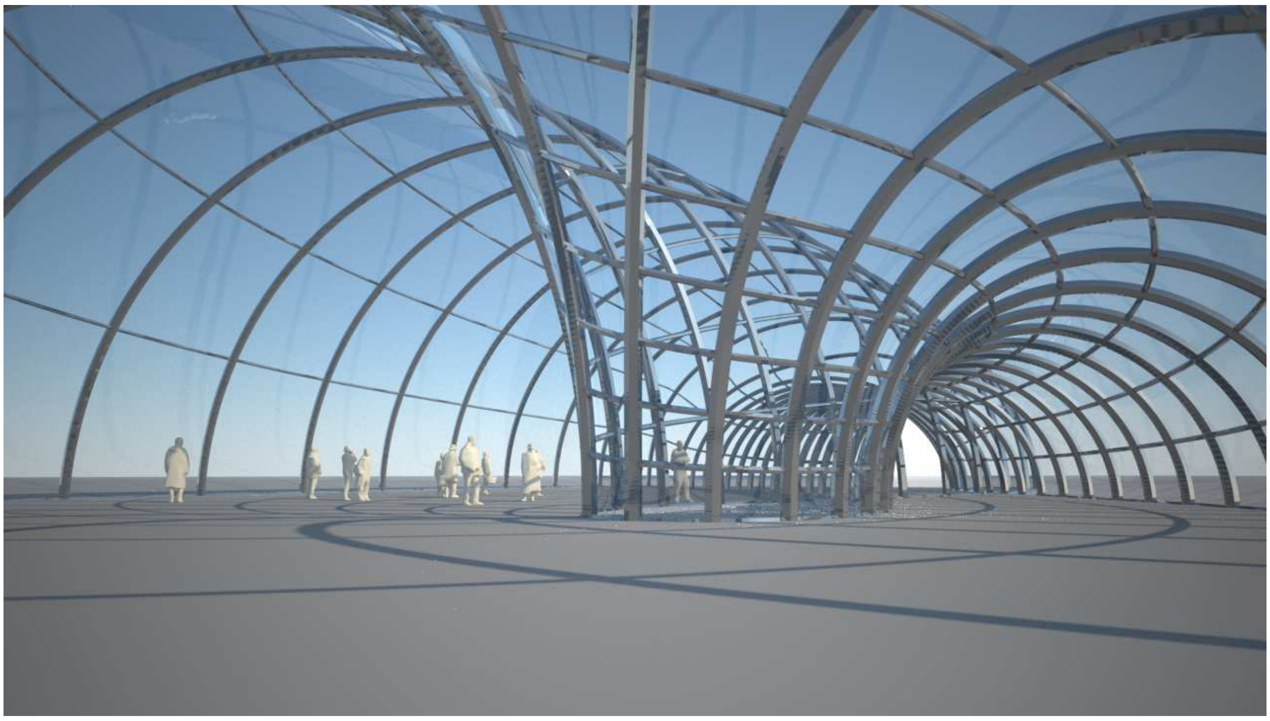


Figure 5: Rendering of a Klein Bottle (from the inside).
The orthogonality of the grid has an aesthetic impact.

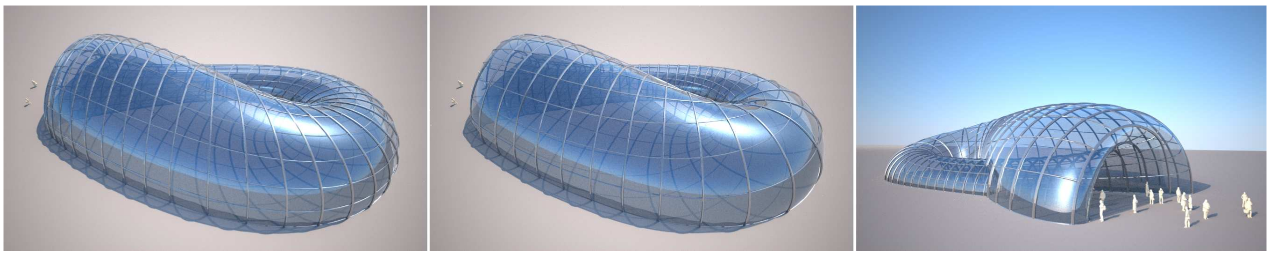


Figure 6: Concept renderings of a hall which in fact is a Klein Bottle cut in half. The girders in the middle and right image where laid out with the presented algorithm. The left image shows the initial configuration.

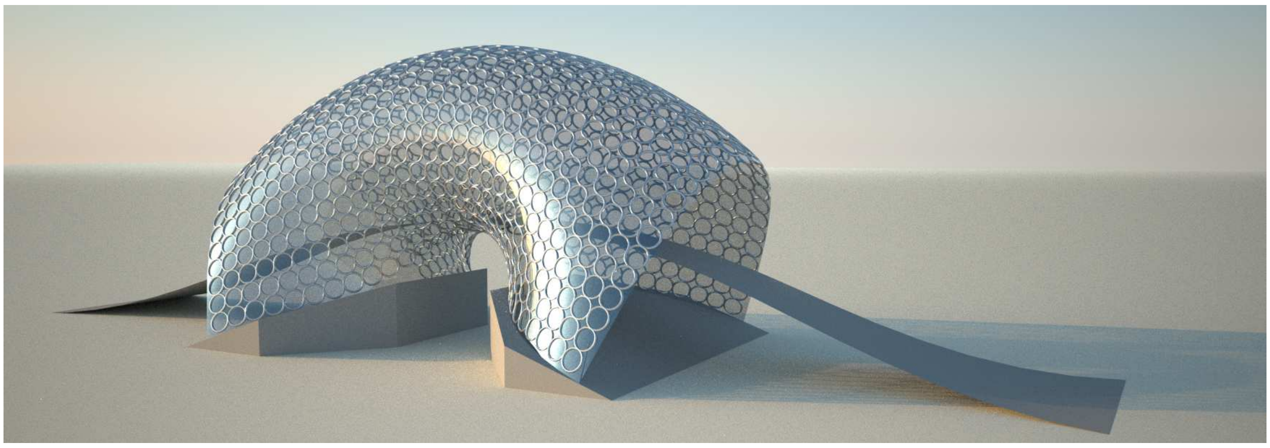


Figure 7: A concept for a bridge which consists of a part of a Dupin cyclide which is enclosed with metallic rings.

Table 1: Results for the examples shown in Fig. 5 through Fig. 9(b). The subscripts b and a indicate if the measurements were done before or after the algorithm. The type denotes if a grid is open (C_0), closed in one direction (C_1) or is a double closed grid (C_2).

Scene	**Type**	**MDO**	**ADO**	**MDE**	**ADE**
Klein Bottle$_b$	C_2	33.78°	11.75°	404%	90%
Klein Bottle$_a$	C_2	5.43°	0.48°	152%	28%
Tower$_b$	C_1	40.47°	10.55°	134%	29%
Tower$_a$	C_1	1.43°	0.28°	15%	9%
Tower (Rings)	C_0	5.84°	1.73°	8%	4%
Bridge	C_0	5.37°	2.28°	12%	4%
Function Graph	C_0	4.30°	1.09°	9%	3%

Furthermore we measure the maximum deviation from equilaterality (MDE) and the average deviation from equilaterality (ADE) which are given by

$$\begin{aligned} MDE &= 100 \cdot \left(\max_k \left(\frac{L_k}{l_k} \right) - 1 \right) \\ ADE &= 100 \cdot \left(\frac{1}{n_q} \sum_{k=1}^{n_q} \frac{L_k}{l_k} - 1 \right) \end{aligned} \tag{6}$$

where n_q is the number of quads of the grid. L_k is the maximum and l_k the minimum side length of quad k.

In the current implementation one of these four measurements, depending on the requirements concerning orthogonality and equilaterality, is used as break condition of the algorithm (see Section 3).

4.1. Application in architectural design

Figures 5 through 9(b) show examples from architectural design. Figures 5 and 6 show a hall which is based on a parametric representation of a Klein Bottle with its double closed parameter lines as input grid where the result was cut in half afterward. As evident from Table 1 this type of double closed grid achieves a high degree of orthogonality. If the grid is closed, however, a given number of u and v lines may not be well suited to achieve local equilaterality. In contrast, for the tower in Fig. 9(a) a grid, which is open on one side, was used. This already leads to better local equilateral quadrangles.

Opening the grid on both sides results in almost equilateral quadrangles, which is illustrated in Fig. 9(b), by inscribing circles into each quadrangle. Figure 7 is another example were an open mesh was used. Table 1 shows measurements for those Figures.

As Figs. 7 and 9(b) show, the algorithm of finding ratio-preserving orthogonal quadrangular grids turns out to be a good tool for finding circular pavements of general surfaces. To fulfill the task as good as possible, both degrees of freedom are necessary, i.e., the grid should be open in both directions. Figure 8 shows how the ‘skin’ of the surface changes (contracts) from the given grid and finally converges to a visually pleasing (aesthetic) pattern.

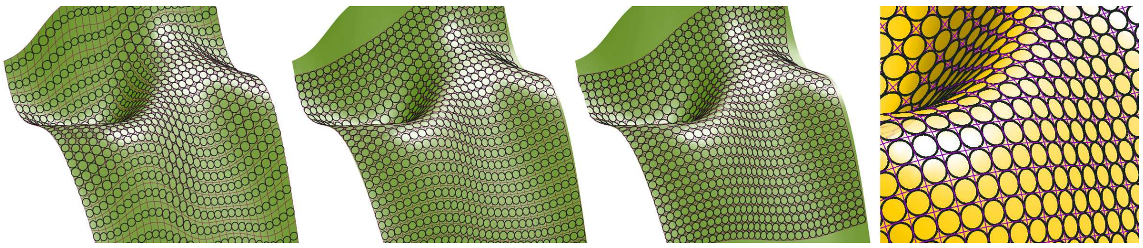
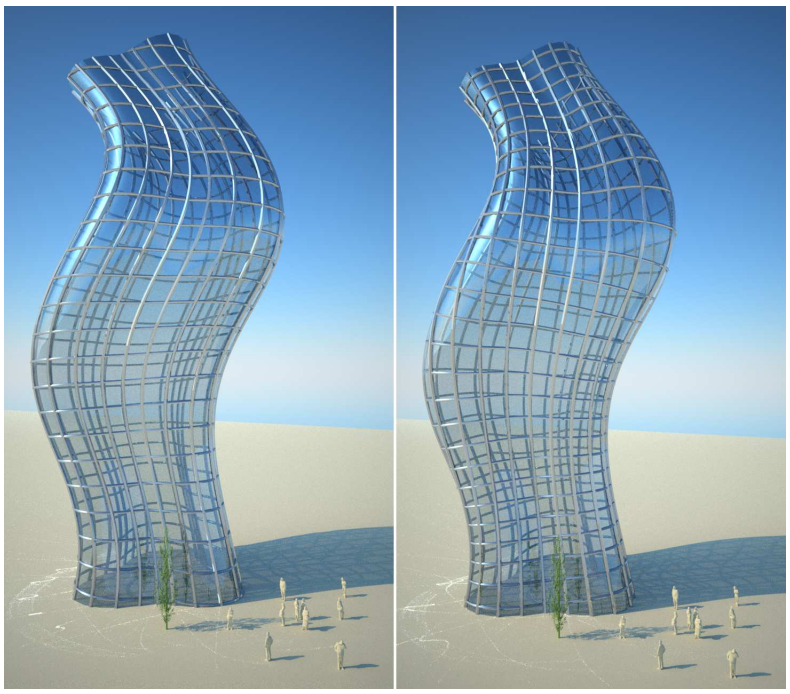


Figure 8: The converging (contracting) grid is perfectly suitable for a circular pattern that covers the surface. Left: Starting position, right (yellowish): converged grid.


(a) A tower, once with the initial grid (left side) and once after our algorithm was applied (right side). Since the mesh is only closed in one direction, the algorithm can more easily obey the equilateral condition and therefore produces locally square quadrilaterals.

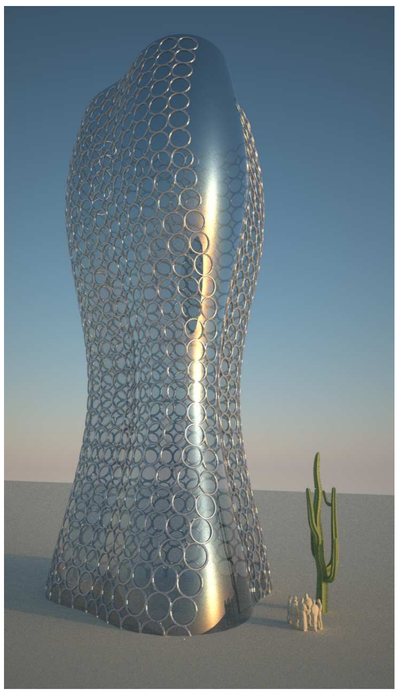

(b) Another tower with a ratio-preserving orthogonal grid in which circles were inscribed.

Figure 9: Concept renderings of two different towers

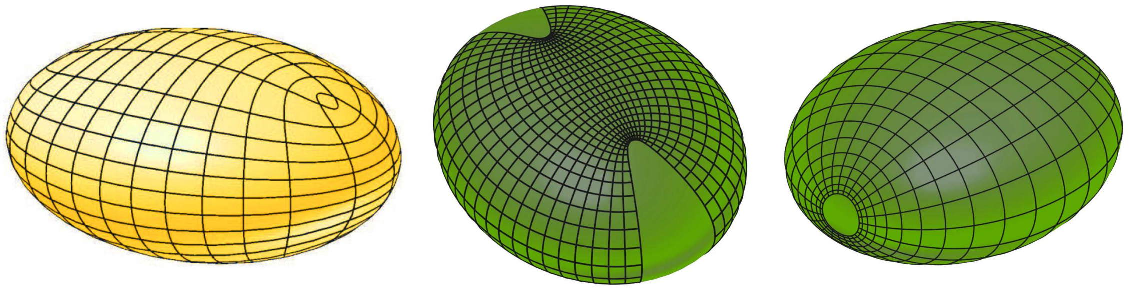


Figure 10: A general ellipsoid (no surface of revolution). Left: the principal curvature lines calculated with analytical methods. Middle: Approximation of the principal curvature lines found by our algorithm. Right: altering the parameters leads to a completely different (and non-trivial) orthogonal net.

4.2. Application in geometry

Among the infinite number of possible orthogonal grids on a surface, the lines of principal curvature play an important role both in theoretical and applied geometry. They are usually found by solving differential equations (see [12]). By the way of example, Fig. 10 shows these lines for a general ellipsoid. The principal curvature lines of such an ellipsoid form a perfect mesh, with four 'problematic points' (the so-called umbilical points) where the surface is osculated by a sphere. These points mostly implicate a non-trivial topology of the principal curvature field. However, the topology of the principal curvature lines usually differs from the topology of a quadrangular grid and therefore it is impossible for our algorithm to find them. Nevertheless, we can find alternative solutions to such curvature lines, depending on the parameters values d_i, d_b and d_e, as shown in Fig. 10.

5. Conclusions

In this paper we described a force-based algorithm for calculating orthogonal quadrangular grids on arbitrary double-curved parametric surfaces. Our algorithm uses simple forces to push vertices in directions that lead to a more orthogonal grid than in the iteration before. The algorithm works with grids which are open or closed in one or two directions. The behavior of the algorithm can be altered via the ideal edge length and the three force parameters d_e, d_i and d_b. Images and quality indicators throughout the paper showed results which where achieved with the presented algorithm. Currently our method works only on parametric surfaces (function graphs, parametric representations, implicit functions and Bezier and NURBS surfaces). It would be useful, however, to extend the algorithm to polygonal surfaces since they are not less common in architecture and design than parametric surfaces.

Local equilaterality can be best achieved if the grid is open in both directions. In case of a closed grid a suitable number of parameter lines has to be chosen to achieve local equilaterality. The quality may be improved by adaptively adding u and v parameter lines. We also attempt to incorporate the local curvature of the surface into the calculation to be able to use individual ideal edge lengths. This should give a better convergence toward local equilaterality since the curvature naturally corresponds to the grid size.

References

[1] V. Akcelik, B. Jaramaz, O. Ghattas: *Nearly orthogonal two-dimensional grid generation with aspect ratio control.* J. Comput. Phys. **171**(2), 805–821 (2001).

[2] J. E. Castillo (ed.): *Mathematical Aspects of Numerical Grid Generation.* Society for Industrial Mathematics, 1987.

[3] A. Creek: *Forces of nature.* Master Thesis, University of Canterbury, 2001.

[4] P. Eades: *A heuristic for graph drawing.* Congressus Nutnerantiunt **42**, 149–160 (1984).

[5] Y. Frishman, A. Tal: *Dynamic drawing of clustered graphs.* EuroVis, 75–82 (2007).

[6] T.M.J. Fruchterman, E.M. Reingold: *Graph drawing by force-directed placement.* Software – Practice and Experience **21**, 1129–1164 (1991).

[7] F. Gruber, G. Glaeser: *Magnetism and minimal surfaces — a different tool for surface design.* Computational Aesthetics, 81–88 (2007).

[8] W. Li, P. Eades, N. Nikolov: *Using spring algorithms to remove node overlapping.* In APVis '05: Proceedings of the 2005 Asia-Pacific symposium on Information visualisation, Australian Computer Society, Inc. 2005, pp. 131–140.

[9] Y. Liu, H. Pottmann, J. Wallner, Y.-L. Yang, W. Wang: *Geometric modeling with conical meshes and developable surfaces.* ACM Trans. Graph. **25**(3), 681–689 (2006).

[10] M. Matyka, M. Ollila: *Pressure model of soft body simulation.* SIGRAD 2003.

[11] J. Mesit, R. K. Guha, S. Chaudhry: *3d soft body simulation using mass-spring system with internal pressure force and simplified implicit integration.* JCP **2**(8), 34–43 (2007).

[12] H. Pottmann, A. Asperl, M. Hofer, A. Kilian: *Architectural Geometry.* Bentley Institute Press, 2007.

[13] X. Provot: *Deformation constraints in a mass-spring model to describe rigid cloth behavior.* Graphics Interface 1995, Canadian Human-Computer Communications Society, pp. 147–154.

[14] N. Quinn, M. Breuer: *A force directed component placement procedure for printed circuit boards.* IEEE Transactions on Circuits and Systems, 377–388 (1979).

[15] J.F. Thompson, B.K. Soni, N.P. Weatherill (eds.): *Handbook of Grid Generation.* CRC Press, 1998.

[16] G. Wallner: *Force directed embedding of hierarchical cluster graphs.* ROGICS 2008: Proceedings of the International Conference on Relations, Orders and Graphs: Interaction with Computer Science, 2008.

Received August 7, 2010; final form November 15, 2010