Proceedings of the IMProVe 2011
International conference on Innovative Methods in Product Design
June 15th – 17th, 2011, Venice, Italy

Shape optimization of smooth surfaces with arbitrary topology

Przemysław Kiciak [(a)]

[(a)] Institut Matematyki Stosowanej i Mechaniki, Uniwersytet Warszawski

Article Information

Keywords:
B-spline surfaces,
Curvature continuity,
Shape optimization

Corresponding author:
Przemysław Kiciak
Tel.: +48 22 55 44 501
e-mail: przemek@mimuw.edu.pl
Address: Institut Matematyki Stosowanej i Mechaniki, Uniwersytet Warszawski, ul. Banacha 2, 02-097 Warszawa, Poland

Abstract

A popular method of constructing a smooth surface with arbitrary topology is choosing an irregular mesh and applying iteratively the Catmull-Clark algorithm. The sequence of meshes obtained in this way converges to a surface, whose curvature is continuous, except in a vicinity of special mesh elements; apart from the curvature discontinuity there, the limiting surface exhibits undesirable undulations, visible on curvature images. In this paper a shape optimization method is described, modifying vertices of the mesh to produce a surface with the curvature continuous and the undulations significantly reduced.

1 Introduction

To be convenient for designers, representations of surfaces with complicated shapes must have a rather small complexity. The number of points manipulated directly by a designer may be reduced by allowing the designer to specify only the surface boundary and using an automatic procedure to compute the other control points. A typical approach for surfaces of arbitrary topology is approximating smooth surfaces by planar triangles, and solving geometric flow equations (see e.g. [2], [14], [15]). The optimization of shape explicitly using B-spline representations of surfaces is harder, though this approach turned out to be competitive for tensor product patches [8]. The main difficulty in extending the latter approach to surfaces of arbitrary topology is the necessity of constructing an appropriate function space. In this paper such a construction has been developed for surfaces with curvature continuous, represented by irregular meshes, and made (almost entirely) of bicubic polynomial patches.

2 Mesh refinement

In this section we recall the mesh representation of surfaces and the mesh refinement algorithm; it may be used to increase the resolution of surface representations in order to represent finer details of shape, and in the construction of special element domain nets described in Section 3. A mesh consists of vertices, edges and facets. An edge may belong to one or two facets; in the former case it is called a boundary edge, and in the latter case it is called internal. The vertices, whose all incident edges are internal, are called internal. Any facet with all vertices internal is also called internal. Other vertices and facets are called boundary vertices and facets. The internal vertices, whose numbers of incident edges are other than four, and the non-quadrangular internal facets, are called **special elements** of the mesh.
The mesh refinement algorithm is implemented using two operations, called doubling and averaging.

Doubling is a construction of a new mesh, with copies of all original facets and with additional facets, corresponding to the original edges and vertices. A~facet corresponding to an edge has four edges and vertices; its two opposite edges have the vertices, which coincide with the original edge end points, and thus the facet is reduced to a line segment. A facet corresponding to a vertex is degenerated to a point; the number of its vertices is the number of edges incident with the original vertex (plus 1 for a boundary vertex, fig. 1a)

Averaging is a construction of a new mesh, whose vertices correspond to the facets of the given mesh; they are positioned at the gravity centres of the sets of vertices of those facets. Each internal edge produces an edge between the vertices corresponding to the two facets sharing that edge. Facets of the new mesh correspond to the internal vertices of the given mesh (fig. 1b).

Refinement is a doubling followed by n averaging operations. Note that

- All facets of a mesh obtained by doubling, corresponding to the edges of the original mesh have four vertices and edges. Moreover, all internal vertices are incident with four edges. Thus all special elements of a mesh obtained by doubling are facets.
- Averaging converts special vertices to special or boundary facets and special facets to special or boundary vertices. If n is even, then the mesh refinement produces a mesh, whose all special elements are facets. If n is odd,

then all special elements of the result are vertices. No new special elements are produced by the operations discussed here.

- If a mesh has no special element, then the refinement operation implements the Lane-Riesenfeld algorithm [11] (see also [12]). Iterating it produces a sequence of meshes convergent to a limiting surface. Any part of a mesh from this sequence, consisting of *n* x *n* facets and *n+1* x *n+1* vertices, which form a square array with the edges joining the neighbouring vertices in the columns and rows, is the B-spline representation with equidistant knots of a polynomial patch of degree *(n,n)*, being a part of the limiting surface. The convergence of the sequence of meshes is fast: some estimate of the distance between the mesh and the limiting surface, after the refinement is decreased four times.
- If a mesh has special elements, then the limiting surface consists of an enumerable set of polynomial patches of degree *(n,n)*. The refinement operations with *n*=2 and *n*=3 are special cases of the Doo-Sabin and Catmull-Clark algorithms (see [3], [1], [12]).

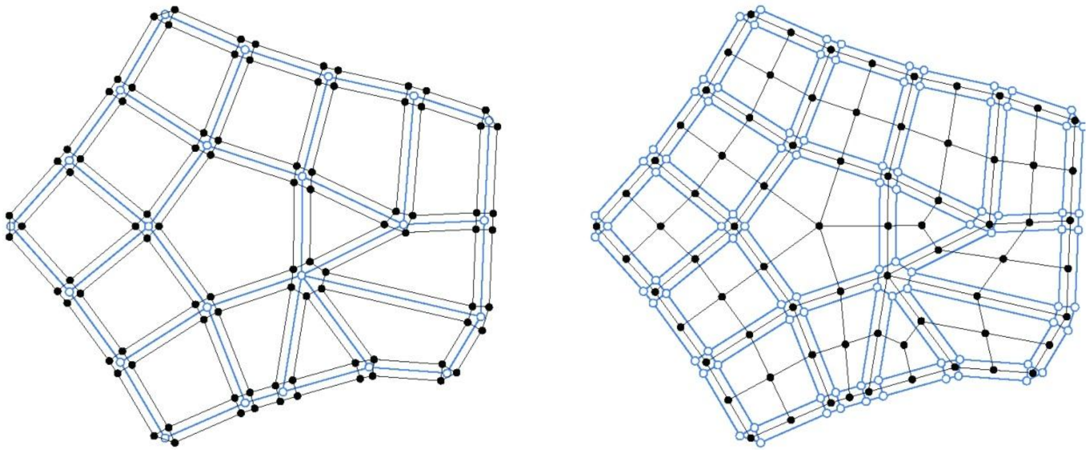
Fig. 1: Doubling (a) and averaging (b) of a mesh

A mesh to be optimized must have all facets quadrangular. The shortest path between any two special vertices or between a special vertex and a boundary vertex must consist of at least two edges, which belong to different facets. Usually two iterations of the Catmull-Clark algorithm suffice to produce a mesh satisfying these conditions from an arbitrary mesh.

3 Space and basis construction

The surface representation may be written in the form

$$p = \sum_{i=1}^{n} p_i \varphi_i \qquad (1)$$

where $p_1, \dots, p_n \in \mathbb{R}$ are the mesh vertices, and $\varphi_1, \dots, \varphi_n$ are functions defined in the domain made of elements described below. The shape optimization is done by modifying some vertices of the mesh.

The surface consists of polynomial patches; each of them corresponds to an internal facet of the mesh. There are two types of the patches. An **ordinary patch** corresponds to a facet surrounded by eight other facets, whose vertices may be arranged in 4 x 4 array (fig. 2). Such an assembly of 9 facets, 16 vertices and 24 edges will be called an **ordinary patch control net**. It is associated with a so called **ordinary element**, which is a unit square.

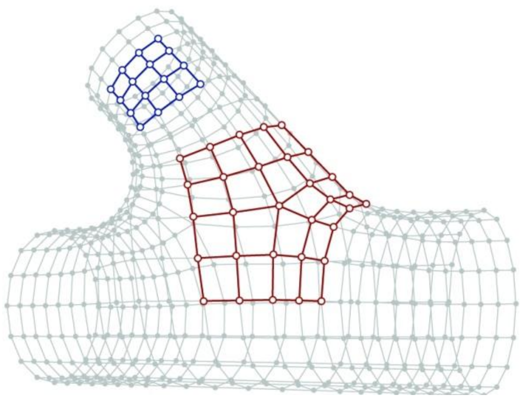
Fig. 2: An ordinary patch control net and a Sabin net of radius 2 in a mesh

There are 16 basis functions ϕ, nonzero in any ordinary element; they are associated with the 16 vertices of the ordinary patch control net. The restriction of ϕ_i to the ordinary element is the tensor product of the polynomials, which describe cubic B-spline functions with integer knots. Thus the ordinary patch control net is the bicubic B-spline representation of the ordinary patch.

An example of a function ϕ_i, nonzero in 16 adjacent ordinary elements, is shown in fig. 3.

Parts of a mesh surrounding special elements are referred to as Sabin nets [4]. In our case all special elements are vertices incident with $k \neq 4$ edges; a part of a mesh surrounding such a vertex, made of kr^2 facets, is called a Sabin net of radius r. A Sabin net of radius 2 (fig. 2) represents k **special patches**.

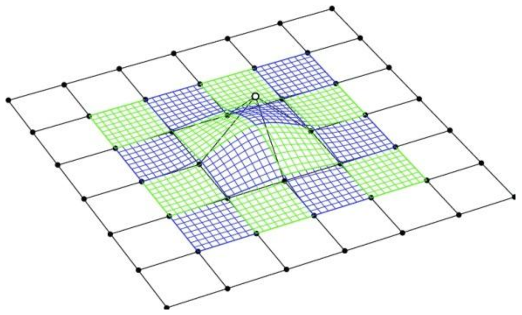

Fig. 3: A basis function with support consisting of ordinary elements

A **special element** is the domain of a parameterization of the k special patches represented by a Sabin net. The idea of construction of a special element and basis functions nonzero in it is as follows: let $\mathfrak{p}_k$ denote the composition of the refinement operator (doubling followed by 3 averaging operations) and the operation extracting a Sabin net of radius 3 around a vertex incident with k edges from the refined mesh. The operator $\mathfrak{p}_k$ has the greatest eigenvalue 1 and the multiplicity of the next greatest eigenvalue is 2. There exists a planar Sabin net with k axes of symmetry, which is an eigenvector of $\mathfrak{p}_k$ associated with that double eigenvalue, (see fig. 4).

In the construction its size is chosen so as to obtain the area Ω_k described below of measure k, which is relevant for the choice of the constants C_i (see Section 4). Such a Sabin net is called a **special element domain net**. Subnets of a special element domain net represent $3k$ bicubic patches, which surround a curvilinear polygon Ω_k with k corners. The area Ω_k is divided into k curvilinear quadrangles $\Omega_0^k, \dots, \Omega_{k-1}^k$, identical up to rotations and axial symmetries. A polynomial mapping $\delta_j\colon [0,1]^2 \to \bar{\Omega}_j^k$ is defined for $j = 0, \dots, k-1$. Then Sabin nets in $\mathbb{R}^3$ are constructed as follows: the vertices have the x,y coordinates as in the domain net. The z coordinate of one of $6k+1$ vertices of the Sabin subnet of radius 2 is set to 1, and the other vertices have $z = 0$. The surface made of the ordinary patches represented by such a net is a graph of a scalar function ψ_i defined in an area with the hole Ω_k. A set of polynomials μ_{ij} of degree (9,9) is constructed, such that the function ϕ_i defined piecewise as $\phi_i(x) = \mu_{ij}(\delta_j^{-1}(x))$ for $x \in \Omega_j^k$ is of class $C^2(\Omega^k)$ and it is a numerical solution of the triharmonic equation $-\Delta^3\varphi = 0$ in Ω^k, with the boundary condition fixing the function value and the first and second order cross derivatives at the boundary of Ω^k, determined by the function ψ_i. As a consequence, the special patches have polynomial parameterizations of degree (9,9), and if the surface represented by a mesh is regular, then its curvature is continuous. Examples of basis functions constructed in this way are shown in fig. 5. Details of this construction are described in [6], where it has been used to fill polygonal holes in mesh surfaces (see also [7]). The sum of the basis functions in each element (special as well as ordinary) is 1, and due to this fact the relation between the surface and the mesh representing it is an affine invariant.

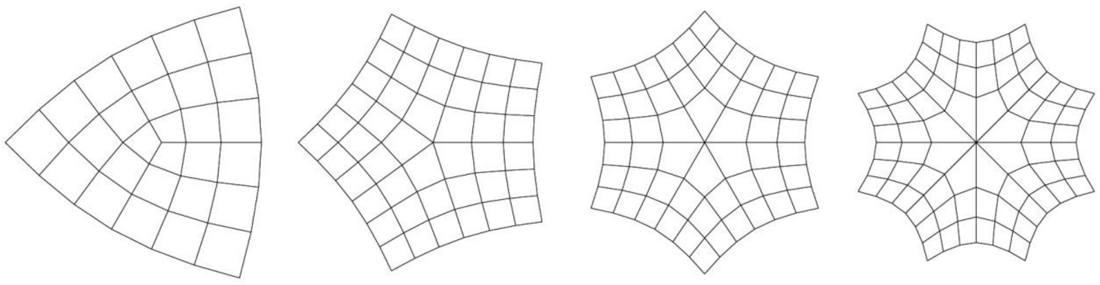

Fig. 4: Special element domain nets for k = 3, 5, 6, 8

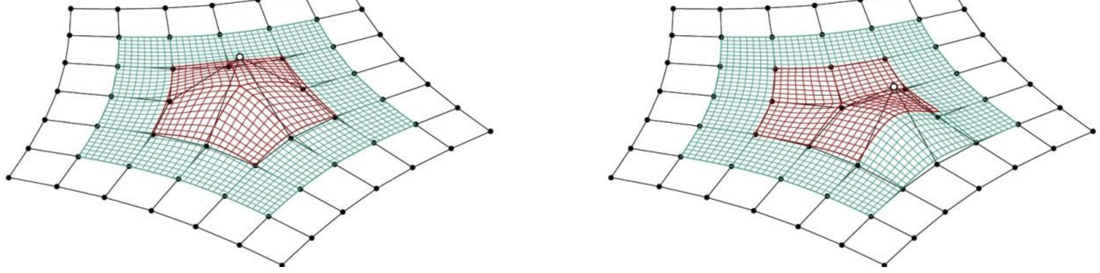

Fig. 5: Examples of basic functions nonzero in a special element

4 Optimization criterion

It is assumed that the surface has a boundary, which is fixed together with the tangent plane and mean curvature at each point of the boundary. This is done by fixing all vertices of the mesh, whose distances from any boundary vertex (measured by the number of edges in between) are less than 3. If necessary, one can fix additional vertices, which is a method of imposing constraints on the surface. Such constraints are called soft, as opposed to hard constraints, having the form of linear equations (to be satisfied by mesh vertices), expressing e.g. interpolation conditions for the surface. Examples of using constraints are discussed in section 6. The shape optimization is done by searching a minimum of the functional

$$S(p) \overset{\text{def}}{=} \int_{\mathcal{M}} \|\nabla_{\mathcal{M}} H\|_2^2 d\mathcal{M} \tag{2}$$

Here $\mathcal{M}$ is a surface, whose parameterization p is represented by a mesh, as described in previous sections. The symbol $\nabla_{\mathcal{M}} H$ denotes the gradient of the mean curvature on the surface; the integration is done with respect to the surface measure. The value of the functional S may be computed by integration over the elements:

$$S(p) = \sum_{i\in\epsilon} \int_{E_i} \nabla H G^{-1} \nabla H^T \sqrt{\det G}\, dE_i \tag{3}$$

4.1 where ϵ denotes the set of element indices, E_i is an element, and G is the matrix of the first fundamental form of the parameterization p. The mean curvature H is interpreted here as a function defined in the elements, and $\nabla H = \left[\frac{\partial H}{\partial u}, \frac{\partial H}{\partial v}\right]$ is the ordinary gradient. Though the evaluation of the functional S may be done using derivatives of the parameterization, the functional is parameterization independent, i.e. its value is determined by the shape of the surface $\mathcal{M}$. This causes a trouble for the numerical algorithm, as the minimization problem may not have a unique solution. To overcome this problem, a regularization term is added. The optimization goal is to minimize the functional

$$F(p) \overset{\text{def}}{=} S(p) + cQ(p) \tag{4}$$

where c is a positive constant discussed later, and

$$Q(p) \overset{\text{def}}{=} \sum_{i\in\epsilon} C_i \int_{E_i} \|P\nabla\Delta p\|_F^2 dE_i \tag{5}$$

The letter P denotes the orthogonal projection of $\mathbb{R}^3$ on the tangent plane of the surface; this projection is supposed to weaken the influence of the surface curvature on the value of the functional Q, which increases with the growth of undulations of constant parameter curves of the parameterization p over the elements. The coefficients C_i are arbitrary positive constants. If a mesh represents a tensor product bicubic B-spline patch, then one can take the same C_i for all elements. Experiments show that taking $C_i = C = const$ works also for meshes with special elements, i.e. it makes the optimization method convergent. However, the regularization term affects the shape of constructed surfaces, and the effect is most pronounced in a vicinity of special patches. The resulting quality degradation may be reduced by taking smaller coefficients C_i for special elements and their neighbouring ordinary elements. Taking C_i too small causes troubles with the convergence of the optimization algorithm. Studying the influence of the parameters C_i on the result and developing alternative regularization methods are open problems. If an isometry is applied to the surface, the values of both terms in Formula (4) remain unchanged. The choice of the factor

$$c = \frac{|\epsilon|^2}{D_{\mathcal{M}}^4}$$

where $D_{\mathcal{M}}$ is the diameter of the (fixed) surface boundary and $|\epsilon|$ is the total number of elements, is based on the analysis of behaviour of S and Q when a homotetia is applied to the surface or to the parameterization domain (see [8]). With c chosen as above any homotetia changes both terms in (4) by the same factor, which ensures the invariance of the construction with respect to geometric similarities.

5 Optimization algorithm

The optimization algorithm is based on the classical Ritz method. The arguments of the function

$$\tilde{F}(\dots, x_k, y_k, z_k, \dots) \overset{\text{def}}{=} F\left(\sum_k p_k \phi_k\right) \tag{6}$$

are the coordinates of the non-fixed mesh vertices. The numerical procedure is searching for a zero of the gradient of $\tilde{F}$. Subsequent approximations x_i of the solution may be constructed using an iterative procedure described in [8], where it was used to optimize the shape of bicubic B-spline tensor product patches. During an iteration, the system of linear equations

$$H^{(i)}\delta = -g^{(i)} \tag{7}$$

is set up, where $g^{(i)}$ is the gradient and $H^{(i)}$ is the Hessian of the function $\tilde{F}$ at $x^{(i)}$. If the matrix $H^{(i)}$ is positive-definite, then the system [7] is solved and then, if $\tilde{F}(x^{(i)} + \delta) < \tilde{F}(x^{(i)})$, the procedure takes $x^{(i+1)} = x^{(i)} + \delta$; this is a Newton method step. If the Hessian matrix is not positive-definite or the Newton method does not decrease the function value, minimization along the Levenberg-Marquardt trajectory [5] is done, i.e. searching for a minimum of the function

$$f(\nu) \overset{\text{def}}{=} \tilde{F}(x^{(i)} + \delta_\nu) \tag{8}$$

where δ_ν is the solution of the system of equations

$$\left(H^{(i)} + \nu I\right)\delta_\nu = -g^{(i)} \tag{9}$$

This method makes it possible to use starting points insufficient for the Newton method. The formulae to evaluate (using quadratures) the function $\tilde{F}$, its gradient and the Hessian may be found in [8].

The Hessian matrix is sparse, with nonzero coefficients scattered irregularly. By renumbering of variables it is possible to transform it to a matrix with an irregular band containing relatively few zeros, which may be decomposed at a reasonable cost using the Cholesky's method. The most time-consuming part of the algorithm is the computation of the Hessian coefficients and therefore if the Newton method step is effective, i.e. in addition to decreasing the value of $\tilde{F}$ it decreases the length of gradient by the factor at least 4, the Hessian matrix and its decomposition factor are reused in subsequent iterations.

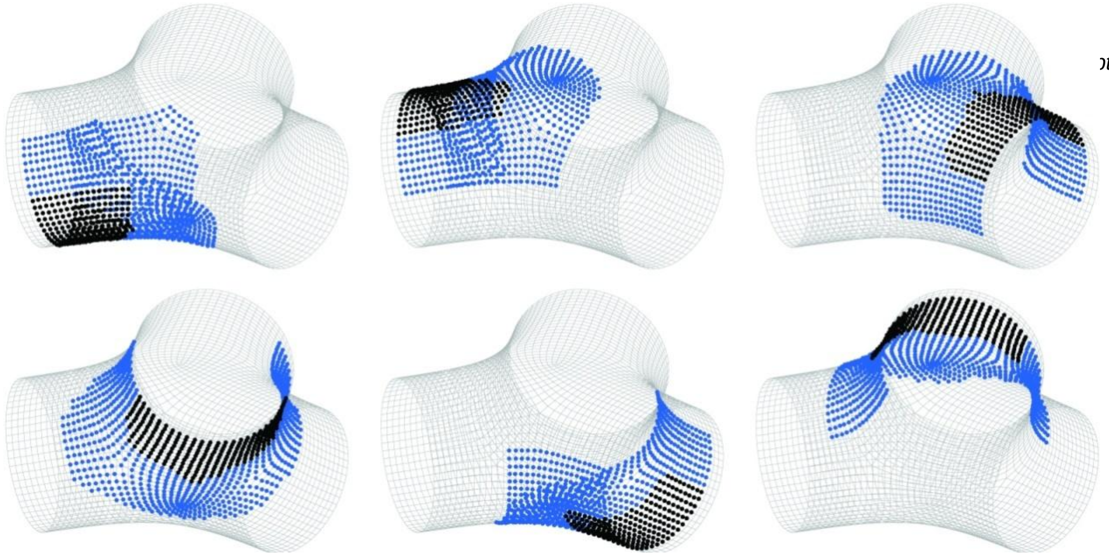

Fig. 6: Blocks of vertices and their overlapping parts

The rate of convergence is thus decreased, but the total computation time is shorter.

The algorithm outlined above works well enough if the number of unknown variables is moderate --- up to about 15000 (i.e. approximately 5000 mesh vertices).

Beyond that the Cholesky's decomposition is too inefficient. The optimization method appropriate for meshes with greater numbers of vertices is based on the domain decomposition. The idea is as follows: subsets of the set of vertices, called **blocks**, are chosen, which cover with overlaps the set of the non-fixed mesh vertices. Iterations of the minimization method described above are made for subsequent blocks until the Newton method step for each block is effective. Then the final phase of optimization is entered: the Newton method for the entire set of unknown variables. The system [7] is solved using the conjugate gradient method [10], with a preconditioner described below.

Let b be a block index and let

$$H_b^{(i)} = P_b R_b H^{(i)} R_b^T P_b^T$$

Where $H^{(i)}$ is the Hessian matrix of $\tilde{F}$ at $x^{(i)}$, R_b is the diagonal matrix with the k-th diagonal coefficient equal to 1 if the k-th argument of $\tilde{F}$ belongs to the b-th block, and 0 otherwise, and P_b is the permutation matrix, chosen so as to obtain

$$H_b^{(i)} = \begin{bmatrix} A_b & 0 \\ 0 & 0 \end{bmatrix}$$

where A_b is a nonsingular matrix with a possibly narrow band. The preconditioner is the matrix B such that

$$B^{-1} = \sum_b P_b^T \begin{bmatrix} A_b^{-1} & 0 \\ 0 & 0 \end{bmatrix} P_b.$$

The conjugate gradient method may fail, as the matrix $H^{(i)}$ may be undefinite even if all matrices A_b are positive-definite. In that case a number of additional iterations for the blocks are made.

Fig. 6 shows an example of blocks. Black dots mark the vertices, which belong to one block, and the vertices from more than one block are marked with blue dots. Let the diameter of a block be defined as the maximal distance (the number of edges in the shortest path) between its vertices. It is desirable (because of the rate of convergence of the optimization algorithm) to choose blocks, whose diameters are small relative to their numbers of vertices, and making rather wide overlaps. The algorithm used in the existing implementation choses **seeds**, i.e. some vertices of the mesh, and then it builds in the set of vertices a discrete Voronoi diagram for these seeds. Each block consists of the vertices from one Voronoi region extended to create overlaps. The resulting rate of convergence is often satisfactory, but further studies of the subject are planned.

Using the preconditioner made of blocks as described above is a special case of the additive Schwarz method [13]. With the block algorithm is possible to optimize the shape of surfaces represented by meshes having up to 30000 vertices. If an even finer representation of the surface was necessary, a multilevel method would have to be used; currently it is developed.

6 Results

Fig. 7 shows a surface obtained by iterating the Catmull-Clark mesh refinement operator and two surfaces with shape optimized. The mean curvature of the two surfaces, shown using the texture, is distributed much more evenly. The number of vertices of the mesh representing the surface in the middle is 1007, and 719 of these vertices were computed by the optimization procedure (the other vertices determine the surface boundary). For the surface on the right side there are 3743 vertices, 3167 of which were non-fixed. The optimization of the surface in the middle took 134 seconds. Then the mesh was refined and used as the starting point for the optimization leading to the surface on the right side, which was done in 791 seconds. In both cases the non-block optimization procedure was used.

The optimized surfaces shown in fig. 7 slightly bulge at the central parts. To improve the shape one can impose constraints by fixing some vertices in addition to the vertices, which determine the surface boundary. The surfaces shown in figures 9-11 were obtained with constraints; the meshes representing the surfaces on the left side are shown in fig. 8. The vertices, which determine the surface boundary, are marked with blue dots, while red dots denote the vertices fixed in addition.

The constraints for the junctions of three and five tubes in figures 9 and 10 fix the marked vertices along straight lines in two parallel planes. The resulting surfaces contain curves very close to line segments in two parallel planes. To obtain surfaces containing these line segments one might use hard constraints, i.e. interpolation consitions for the surface. However, the implementation of the construction with hard constraints is currently an open problem, and the soft constraints make it possible to obtain very close results.

The junctions of tubes in fig. 11 were obtained after fixing some mesh vertices so as to obtain a B-spline approximation of a part of a cylinder. Additionally the vertices of a polyline made of the mesh edges were fixed in order to outline the desired shape of the appropriate part of the surface.

Some information about the experiments, which gave the nine surfaces shown in figures 9-11, are given in tab. 1. The symbol n_V denotes the total number of vertices, n'_V is the number of vertices computed by the optimization procedure, n_E is the number of elements, n_B is the number of blocks and n_H denotes the number of Hessian evaluations during the optimization; if more than one block was used, then the first number in this column denotes the number of block Hessians (the matrices A_b, see Section 5), and the second number is the number of Hessian matrices for the entire system of equations computed in the final stage of optimization. Computation times are given in the last column. All these experiments were done using a sequential, single-thread implementation of the algorithm, and a computer with a 2.4GHz Pentium Core 2 processor.

fig.	n_V	n'_V	n_E	n_B	n_H	time (s)
9	1007	667	836	1	5	75
9	3743	3033	3428	1	5	563
9	14399	12969	13796	6	20, 1	2115
10	1677	1095	1392	1	4	129
10	6237	5055	5712	5	9, 1	525
10	23997	21615	22992	10	26, 1	3523
11	1871	925	1712	1	6	163
11	7199	4021	6896	1	6	810
11	28223	16741	27632	8	27, 3	4647

Tab. 1: Data sizes and computation times

7 Conclusion

Spline surfaces represented by meshes are truly smooth, as opposed to sets of planar triangles computed by most shape optimization procedures described in recent publications. If smoothness is a mark of quality, then the construction described in this paper may produce surfaces of very high quality; to see any undulations one has to use very sharp visualisation tools, like curvature maps, as these undulations are invisible on pictures rendered using a lighting model and no texture, and even with weaker tools, like reflection or highlight lines, it is hard to reveal them. Imposing constraints may cause the appearance of undulations, however when constraints are used with sufficient care, one can obtain also surfaces of such a quality.

There is no universal optimization criterion, and obviously one may need to modify the functional considered in this paper. The simplest possibility is to take advantage of the lack of affine invariance of the construction, by changing the coordinate system with a nonuniform scaling; this technique was used in [8] for tensor product patches. The rule is, that to decrease undulations in some direction one should scale down the surface in that direction. While this possibility greatly enhances the flexibility of the construction, it may be insufficient if the boundary conditions enforce rapid changes of the surface curvature. It seems that using curvilinear coordinate systems, specified with free-form deformations, might make it possible to adopt the optimization criterion to a really wide variety of applications. Implementing that is, however, another open problem.

Soft constraints may be used, to a certain extent, to simulate hard constraints in the following way: each polynomial patch of the surface corresponds to an internal facet, and each corner of the patch may be associated with a vertex of this facet. If interpolation conditions (i.e. hard constraints) are to be satisfied by the corners of the patches, then at first one can fix the vertices at the desired locations of the patch corners. The interpolation conditions will be satisfied by the surface optimized with these soft constraints with some error. For each patch corner one can find the position error vector, subtract it from the associated mesh vertex, and fix the vertex at the new location. Then the optimization may be repeated. After one or at most a few iterations the interpolation conditions should be satisfied with a very small error.

Having an optimized mesh, one can generate Bézier patches forming the surface. If the degree (9,9) of the special patches is too high, one can replace them by patches of lower degree, filling the polygonal hole in the surface, e.g. using the method described in [7], which produces biquintic patches forming a tangent plane-continuous surface with small discontinuities of curvature. Another possibility is using the Catmull-Clark algorithm to obtain a better approximation of the limiting surface represented by the optimized mesh.

An open source implementation of the construction described here is available from the author's web page [9]. This implementation is written in C; it was compiled using the GCC compiler and built into an interactive program running in the XWindow environment. It may be used directly to design surfaces, or it may serve as a reference in developing highly optimized, parallel industrial implementations.

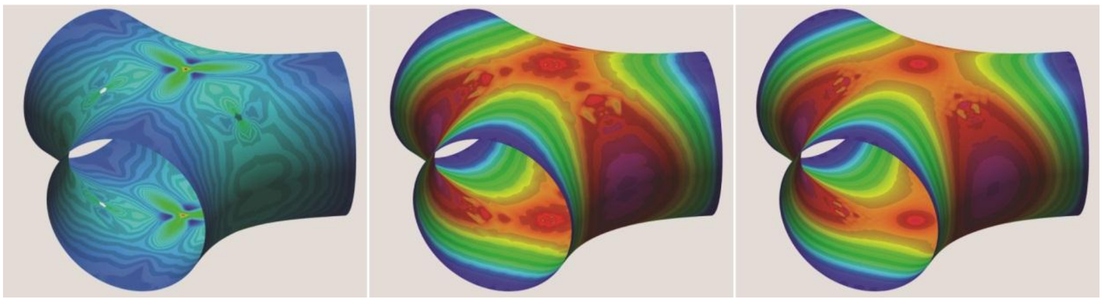

Fig.7: A surface obtained by iterating the Catmull-Clark and results of shape optimization

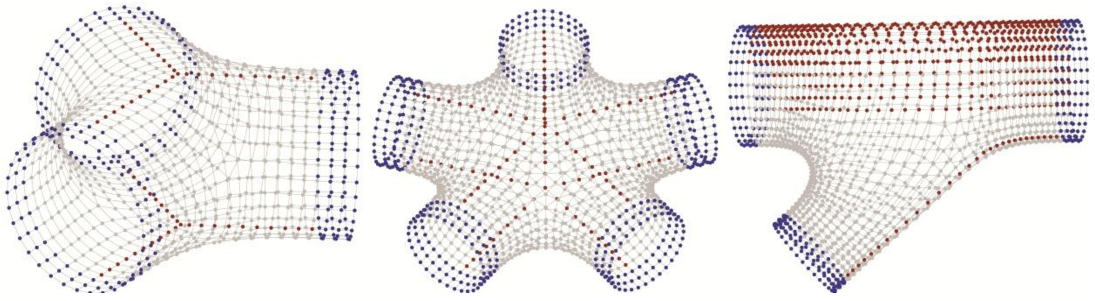

Fig. 8: Meshes with some vertices fixed to impose constraints

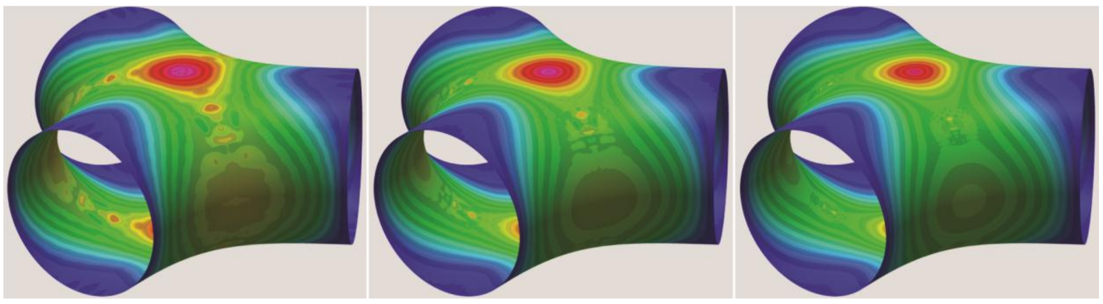

Fig. 9: A junction of three tubes

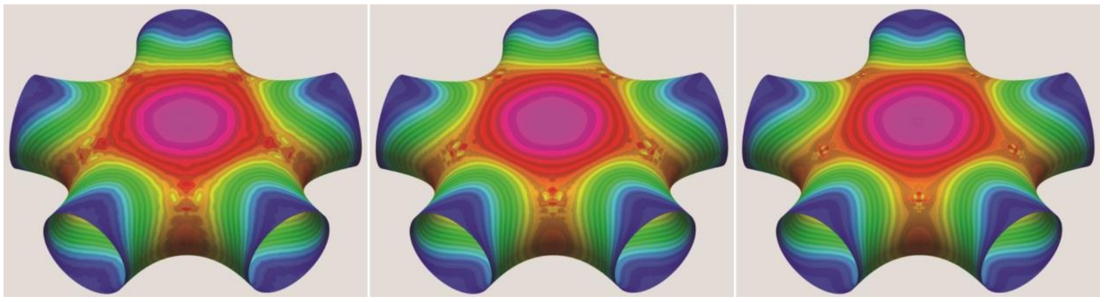

Fig. 10: A junction of five tubes

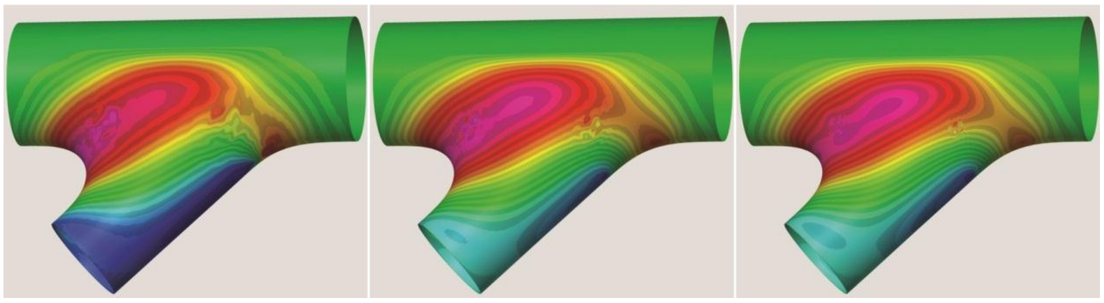

Fig. 11: Another junction of tubes

\parskip 0pt

References

[1] Catmull E., Clark J.: Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design,* 1978, 10(6), p. 350-355.

[2] Clarenz U., Diewald U., Dziuk G., Rumpf M., Rusu R.: A finite element method for surface restoration with smooth boundary conditions, *Computer Aided Geometric Design,* 2004, 21, p. 427-445.

[3] Doo D.W.H., Sabin M.A.: Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design,* 1978. 10(6), p. 356-360.

[4] Loop C., DeRose T.: Generalized B-spline Surfaces of Arbitrary Topology. *Computer Graphics,* 1990, 24(4), p. 347-356.

[5] Fletcher R.: *Practical Methods of Optimization.* Wiley, 2006.

[6] Kiciak P.: *Konstrukcje powierzchni gładko wypełniających wielokątne otwory.* Oficyna Wydawnicza Politechniki Warszawskiej, z. 159, Warszawa, 2007.

[7] Kiciak P.: Surfaces Filling Polygonal Holes with G^1 Quasi G^2 Continuity, *Machine Graphics & Vision,* Vol 19, No. 1, 2010, p. 63-96.

[8] Kiciak P.: Bicubic B-spline blending patches with optimized shape, *Computer-Aided Design,* 43 (2011), p. 133-144.

[9] Kiciak P.: *BSTools software package,* http://www.mimuw.edu.pl/˜przemek.