Filomat 29:3 (2015), 651–660
DOI 10.2298/FIL1503651R

Published by Faculty of Sciences and Mathematics,
University of Niš, Serbia
Available at: `http://www.pmf.ni.ac.rs/filomat`

Cutting Patterns of Membrane Structures

Svetozar R. Rančić[a], Milan Lj. Zlatanović[a], Nikola M. Velimirović[b]

[a]*Faculty of Science and Mathematics*
[b]*Faculty of Civil Engineering and Architecture*

Abstract. Membrane structures are very lightweight and highly optimized structures. Due to the constant stress state, strength of materials is used optimally. In order to prevent the occurrence of large deformations even for small external loads, membrane structures are designed as a double curved surfaces and are stabilized by applying prestress. Minimal surfaces has zero mean curvature and the basic advantage is that the stress at all points and directions is equal and there are no extreme stresses anywhere on the surface. Also have minimal area for the given contour, so the weight and amount of material is reduced to minimum, which make them suitable for application in architecture. Practical realization involve process of cutting pattern generation, which divide surfaces in parts that are developable surfaces. When patterns are assembled and prestressed they provide three-dimensional surface. Ideally, the cutting lines should follow the geodesics lines. We use geodesics as the shortest path between two points on a surface. In the article we give method for finding shortest paths on polygonal representations of surfaces follows continual Dijkstra paradigm which, on some conditions, can give improved accuracy on a computer despite the restriction of available memory and execution time.

1. Introduction

Membrane structures are very lightweight and highly optimized structures. Due to the constant stress state, strength of materials is used optimally. In the last decade, the use of membranes in the civil engineering is in expansion. With the development of new materials, a range of membrane structures has become wider. They are used as roofs of large spans, and as the elements for the cladding of buildings. Membrane structures are very special type of construction because of the properties of membranes. Membranes have no or very low bending stiffness. Therefore, their load carrying capacity is based on the tension stress tangential to the surface. Based on their properties, membrane structures can be identified as tensile structures, see [19]. In the case of tangential compression stresses, the membrane loses its stiffness and begins to wrinkle. External no tangential load on the surface of the membrane causes relatively large strains. The sensitivity of membrane structures on the external load is determined by their stiffness, which is mainly due to two factors: geometry and prestressing. In order to prevent the occurrence of large deformations even for small external loads, membrane structures are designed as a double-curved surfaces and are stabilized by applying prestress. These double-curved surfaces can be classified as sinclastic or anticlastic, depending

2010 *Mathematics Subject Classification*. Primary 52B05; Secondary 52B10
Keywords. (Membrane structures, Shortest path, Geodesic)
Received: 20 February 2015; Accepted: 17 March 2015
Communicated by Ljubica Velimirović and Mića Stanković
Research supported by MNTR grant 174012
Email addresses: `rancicsv@yahoo.com` (Svetozar R. Rančić), `zlatmilan@pmf.ni.ac.rs` (Milan Lj. Zlatanović), `velimirovic.nikola@gmail.com` (Nikola M. Velimirović)

on the sign of Gaussian curvature.There are various synthetic materials with different properties available for the realization of the membrane structures and can be divided into two groups: fabrics and foil.

Minimal surface is surface with a zero mean curvature. It can be defined as the surface which for a given contour has the minimal area, and therefore is suitable for application in civil engineering. Benefits of minimal surfaces as a form of prestressed membranes are great. The basic advantage is that the stress at all points and directions is equal and there are no extreme stresses anywhere on the surface, so there are no critical points which can cause loosening of the membrane. This indicates that the load carrying capacity of the membrane is equally used everywhere. Another reason for application is that the weight and amount of material is reduced to a minimum.

The three main processes involved in the design of the membrane structures are: form-finding, static load analysis and cutting pattern generation as it is presented in [7]. Form-finding represents the problem of determining a structural form, in most cases a surface, which is in equilibrium and satisfies the additional restriction. Static load analysis is performed to check whether a certain form of surfaces satisfies the final and temporary restrictions. Finally, finding a form of surfaces must be converted into a set of flat strips to produce, it is called cutting pattern generation. That process deals with the problem of defining the division of large surfaces, and ensures that these new surfaces are developmental. Cutting patterns are two-dimensional shapes that when assembled together and prestressed provide three-dimensional surface. The main problem in cutting pattern generation is how to partition structure on the strips. That procedure is also known as cutting line definition. Seam lines prediction, i.e. cutting line definition is very specific and difficult job. In ideal circumstances, if we neglect available width and length of the panels, material properties and some other characteristics, seam lines should coincide with geodesics that is proposed by [8]. Geodesic lines are lines which connect two points on the surface by the shortest possible route through surfaces. Using geodesics to define the seam line has the advantage of providing minimizing of the width of the fabric. In order to save material and be precise, and to avoid wrinkles on surface, the cutting line should follow the geodesic lines on surfaces.

2. Geodesics lines

Intuitively, a geodesic is the shortest arc between two points on a surface. A geodesic C on a surface S has the properties that at each point of C the principal normal coincides with the normal to S and the geodesic curvature vanishes identically.

Beltrami's formula for geodesic curvature. Given a curve $C : u = u(s), v = v(s)$ on a surface S: where s is arc length. Beltrami's formula for the geodesic curvature at point P of the curve is:

$$\kappa_g = \Big[\Gamma^2_{11}\Big(\frac{du}{ds}\Big)^3 + (2\Gamma^2_{12} - \Gamma^1_{11})\Big(\frac{du}{ds}\Big)^2\frac{dv}{ds} + (\Gamma^2_{22} - 2\Gamma^1_{12})\frac{du}{ds}\Big(\frac{dv}{ds}\Big)^2 - \Gamma^1_{22}\Big(\frac{dv}{ds}\Big)^3 + \frac{du}{ds}\frac{d^2v}{ds^2} - \frac{d^2u}{ds^2}\frac{dv}{ds}\Big]\sqrt{EG - F^2}, \quad (2.1)$$

where the Γ^i_{jk} are the Christoffel's symbols of the second kind, and E, F, G are coefficients of the first fundamental form of the surface S.

Definition 2.1. *A curve on a surface is a geodesic if and only if it's geodesic curvature is zero everywhere.*

Definition 2.2. *An orthogonal trajectory is a curve which cuts all the members of a given family of curves (or surfaces) at right angles.*

Theorem 2.1. *[9] If there exists on a surface an orthogonal system of geodesics, the surface is a developable or a plane.*

Definition 2.3. *Let S be a simple surface element defined by the equation*

$$r(u, v) = (x(u, v), y(u, v), z(u, v)),$$

Then a directed curve C on S represented parametrically in terms of arc length s by $u = u(s)$, $v = v(s)$ is a geodesic if and only if $u(s)$ and $v(s)$ satisfy the following differential equations:

$$\begin{aligned} &\frac{d^2u}{ds^2} + \Gamma^1_{11}\Big(\frac{du}{ds}\Big)^2 + 2\Gamma^1_{12}\frac{du}{ds}\frac{dv}{ds} + \Gamma^1_{22}\Big(\frac{dv}{ds}\Big)^2 = 0 \\ &\frac{d^2v}{ds^2} + \Gamma^2_{11}\Big(\frac{du}{ds}\Big)^2 + 2\Gamma^2_{12}\frac{du}{ds}\frac{dv}{ds} + \Gamma^2_{22}\Big(\frac{dv}{ds}\Big)^2 = 0 \end{aligned} \tag{2.2}$$

Definition 2.4. *The orthogonal trajectories of a family of curves in the plane have the property that the segments cut from the curves by any two of them are all equal.*

Conversely, if any two orthogonal trajectories of a family of curves on a surface cut equal segments from the curves of the family, the curves of the family are geodesics on the surface.

The orthogonal trajectories of a family of geodesic are known as geodesic parallels.

By Theorem 2.1, a family of geodesic parallels can consist of geodesics only if the surface is a developable or a plane.

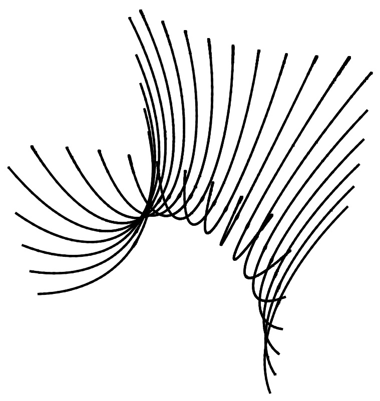


Fig. 1: Geodesic parallels of Sherk's surface

2.1. Geodesic mappings of surfaces

A *geodesic mapping* of surface S onto surface $\overline{S}$ is a diffeomorphism $f : S \to \overline{S}$ under which the geodesics of the surface S corresponds to the geodesics of the surface $\overline{S}$. Let g_{ij} and $\overline{g}_{ij}$ be the metrics of these surface, respectively. E. Beltrami was the first who considered and established problem of geodesic mapping of two surfaces $f : S \to \mathbb{E}^2$, where $\mathbb{E}^2$ is Euclidean two-plane. (see [2]). At the corresponding points M and $\overline{M}$ we can put

$$\overline{\Gamma}^i_{jk} = \Gamma^i_{jk} + P^i_{jk}, \quad (i, j, k = 1, 2), \tag{2.3}$$

where P^i_{jk} is *the deformation tensor* of the connection Γ^i_{jk} of S according to the mapping $f : S \to \overline{S}$.

A necessary and sufficient condition for the mapping f to be geodesic [9] is that the deformation tensor P^i_{jk} from (2.3) has the form

$$P^i_{jk} = \delta^i_j \psi_k + \delta^i_k \psi_j, \quad (i, j, k = 1, 2), \tag{2.4}$$

where

$$\psi_j = \frac{1}{3}\frac{\partial}{\partial x^j}\left(\ln\left|\frac{\sqrt{\overline{g}}}{\sqrt{g}}\right|\right), \qquad (g = \det\|g_{ij}\|,\ \overline{g} = \det\|\overline{g}_{ij}\|) \quad (i, j = 1, 2). \tag{2.5}$$

From previous equation it is easy to see that ψ_j is a gradient vector.

U. Dini solved more general problem, existence of geodesic mapping between two surfaces.

Theorem 2.2. (U. Dini, [6]) *There is a geodesic mapping between two nonisometric surfaces S and $\overline{S}$ if and only if their metrics have Lioville form:*

$$ds^2 = (U - V)(du^2 + dv^2) \text{ and } d\overline{s}^2 = \Big(\frac{1}{U} - \frac{1}{V}\Big)\Big(\frac{du^2}{U} + \frac{dv^2}{V}\Big), \tag{2.6}$$

where $U(u)$ and $V(v)$ are positive functions.

Example 2.1. *Let us consider geodesic mapping $f : S \to \overline{S}$, where*

$$S : \mathbf{r}(u, v) = \{u, v, uv\}, \qquad \overline{S} : \mathbf{r}(u, v) = \{x_1 + a_1u + b_1v, x_2 + a_2u + b_2v, x_3 + a_3u + b_3v\}.$$

How $\overline{\Gamma}^i_{jk} = 0$, from equation (2.5) we get

$$\psi_1 = -\frac{u}{3(1 + u^2 + v^2)}, \quad \psi_2 = -\frac{v}{3(1 + u^2 + v^2)}.$$

In this way we determined geodesic mapping of hyperbolic paraboloid onto plane.

Geodesic mappings of Riemannian spaces (surface is two-dimensional Riemannian space) were investigated by many authors, for example J. Mikeš, V. Kiosak and A. Vanžurová [12] and many others. The theory of geodesic mappings of two non-symmetric affine connection spaces were developed by S. Minčić, M. Stanković and Lj. Velimirović in the papers [13, 14, 20–22].

Definition 2.5. *A polyhedral surface P is a two-dimensional manifold consisting of finite or countable set F of topological triangles and intrinsic metric g such that:*

- *Any point $p \in P$ lies in at least one triangle $f \in F$;*
- *Each point $p \in P$ has a neighborhood that intersects only finitely many triangles $f \in F$;*
- *The intersection of any of two non-identical triangles $f, g \in F$ is either empty, or consists of a common vertex, or of a simple arc that is an edge of each of the two triangles;*
- *The intrinsic metric g is flat on each triangle, i. e. each triangle is isometric to a triangle in $\mathbb{R}^2$.*

Definition 2.6. *Let P be a polyhedral surface and $C \subset P$ a curve. Then C is a* ***straightest geodesic*** *on P if for each point $p \in C$ the left and right curve angles are equal.*

Definition 2.7. *Let C be a curve on a polyhedral surface P. Let θ be the total vertex angle and φ one of two curve angles of C at p. Then the* **discrete geodesic curvature** *of C at p is given by*

$$\kappa_g = \frac{2\pi}{\theta}\Big(\frac{\theta}{2} - \varphi\Big) \tag{2.7}$$

If we choose the other curve angle, geodesic curvature κ_g will change the sign.

For more details about straightest geodesics one can see [15, 16].

3. Shortest paths calculation

We present an optimal-time algorithm for computing an explicit representation of the shortest-path between two fixed points on the surface in R^3, source S and target T. It is supposed that surface has triangulated representation and the algorithm can work on both convex and non-convex surfaces. It is known that there are algorithms and methods for triangulating polygons, so we are able to transform surfaces from general polytopes representation to triangulated meshes. There is optimal time algorithm, see Amato et al. [1], which run in $O(n)$. This significantly simplifies problem. It is known that there exist optimal time algorithm for shortest paths on general polytopes, see Kapoor [11] improvement with $O(n \log^2 n)$ time over Chen and Han [4] $O(n^2)$ time. In article Kaneva and O'Rourke [10] is pointed out that Kapoor algorithm, which uses wave propagation, despite the advances is complex and that Chan and Han algorithm is feasible method for computing shortest paths on a polyhedral surface. Y. Schreiber gives $O(n \log n)$ algorithms in his PhD thesis [18] for convex and in [17] for general include nonconvex polyhedral surfaces.

If source or target point is not placed in vertices of the mesh, an additional two triangles subdivision is needed to fix points in a vertexes of the triangles.

On triangulated mesh we put the undirected graph such that in every node of the mesh we added a graph vertex, further vertexes are connected if respecting nodes of the mesh are connected. Added graph edge has weight equal to the Euclidian distance of nodes. Also we keep information of neighbouring: for every triangle on the mesh we know that the respecting vertexes in the graph are connected and forming a triangle and vice versa for every vertex we know to which triangles it belongs. This information are important for the algorithm steps where we need to update distances when we pop a vertex from the heap. Also for two neighboring vertexes which are edge-connected we can find one triangle (if it is placed on a border of a surface) or two triangles which share that edge. It is important in algorithm expansion phase when we are looking for shortest path length, also when we are reconstructing the shortest path.

3.1. *Shortest paths algorithm*

Dijkstra's algorithm [5] uses heap to store vertex weights, ie. paths to expand. In order to improve efficiency of heap operations we masked vertexes in an array named *heapind* with -1 if they are not reached yet and not pushed into heap. When we reached them we push into heap with at that moment founded path length. When removed from heap it obtain value $2 * N$, where N is the number of vertices, which means that it is processed. So -1 and $2 * N$ has special meaning. We keep track of the index in heap for every vertex in heap, so it also improve update operation for heap. Such heap operations demand nonstandard binary heap implementation. We keep count of processed nodes for every mesh triangle.

Shortest path for a node v other than starting node goes over some triangle in its part closest to the v. There are two possibilities for the last part of the shortest path: it is the edge of the triangle or it goes over opposite edge of the node v and intersects it. For the sake of simplicity of shortest path reconstruction, we maintain the indexes of previous nodes in two additional arrays *prevFirst* and *prevSecond*. If a node v has only one previous node in the *prevFirst*[v] we remember its index and in the *prevSecond*[v] we have -1. This is the obvious case with the neighbouring nodes of the start node, and also can appear for other nodes, especially for nonconvex meshes. If a node v has shortest path which goes over a triangle to whom it belongs and intersects triangle edge we remember indexes of the nodes on the ends of the edge.

Dijkstra's algorithm looking for shortest paths which goes over nodes and edges exclusively. Unlike the Dijkstra's algorithm, shortest path for polytopes, or meshes, can goes over polygons, or triangles not only over nodes and edges. Fast marching algorithm is very similar to the Dijkstra's algorithm, but it takes into account that path can goes over triangles which are parts of meshes, see Bronstein et al. [3]. Our algorithm is very similar to Fast marching algorithm. It differs in the update step and has an explicit and easy way for reconstructing shortest path. Here Fast marching algorithm needs to solve problem with obtuse triangles and it is recommended to add virtual connections to non-adjacent vertices, see [3]. We carefully examine all kind of triangles and contexts in expanding steps so we do not need special treatment for obtuse triangles. Therefore we can think of our algorithm as a modification of Fast marching algorithm.

The our shortest path algorithm, S (index s) and T (index t) are starting and ending node:

```
mark $d[s] = 0$, all other nodes with $d[v] = \infty$;
mark $heapind[s] = 0$, all other nodes with $heapind[v] = -1$;
initialize heap and push s into heap;
set $counter[\mathcal{T}] = 0$, for all mesh triangles $\mathcal{T}$
**do**
    $v$ = pop from heap;
    mark $heapind[v] = 2 * N$;
    **if** $v == t$ **then**
        **return**; // reconstruct the shortest path;
    else
        **for** each triangle $\mathcal{T}$ containing $v$;
            increment $counter[\mathcal{T}]$
            **for** each vertex $u$ belonging to $\mathcal{T}$ which is not processed
                update_or_push($u$);
```

The update_or_push operation is aimed to calculate for node u shortest path length according to information available at that moment. From heap it is just popped node v and node u is another node belonging to triangle $\mathcal{T}$. There are three cases according to the number of processed nodes of the triangle:

Case 1. One processed node: Node v is the first processed node, therefore we can update shortest path for another two nodes u according to

$$d[u] = min(d[u], d[v] + distance(v, u)). \tag{3.8}$$

If the distance $d[u]$ is updated v is the only one preceding node, so $prevFirst[u] = v$ and $prevSecond[u] = -1$.

Case 2. Two processed nodes: Node v is the second processed node, we previously processed first node. Now we should find and update shortest path length for the remaining third unprocessed node of the triangle $\mathcal{T}$. Let denote first processed node as A, second node with index v, as B and third node as C. According to the processing order, after removing from the top of the heap, we know that distance for A and B satisfy $d_A \leq d_B$. Now we have one virtual triangle which can be obtained with unfolding and rotating some mesh triangles to form shortest paths from starting node S to nodes A and B. This triangle $\triangle SAB$ and triangle $\mathcal{T} = \triangle ABC$, share the same edge AB.

Now we should examine all cases which can arose with mentioned two triangles. We have to take into account that quadrangle formed by them can be convex or nonconvex. There are 2 nonconvex and 6 convex cases for calculation:

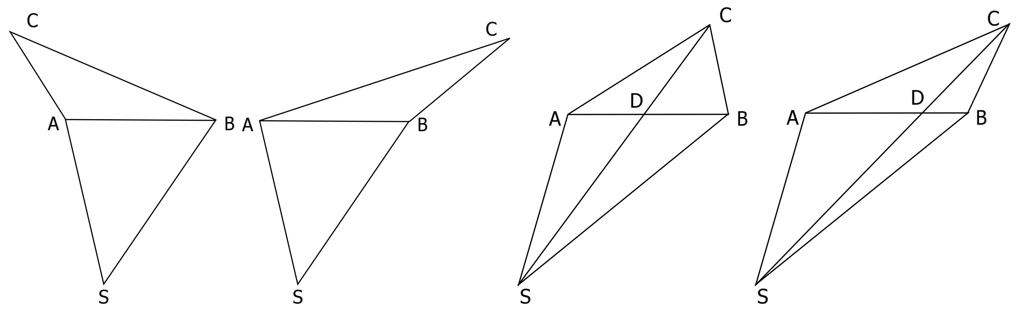


Fig. 2: First and second nonconvex; first and second convex examples.

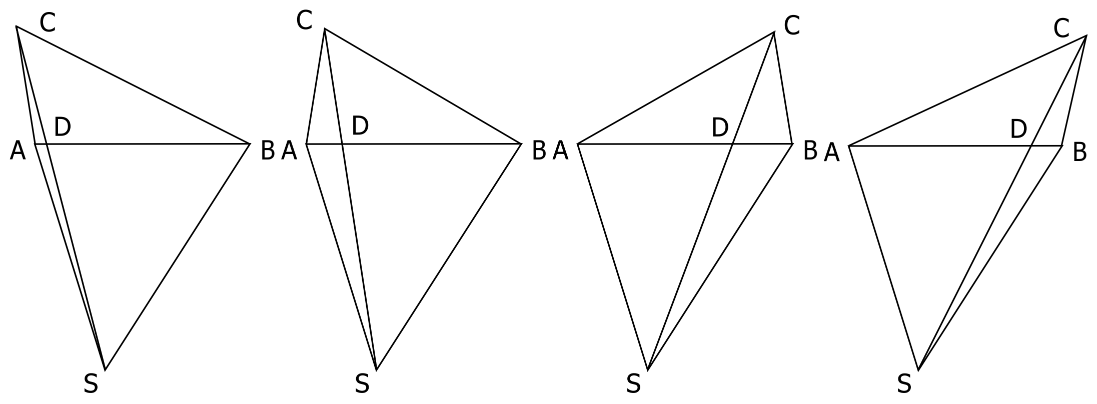


Fig. 3: Third, fourth, fifth and sixth convex examples.

For the first and second nonconvex example shortest path form node S to C can not cross the triangle $\mathcal{T} = \triangle ABC$. So shortest path for that cases goes over triangle edge AC for first, or over BC for second and is basically calculated as (3.8). Note that node u has one preceding node v. Here are shown cases where exist triangle $\triangle SAB$, but if $d[A] + distance(AB) = d[B]$ three points S, A and B are collinear. That case can appear but it is not shown on figures.

For the all six convex cases shortest path crosses the edge AB and path length can be calculated based on geometry of triangles $\triangle SAB$ and $\mathcal{T} = \triangle ABC$. For convexity examination and further shortest path length calculation we used cosine theorem, because we know the length of all edges in triangles $\triangle SAB$ and $\mathcal{T} = \triangle ABC$. After that calculation of possibly new distance value *newdist* we should do update for node C, $d[u]$ according to $d[u] = min(d[u], newdist)$. Also if the distance $d[u]$ is updated, we should update *prevFirst*$[u]$ and *prevSecond*$[u]$.

Case 3. For that case all triangle nodes are already processed.

From the above mentioned activities on nodes, heap insertion, update and removing, we can conclude that performance of the proposed algorithm is $O(n \log n)$.

3.2. Shortest paths reconstruction

Let us remind that we have calculated shortest paths length from starting node S to target node T, also calculating shortest paths length for all processed nodes which have removed from the heap and have $d[u] \leq d[t]$. We also have for every processed node in arrays *prevFirst* and *prevSecond* indexes of triangle nodes which contains last part of respecting shortest path.

The our shortest path reconstruction procedure where S (index s) and T (index t) are starting and ending node:

```
mark $prevNode = T$;
mark $prevIsNode$=true;
mark $prevTriangle$ = triangle consisting of $t$, and $prevFirst[t]$, and $prevSecond[t]$;
initialize empty $pathCollection$;
do
    if $prevTriangle$ contains $S$ then
        add segment $[prevNode, S]$ to $pathCollection$;
        return $pathCollection$;
    else
        if $prevIsNode$ then
            calculate intersection point $D$; // see figures Fig. 2. and Fig. 3.
            add segment $[prevNode, D]$ to $pathCollection$;
            update $prevNode = D$;
            update $prevTriangle$ = find triangle contains nodes $A$ and $B$ and not contains $C$;
            update $prevIsNode$=false;
        else
            find intersection point $E$ on $prevTriangle$ edges; // see figures for path reconstruction
            add segment [$prevNode, E$] to $pathCollection$;
            update $prevNode = E$;
            if $E$ is not mesh node then
                update $prevTriangle$ = find triangle contains edge with $E$ other than $prevTriangle$;
            else
                update $prevIsNode$=true;
                update $prevTriangle$ = triangle consisting of $prevNode$, and $prevFirst[t]$, and $prevSecond[t]$;
```

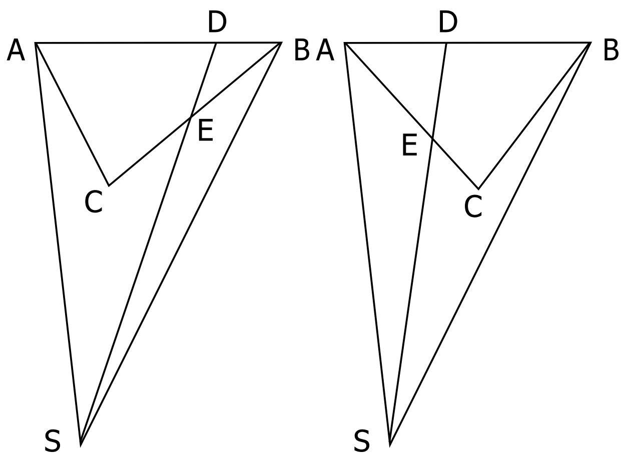
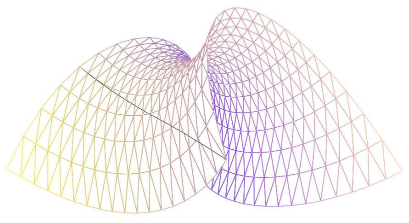


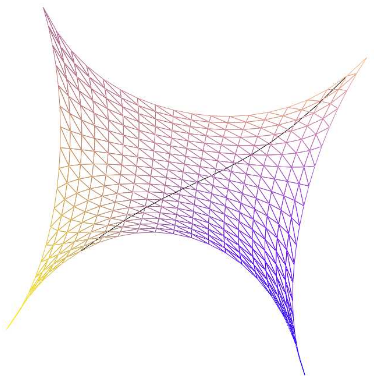
Fig. 4: Shortest path reconstruction.

In the path reconstruction procedure step calculate intersection point D is responsible for the backward calculation for convex examples. It also reconstructs the backward step for nonconvex examples, where D is point A or B.

4. Experimental results

We implemented our algorithm in C++. All calculations are done with double precision. We have tested it on several examples and did not find accuracy errors. Here are presented results for parametric surfaces:





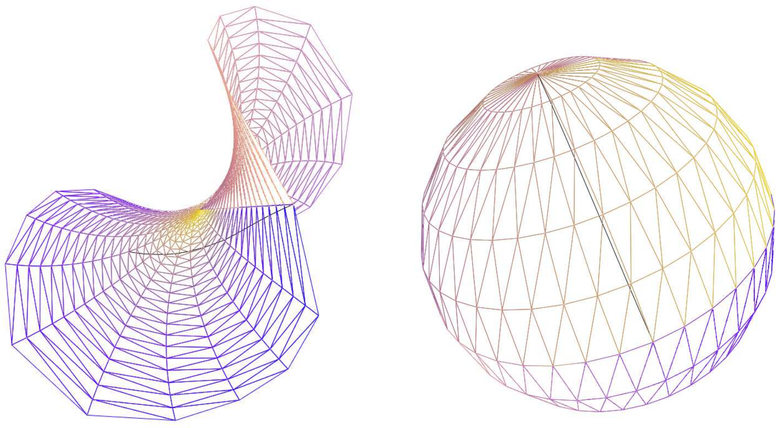


Fig. 5: Shortest path examples: Eneper, Eneper, Sherk and Sphere.

We tested accuracy of computation by calculating shortest path on a sphere with radius 5. For given two points the shortest path is $2.5 * \pi$. The first sphere calculation is done with initial dividing on 20 segments for parameter values, see figure Fig. 5. After that we are doing refinement of surface representation, by selecting triangles which contains shortest path. Then we are forming new surface by dividing every triangle on four new triangles using median value for both parameter values. Applying parametric sphere definition we obtain three new points for edge middle values. Connecting them with existing points we obtain four new triangles, which are closer to the sphere. After that we again calculate new shortest path, but only on newly obtained triangles. This procedure we repeat several times and the obtained shortest path length are shown in the table:

Refinement step	Number of Segments	Shortest path length
1	10	7.8195187101063
2	20	7.8453416023752
3	40	7.8518201061036
4	80	7.8534411570078
5	160	7.8538465088130
6	320	7.8539478523321
7	640	7.8539731885602
8	1280	7.8539795226405
9	2560	7.8539811061718
10	5120	7.8539815021035
11	10240	7.8539816014195
12	20480	7.8539816262986
13	40960	7.8539816335481
	Expected value	7.8539816339745

Table 1: Shortest paths length for refined surface.

The expected value rounded on 13 digits is 7.8539816339745 and we can see that applying refinement improves accuracy. We can conclude that accuracy of calculations is satisfiable. Also we need less memory resources to obtain result with the same accuracy. Without using refinement we need graph with $2 \times 40960 \times 2 \times 40960 \approx 6.7 \times 10^9$ nodes, instead of beginning 800 nodes and refinement adds $\approx 10^5$ nodes per one shortest path calculation.

5. Conclusion

In the refinement phase of our algorithm for shortest path calculation we obtain more finer and accordingly more precise polygonal representation of surface which enable more precise shortest path length calculation. We doing this calculations using computer resources like memory and processing time more economically, so we can obtain results with higher precision under the resource constraint.

References

[1] N. M. Amato, G. M. Goodrich, E. A. Ramos, A Randomized Algorithm for Triangulating a Simple Plygon in Linear Time, Discrete & Computational Geometry 26 (2), 245-265, 2001.

[2] E. Beltrami, Risoluzione del problema: riportari i punti di una superficie sopra un piano in modo che le linee geodetiche vengano rappresentate da linee rette, Ann. di Mat. **(1) 7**, 1865.

[3] A. M. Bronstein, M. M. Bronstein, R. Kimmel, Numerical Geometry of Non-Rigid Shapes, Springer, Series: Monographs in Computer Science, 2009.

[4] J. Chen, Y. Han, Shortest paths on a polyhedron, International Journal of Computational Geometry & Applications, (1996) 6:127-144.

[5] E. W. Dijkstra, A note on two problems in connection with graphs, Numerische Mathematik 1, (1959), 269-271.

[6] U. Dini, Sobre un problema che is presenta nella theoria generale delle representazioni geografiche di una superficie su di un'altra, Ann. Mat. **3(2a)**, 1869.

[7] B. Forster, M. Mollaert (eds), Europearn Design Guide for Tensile Surface Structures, TensiNet, Brussesl, 2004.

[8] L. Gründig, E. Moncriesff, P. Singer and D. Ströbel, High-performance cutting pattern generation od architectural textile structures, in E. Papadrakakis (Ed.), Proc IASS-IACM Fourth International Colloquium on Computation of Shell & Spartial Structures, JUne 4-7, 2000 Chanina-Crete, Greece, 2000.

[9] В. Ф. Каган, Основы теории поверхностей в тензором изложении, Государственное издательство технико-теоретической литературы Москва, 1948.

[10] B. Kaneva, J. O'Rourke, An implementation of Chen and Han's Shortest Paths Algorithm, 12th Canadian Conference on Computational Geometry, 2000.

[11] S. Kapoor, Efficient computation of geodesic shortest paths, Proceedings of 32nd Annual ACM Symposium of Theory of computing, 1999, 770-779.

[12] J. Mikeš, V. Kiosak, A. Vanžurová, Geodesic Mappings of Manifolds with Affine Connection, Olomounc, 2008.

[13] S. M. Minčić, M. S. Stanković, Lj. S. Velimirović, *Generalized Kahlerian Spaces*, Filomat 15 (2001), 167–174.

[14] S. M. Minčić, M. S. Stanković, Lj. S. Velimirović, *Generalized Riemannian spaces and spaces of non-symmetric affine connection*, Faculty of Science and Mathematics, Niš, 2013.

[15] K. Polthier, M. Schmies, Geodesic Flow on Polyhedral Surfaces, Data Visualization, Springer-Verlag (1999).

[16] K. Polthier, M. Schmies, Straightest Geodesics on Polyhedral Surfaces, Mathematical Visualization pp. 135-150, Springer-Verlag, Heidelberg (1998).

[17] Y. Schreiber, An Optimal-Time Algorithm for Shortest Paths on Realistic Polyhedra, Discrete Computational Geometry, 43, (2010), 21-53.

[18] Y. Schreiber, Eucledian Shortest Paths on Polyhedra in Three Dimsnsion, PhD thesis, Tel Aviv University, 2007.

[19] M. Seidel, Tensile Surface Structures: A Practical Guide to Cable and Membrane Consruction, Ernst & Sohn, Berlin, 2009.

[20] M. S. Stanković, *On a canonic almost geodesic mappings of the second type of affine spaces*, FILOMAT 13, (1999), 105-114.

[21] Stanković, M. S., Minčić, S. M., Velimirović, Lj. S., *On equitorsion holomorphically projective mappings of generalized Kählerian spaces*, Czechoslovak Mathematical Journal, 54(129), (2004), 701–715.

[22] M. S. Stanković, *Special equitorsion almost geodesic mappings of the third type of non-symmetric affine connection spaces*, Applied Mathematics and Computation, 244, (2014), 695-701.